Universidad de Murcia

Departamento de Ingeniería de la Información y las Comunicaciones

Optimización del rendimiento de Redes Inalámbricas Multi-salto mediante balanceo de carga considerando interferencias

Tesis Doctoral

Presentada por:
Juan José Gálvez García

Supervisada por:
Dr. Pedro Miguel Ruiz Martínez
Dr. Antonio Fernando Gómez Skarmeta

Murcia, Junio de 2011

D. Pedro Miguel Ruiz Martínez, Profesor Titular de Universidad del Área de Ingeniería Telemática y Director del Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, INFORMA:

Que la Tesis Doctoral titulada *"OPTIMIZACION DEL RENDIMIENTO DE REDES INALAMBRICAS MULTI-SALTO MEDIANTE BALANCEO DE CARGA CONSIDERANDO INTERFERENCIAS"*, ha sido realizada por D. Juan José Gálvez Gárcia, bajo la inmediata dirección y supervisión de los Doctores D. Pedro Miguel Ruiz Martínez y D. Antonio Fernando Gómez Skarmeta, y que el Departamento ha dado su conformidad para que sea presentada ante la Comisión de Doctorado.

En Murcia, a 30 de Junio de 2011

D. Pedro Miguel Ruiz Martínez

D. Pedro Miguel Ruiz Martínez, Profesor Titular del área de Ingeniería Telemática en el Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, AUTORIZA:

La presentación de la Tesis Doctoral titulada *"OPTIMIZACION DEL RENDIMIENTO DE REDES INALAMBRICAS MULTI-SALTO MEDIANTE BALANCEO DE CARGA CONSIDERANDO INTERFERENCIAS"*, realizada por D. Juan José Gálvez Gárcia, bajo mi inmediata dirección y supervisión, en el Departamento de Ingeniería de la Información y las Comunicaciones, y que presenta para la obtención del grado de Doctor Europeo por la Universidad de Murcia.

En Murcia, a 30 de Junio de 2011

D. Pedro Miguel Ruiz Martínez

D. Antonio Fernando Gómez Skarmeta, Catedrático de Universidad del área de Ingeniería Telemática en el Departamento de Ingeniería de la Información y las Comunicaciones de la Universidad de Murcia, AUTORIZA:

La presentación de la Tesis Doctoral titulada *"OPTIMIZACION DEL RENDIMIENTO DE REDES INALAMBRICAS MULTI-SALTO MEDIANTE BALANCEO DE CARGA CONSIDERANDO INTERFERENCIAS"*, realizada por D. Juan José Gálvez Gárcia, bajo mi inmediata dirección y supervisión, en el Departamento de Ingeniería de la Información y las Comunicaciones, y que presenta para la obtención del grado de Doctor Europeo por la Universidad de Murcia.

En Murcia, a 30 de Junio de 2011

D. Antonio Fernando Gómez Skarmeta

Resumen

Las redes inalámbricas multi-salto (RIMs) son colecciones de nodos autónomos capaces de establecer una red inalámbrica de forma dinámica sin necesidad de una infraestructura previa. Estas redes se caracterizan por el uso de comunicaciones multi-salto en las que, en general, un paquete de datos necesita recorrer múltiples enlaces inalámbricos (saltos) para alcanzar su destino. Las RIMs pueden admitir una topología dinámica y tienen restricciones significativas de recursos, como por ejemplo un canal inalámbrico compartido. Una de las características mas interesantes de las RIMs es que permiten desplegar una red de forma sencilla y rápida en un área relativamente extensa. Recientemente, una clase especial de RIMs llamadas Redes Malladas Inalámbricas (RMIs) han atraído mucha atención. En ellas, un subconjunto de nodos llamados enrutadores *mesh* son estacionarios y pueden formar una red troncal inalámbrica de alta velocidad. Estos nodos pueden equiparse con múltiples interfaces radio para permitir más transmisiones simultáneas usando varios canales de frecuencia no solapados. Se espera que las RMIs sean en un futuro una forma de desplegar una red de área extensa con bajo coste capaz de ofrecer servicios como conectividad a Internet. Uno de los escenarios de aplicación más relevantes para RMIs es por tanto el escenario de acceso a *puertas de enlace* (*gateways* en inglés) que pueden estar a varios saltos de un enrutador. En este escenario, la mayoría del tráfico se intercambia entre los enrutadores *mesh* y las puertas de enlace.

En RIMs, los nodos transmiten de forma inalámbrica y comparten el acceso al medio. Enlaces en el mismo dominio de colisiones no pueden transmitir de forma simultánea. La interferencia inalámbrica entre enlaces del mismo dominio de colisiones puede limitar drásticamente la capacidad de la red y

ser especialmente devastador para el rendimiento de flujos multi-salto. En estas redes, se pueden distinguir dos tipos fundamentales de interferencia: (i) interferencia intra-flujo, la cual es el resultado de interferencia entre enlaces del mismo camino seguido por un flujo de tráfico, y (ii) interferencia inter-flujo, que se refiere a transmisiones que interfieren de flujos que usan caminos diferentes. Debido a la capacidad limitada de las RIMs, la utilización eficaz de esta capacidad es esencial. El balanceo de carga es una técnica apropiada para optimizar la utilización de recursos, y consiste en distribuir una carga de trabajo a través de múltiples recursos. En una red, esta técnica puede contribuir a un aumento del rendimiento e igualdad de los flujos de tráfico. Para balancear carga en una RIM, es necesario distribuir la carga a través de diferentes dominios de colisiones, reduciendo de esta forma el número de transmisiones en un mismo dominio. Esto se consigue evitando interferencia intra-flujo e inter-flujo. En otras palabras, para balancear carga de forma eficaz son necesarias estrategias que consideran interferencias. Además, ya que el tráfico en una RIM varía con frecuencia y de forma impredecible, es necesario balancear la carga instantánea de la red para obtener un rendimiento óptimo.

El balanceo de carga mediante enrutamiento se puede conseguir distribuyendo el tráfico a través de caminos que no interfieren. Si se consideran redes multi-canal, el enrutamiento también debe seleccionar caminos con baja interferencia intra-flujo. La asignación de canales implica asignar canales de transmisión a interfaces radio, lo cual determina el canal usado por un enlace para transmitir. Dos enlaces transmitiendo en canales ortogonales no interfieren. Como el conjunto de radios y canales es finito, la interferencia no puede ser eliminada de forma completa, y una decisión de asignación de canales debe elegir con cuidado qué canales asignar a qué interfaces. La asignación de canales se puede usar para balancear carga, mitigando la interferencia intra-flujo e inter-flujo en regiones sobrecargadas, reduciendo de esta forma las transmisiones simultáneas en dominios de colisión. Esta tesis doctoral propone mecanismos de balanceo de carga considerando interferencias, basados en enrutamiento y, cuando se trate de redes multi-canal, asignación de canales.

El diseño de protocolos de red y soluciones sensibles a la carga conlleva un reto importante, debido al problema de evitar inestabilidad de la red. La carga puede variar con frecuencia, y esto puede llevar a diversos problemas como sobrecarga alta para comunicar actualizaciones del estado de carga a través de la red, comportamiento no-convergente del protocolo, o inestabilidad de la solución. Si un protocolo no puede converger a tiempo bajo cambios frecuentes de tráfico, puede perturbar la operación de la red. Desde la perspectiva del enrutamiento, la sensibilidad a la carga puede llevar a inestabilidad de las rutas (oscilaciones frecuentes de rutas), no-convergencia o convergencia lenta, y sobrecarga alta para comunicar actualizaciones de rutas en la red. De forma similar, un protocolo de asignación de canales debe garantizar una asignación estable. Si las condiciones de tráfico no varían o lo hacen de forma mínima, el protocolo debe mantener una asignación estable. Además, debe garantizar un tiempo bajo de convergencia, reducir la sobrecarga necesaria para la coordinación entre nodos, y evitar reajustes de canal frecuentes e innecesarios que introducen latencia.

En redes de un solo canal, es posible balancear la carga usando enrutamiento que evite o mitigue la interferencia inter-flujo. Medir interferencia entre enlaces, sin embargo, no es trivial, y una alternativa más fácil consiste en estimar interferencia basándose en la distancia entre nodos. Este trabajo desarrolla la métrica *Path Spatial Distance* (PSD) que mide la separación espacial entre un camino y un conjunto de nodos. Uno de los objetivos principales de PSD es ayudar a una solución de enrutamiento a seleccionar caminos que no interfieren. Una ventaja importante de la métrica es que no requiere información de localización de nodos (como coordenadas geográficas) sino que sólo necesita conocer la distancia en saltos entre nodos. Se demuestra que esta distancia tiene una fuerte correlación con la distancia Euclídea en muchos escenarios de red, con lo que proporciona una estimación buena de la distancia entre nodos en el espacio. Basándose en la métrica PSD, este trabajo propone un algoritmo heurístico, llamado *Spatially Disjoint Path Calculation algorithm* (SD-PCA), para encontrar dos caminos con máxima separación espacial que conectan a dos nodos de un grafo. Un nodo de la red con conocimiento del grafo topológico usa SD-PCA

para encontrar caminos separados espacialmente hacia un destino. Basado en lo anterior, se desarrolla un protocolo multi-camino reactivo para RIMs, llamado *Spatially Disjoint Multipath Routing* (SDMR), que permite a un nodo descubrir el grafo topológico bajo demanda (es decir, sólo cuando las rutas son necesarias) y, usando el algoritmo SD-PCA, calcular un par de caminos separados espacialmente hacia un destino. El uso de caminos separados espacialmente puede proporcionar otros beneficios adicionales en RIMs además de la reducción de interferencia, incluyendo tolerancia a fallos en situaciones de fallo regional, y seguridad adicional mediante la distribución de una transmisión a través de caminos separados espacialmente para evitar intercepción. Resultados de simulaciones muestran que SDMR puede descubrir la topología bajo demanda de forma eficaz, requiriendo menos sobrecarga de control en comparación con OLSR. Además, los resultados muestran que, usando la métrica PSD y el algoritmo SD-PCA, el protocolo descubre caminos con alta separación espacial, a diferencia de otros protocolos multi-camino como AOMDV.

En el escenario de acceso a puertas de enlace en RMIs, la mayoría del tráfico va dirigido a/desde las puertas de enlace. El conjunto de nodos (o alternativamente flujos) al que sirve una puerta de enlace se conoce como su dominio. En redes de un solo canal, no es posible balancear la carga de forma adecuada dentro del dominio de una puerta de enlace, porque cada flujo del dominio compite por acceso en la zona que rodea la puerta de enlace. Esta región representa un cuello de botella inevitable. Si la red tiene múltiples puertas de enlace, es sin embargo posible balancear la carga entre ellas. En otras palabras, la carga de la red se puede balancear asignando tráfico dinámicamente a puertas de enlace. Es importante balancear carga entre puertas de enlace para evitar regiones sobre-utilizadas y regiones infra-utilizadas. Se puede dar fácilmente un desbalanceo de carga debido a numerosos factores, incluyendo demandas de tráfico heterogéneas, tráfico variable en el tiempo, y un número desigual de nodos servidos por las puertas de enlace. Esto puede llevar al uso ineficiente de la capacidad de la red, degradación del rendimiento y desigualdad entre flujos de diferentes dominios. Por otro lado, un balanceo de carga arbitrario puede perjudicar el

rendimiento. La asociación de nodos a puertas de enlaces se debe elegir con cuidado, para evitar interferencia grave intra-flujo e inter-flujo. Este trabajo propone un sistema de selección de puertas de enlace centralizado para RMIs llamado *Gateway Load-Balancing* (GWLB) que, dadas las condiciones actuales de la red, selecciona puertas de enlace de forma eficiente para cada nodo o flujo. GWLB es un algoritmo en-línea que balancea la carga instantánea y al mismo tiempo evita la inestabilidad de la red. La solución no sufre problemas de convergencia ya que se calcula de forma centralizada. La inestabilidad de la selección de puertas de enlace no ocurre, porque es fácil impedir que un nodo cambie de puerta de enlace hasta que transcurra un tiempo. GWLB no introduce ninguna sobrecarga para determinar la carga actual y responde con prontitud, gracias al hecho de que la solución se puede recalcular de forma periódica con una frecuencia muy alta. Para impedir balanceo de carga perjudicial debido a interferencias graves, GWLB tiene en cuenta la interferencia potencial generada por la selección de puertas de enlace usando la métrica PSD, basándose en conocimiento de la topología de red. Otra ventaja importante de GWLB es que balancea tráfico a nivel de flujos TCP, y por tanto mejora el rendimiento e igualdad de los flujos. Por todas estas razones, es válido para implementación en escenarios de RMI prácticos y dinámicos. Hemos demostrado mediante numerosos experimentos que GWLB puede balancear carga eficazmente entre puertas de enlace mientras evita el uso de caminos perjudiciales. Las simulaciones llevadas a cabo en ns-3 demuestran la eficacia de GWLB, y su ventaja sobre propuestas anteriores. En particular, puede alcanzar un incremento en la tasa media de datos de un 128 % sobre la estrategia de la puerta de enlace más cercana. GWLB se distancia de otras soluciones de manera notable cuando la congestión o desbalanceo entre dominios aumenta.

En RMIs multi-radio multi-canal, los nodos disponen de múltiples interfaces radio, cada una de las cuales se puede sintonizar a uno de varios canales de frecuencia no solapados. Esto proporciona un aumento sustancial en las opciones de balanceo en el escenario de acceso a puertas de enlace. La asignación de canales permite eliminar o mitigar interferencia en la región que rodea a una puerta de enlace, haciendo posible balancear carga dentro de

su dominio. En este escenario, se puede balancear carga mediante enrutamiento dentro del dominio a través de la selección de caminos con baja interferencia intra-flujo e inter-flujo. Por otro lado, la asignación de canales se puede usar para dar más capacidad a los enlaces que llevan más tráfico. Lo anterior determina que haya una interdependencia entre enrutamiento y asignación de canales y, cuando son optimizados de forma conjunta, es posible mejorar de forma notable la utilización de la red y el rendimiento e igualdad de los flujos. Este trabajo propone el sistema *Load-Balancing and Channel Assignment* (LBCA) para RMIs multi-radio multi-canal. LBCA realiza enrutamiento y asignación de canales para optimizar el rendimiento de un conjunto de flujos TCP servidos por una puerta de enlace, dentro de la RMI. LBCA es un sistema centralizado que, dadas las condiciones de red y tráfico actuales, asigna canales a interfaces radio y asigna una ruta a cada flujo. El sistema es capaz de adaptarse rápidamente a condiciones cambiantes, y evitar inestabilidad de la red. La solución de LBCA converge, y lo hace rápidamente, gracias a la baja complejidad temporal de los algoritmos propuestos de enrutamiento y asignación de canales. La inestabilidad de rutas no ocurre porque es fácil impedir que un flujo cambie de rutas hasta que transcurra un tiempo determinado. De forma similar, para impedir el reajuste frecuente de canales, el algoritmo propuesto tiene en cuenta explícitamente la asignación previa de canales para calcular una nueva, de tal forma que limita el número de reajustes de canal. Este trabajo muestra que la re-asignación de canales con un periodo de pocos segundos es posible en escenarios de acceso a puerta de enlace. Para difundir la asignación de canales, LBCA implementa un protocolo novedoso que rápidamente distribuye mensajes de cambio de canal a través de la red con baja sobrecarga. Este protocolo de distribución no requiere una interfaz de cada nodo sintonizada a un canal común, a diferencia de otros protocolos existentes. Hemos mostrado mediante abundantes evaluaciones la eficacia de LBCA. Hemos llevado a cabo simulaciones en ns-3 con un periodo de adaptación de un segundo, mostrando que el protocolo puede adaptarse con frecuencia muy alta. Los experimentos muestran que la re-asignación de canales sólo afecta

a entre el 25 % y 50 % de los enlaces. Esto significa que adaptarse a condiciones de tráfico variables sólo requiere re-ajustar el canal de un número limitado de enlaces. Cuando se realiza enrutamiento y asignación de canales conjunto, los resultados en escenarios estáticos muestran un aumento de la tasa mínima de los flujos de hasta un 84 % y tasa media de hasta un 37 % en comparación con una estrategia de camino más corto y una asignación de canales que no tiene en cuenta el tráfico. LBCA es por tanto eficaz impidiendo la inanición de los flujos y aumentando la igualdad. En escenarios dinámicos, la ventaja de LBCA es aún mayor. Se muestra que la frecuencia de adaptación es un factor determinante en escenarios altamente dinámicos. Con un periodo de adaptación de un segundo, los resultados muestran un aumento de la tasa mínima de los flujos de hasta 175 % comparado con LBCA con un periodo de adaptación de treinta segundos, y un aumento de la tasa media de hasta 63 %. La duración de los flujos con un periodo de treinta segundos es hasta 164 % mayor.

University of Murcia

Department of Information and Communications Engineering

Throughput optimization of Multi-hop Wireless Networks through interference-aware load balancing

PhD thesis

Author:

Juan José Gálvez García

Thesis advisors:

Dr. Pedro Miguel Ruiz Martínez

Dr. Antonio Fernando Gómez Skarmeta

Murcia, June 2011

Abstract

Multi-hop wireless networks (MWNs) are collections of autonomous nodes capable of dynamically establishing a wireless network without the need of an existing infrastructure. These networks are characterized by multi-hop communication where, in general, a packet needs to traverse multiple wireless links (hops) to reach its intended destination. They can support a dynamic topology and have significant resource constraints, such as a shared wireless channel. One of the most interesting characteristics of MWNs is that they permit deploying a network easily and quickly over a relatively large area. A special class of MWNs, called Wireless Mesh Networks (WMNs) has recently attracted much attention. In these networks, a subset of nodes called mesh routers are stationary and capable of forming a high-speed wireless backbone. These nodes can be equipped with multiple radio interfaces to allow more simultaneous transmissions using several non-overlapping frequency channels. WMNs are envisioned to provide a cost-effective way of deploying a wide-area network and offer services such as Internet connectivity. One of the most relevant application scenarios of WMNs is thus the gateway access scenario, where nodes access an external network (e.g. the Internet) through a gateway which can be several hops away. In this scenario, most traffic is exchanged between mesh routers and gateways.

In MWNs, nodes transmit wirelessly and share access to the medium. Links in the same collision domain cannot transmit simultaneously. Wireless interference of links in the same collision domain can severely limit the capacity of the network and be specially devastating to the performance of multi-hop flows. In these networks, two fundamental types of interference can be distinguished: (i) intra-flow interference, which is a result of interference

between links in the same path followed by a traffic flow, and (ii) inter-flow interference, which refers to the transmissions of flows using different paths interfering with each other. Due to the limited capacity of MWNs, it is essential to effectively utilize this capacity. Load balancing is a technique suited to optimizing resource utilization, which consists in distributing a workload across multiple resources. In a network, this technique can contribute to increased throughput and fairness of traffic flows. To balance the load in a MWN, it is necessary to distribute load across different collision domains, thus reducing the number of transmissions in the same collision domain. This is achieved by avoiding intra-flow and inter-flow interference. In other words, interference-aware strategies are necessary to effectively balance load. In addition, because traffic in a MWN can vary frequently and unpredictably, it is necessary to balance the instantaneous load of the network to achieve optimum performance.

Load balancing through routing can be achieved by distributing traffic along non-interfering paths. If multi-channel networks are considered, routing must also select paths with low intra-flow interference. Channel assignment involves allocating transmission channels to radio interfaces, which determines the channel used by a link to transmit. Two links transmitting in orthogonal channels will not interfere. Because the set of radios and channels is finite, interference cannot be completely avoided and a channel assignment decision must carefully choose which channels to assign to which interfaces. Channel assignment can be used to balance load, by reducing intra-flow and inter-flow interference in loaded regions, thus reducing simultaneous transmissions in collision domains. This thesis proposes interference-aware mechanisms for load-balancing through routing and, when considering multi-channel networks, channel assignment.

Load-sensitivity introduces a major challenge in the design of network protocols, due to the problem of avoiding network instability. Load can vary frequently, and this can lead to several issues such as high overhead to communicate load status updates through the network, non-convergent protocol

behavior or solution instability. If a protocol cannot converge on time under frequent traffic changes, it may disrupt network operation. From the routing perspective, load-sensitivity can lead to route instability (frequent route oscillations), non-convergence or slow convergence, and high overhead to communicate route updates through the network. In a similar way, a channel assignment protocol has to guarantee channel allocation stability. If traffic conditions do not change or change minimally, the protocol must maintain a stable channel assignment. In addition, it must guarantee low convergence time, deal with overhead resulting from frequent node coordination, and avoid frequent and unnecessary channel re-adjustments that incur delay.

In single-channel networks, it is possible to balance load through interference-aware routing by avoiding or mitigating inter-flow interference. Measuring link interference, however, is not trivial, and an easier alternative consists in estimating interference based on the distance between nodes. This work develops the Path Spatial Distance (PSD) metric to measure the spatial separation of a path to a set of nodes. One of the main goals of PSD is to assist a routing solution in selecting non-interfering paths. An important advantage of this metric is that it does not require location information of nodes (e.g. geographical coordinates) and instead only relies on the distance in hops between nodes. This is shown to correlate strongly with Euclidean distance in many network scenarios, and so provides a close estimate of the distance in space between nodes. Based on the PSD metric, this work proposes a heuristic algorithm, called Spatially Disjoint Path Calculation algorithm (SD-PCA), to find two paths with maximal spatial separation connecting two nodes in a graph. SD-PCA can be used by a node in the network with knowledge of the topology graph to find spatially disjoint paths to a destination node. A reactive multi-path routing protocol for MWNs, called Spatially Disjoint Multipath Routing (SDMR), is then developed that permits nodes to discover the topology graph on-demand (i.e. only when routes are needed) and, using the SD-PCA algorithm, calculate a pair of spatially separated paths to a destination. The use of spatially

separated paths can provide additional benefits in MWNs besides interference avoidance, including increased reliability by gaining fault tolerance to situations of regional failures, and increased security by distributing a transmission over spatially disjoint paths to avoid interception. Simulation results show that SDMR can effectively discover the topology on-demand, requiring less control overhead than OLSR. In addition, results show that, using the PSD metric and SD-PCA algorithm, the protocol discovers paths with high spatial separation, in contrast to other proposed multipath protocols like AOMDV.

In the gateway access scenario in WMNs, most traffic is directed to/from gateways. The set of nodes (or alternatively flows) served by a gateway is referred to as its domain. In single-channel networks, it is not possible to adequately balance the load inside a gateway domain, because every flow in the domain ultimately contends in the region surrounding the gateway. This region constitutes an unavoidable network bottleneck. If the network has multiple gateways, it is possible however to balance load between gateways. In other words, the network load can be balanced by dynamically assigning traffic to gateways. It is important to balance load between gateways to avoid over-utilized and underutilized regions. Load imbalance can easily occur due to a number of factors, such as heterogeneous traffic demands, time-varying traffic and uneven number of nodes served by gateways. This can lead to inefficient use of network capacity, throughput degradation and unfairness between flows in different domains. On the other hand, arbitrary load-balancing can hurt performance. Association of nodes to gateways must be chosen carefully, to avoid severe intra-flow and inter-flow interference. This work proposes a centralized gateway selection system for WMNs called Gateway Load-Balancing (GWLB) which, given the current network conditions, efficiently selects gateways for every node or flow. GWLB is an on-line algorithm that can balance the instantaneous load of the network while avoiding network instability. Because the solution is calculated centrally, it does not suffer from convergence issues. Gateway selection instability, also known as gateway flapping, does not occur, because a node can be prevented from switching gateways until a specified time elapses.

GWLB does not introduce any overhead to determine the current network load and is highly responsive, owing to the fact that the solution can be recalculated periodically with a very high frequency. To prevent harmful load-balancing due to severe interference, the potential interference generated by gateway selection is taken into account based on knowledge of the network topology, using the PSD metric. Another important advantage of GWLB is that it balances traffic at the TCP flow level, thus improving the throughput and fairness of flows. For these reasons, it is suitable for implementation in practical and dynamic WMN scenarios. Numerous tests show that GWLB can effectively balance load between gateways while avoiding the use of harmful paths in the network. The simulations conducted in ns-3 prove the effectiveness of GWLB, and its advantage over previously proposed schemes. In particular, it achieves an average flow throughput gain of 128% over the nearest gateway strategy. GWLB specially distances itself from other solutions when congestion or load imbalance between domains increases, showing that the protocol can cope more effectively in these situations.

In multi-radio multi-channel WMNs, nodes can be equipped with multiple radio interfaces, each of which can be tuned to one of several non-overlapping frequency channels. This provides substantially increased balancing options in the WMN gateway access scenario. Channel assignment can be used to eliminate or mitigate interference in the region surrounding a gateway, making it possible to balance load inside a gateway domain. In this scenario, load-balancing routing can balance the load in the gateway domain by selecting paths with low intra-flow and inter-flow interference. Channel assignment, on the other hand, can be used to give more capacity to links carrying more traffic. The above determines that there is an interdependency between routing and channel assignment, and when optimized jointly, it is possible to substantially improve network utilization, flow throughput and fairness. This work proposes the Load-Balancing and Channel Assignment (LBCA) system for multi-radio multi-channel WMNs. LBCA performs routing and channel assignment to optimize the performance of a set of TCP flows served by a gateway inside the WMN. LBCA

is a centralized on-line system which, given the current traffic and network conditions, allocates channels to radio interfaces and assigns a route to every flow, improving flow performance as a result. The system is capable of rapidly adapting to changing conditions while avoiding network instability. The solution of LBCA is guaranteed to converge, and it does so quickly, owing to the low time complexity of the proposed routing and channel assignment algorithms. Route instability does not occur because a flow is easily prevented from switching routes until a specified time elapses. In a similar way, to prevent frequent channel re-adjustments in the network, the proposed channel assignment algorithm explicitly takes into account the previous channel assignment when calculating a new channel allocation, and limits the number of edge channel re-adjustments. This work shows that channel re-assignment with a period of a few seconds is realizable in gateway access scenarios. To disseminate channel assignments, LBCA implements a novel protocol to quickly distribute channel allocation messages through the network with low overhead. This distribution protocol does not require one interface in every node tuned to a common channel, as opposed to other existing protocols. Extensive evaluations show the effectiveness of LBCA. Simulations have been conducted in ns-3 with an adaptation period of one second, showing that the protocol can adapt to the current network conditions with a very high frequency. Tests have shown that channel re-assignment only affects between 25% and 50% of edges when using the proposed load-balancing routing strategy. This means that adapting to varying traffic conditions only requires re-adjusting the channel of a limited set of edges. When performing joint routing and channel assignment, results in (mostly) static scenarios have shown an increase in minimum flow rate of up to 84% and median flow rate of up to 37% compared to a shortest path routing strategy and traffic-unaware channel assignment. LBCA is thus effective at preventing flow starvation and increasing flow fairness. In dynamic scenarios, the advantage of LBCA is even greater. It is shown that adaptation frequency is a determining factor in highly dynamic networks scenarios. When the adaptation period is one second, results show an increase in minimum flow rate of up to 175% compared with LBCA when the

adaptation period is thirty seconds, and median flow rate up to 63%. And the duration of flows with a thirty second adaptation period is up to 164% higher.

To

Noelia, my wife,
Clara, my daughter, and to
my parents.

Acknowledgements

I am grateful to my advisor Pedro M. Ruiz for his guidance during all these years, and holding me to high standards of quality and accuracy. I am also grateful for his opinions and advice.

I want to thank Antonio for helping and supporting me to embark on this endeavor.

I am most grateful to my wife and family for their love and support. Without them I probably wouldn't have been able to finish this work.

Agradecimientos

Doy las gracias a mi tutor Pedro M. Ruiz por guiarme durante todos estos años, y por ayudarme a alcanzar y mantener niveles altos de calidad y rigor. Agradezco también sus opiniones y consejos.

Quiero agradecer a Antonio por su ayuda y apoyo para poder embarcar en este proyecto.

Sobre todo estoy agradecido a mi esposa y mi familia por su cariño y apoyo. Sin ellos probablemente no hubiera podido alcanzar el final.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ACK	Acknowledgment
AODV	Ad hoc On Demand Distance Vector routing
AOMDV	Ad hoc On demand Multipath Distance Vector routing
BER	Bit error rate
CA	Channel assignment
CBR	Constant bit rate
CDMA	Code division multiple access
CSMA	Carrier sense multiple access
CSMA/CA	Carrier sense multiple access with collision avoidance
CTS	Clear to send
DARPA	Defense Advanced Research Projects Agency
DCF	Distributed Coordination Function
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
ETT	Expected Transmission Time
ETX	Expected Transmission Count
GWLB	Gateway Load Balancing protocol
iAWARE	Interference Aware routing metric
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ILP	Integer linear programming
LACA	Load aware channel assignment
LBCA	Load balancing and channel assignment algorithm
LBR	Load balancing routing algorithm
LPT	Longest processing time
LS	List scheduling
MAC	Medium Access Control
MANET	Mobile Ad hoc Network
MIC	Metric of Interference and Channel switching
MINLP	Mixed integer nonlinear programming
MPR	Multi point relay
MWN	Multi hop Wireless Network
NCO	Normalized control overhead
NLP	Nonlinear programming
OLSR	Optimized Link State Routing
PDR	Packet delivery ratio
PER	Packet error rate
PSD	Path Spatial Distance metric
RERR	Route error
RF	Radio Frequency
RREP	Route reply
RREQ	Route request
RTS	Request to send
RTT	Round trip time
SD-PCA	Spatially Disjoint Path Calculation algorithm
SDMR	Spatially Disjoint Multi path Routing protocol
SINR	Signal to interference noise ratio
SMR	Split Multipath Routing

SNR	Signal to noise ratio
TC	Topology control
TCP	Transmission Control Protocol
TDMA	Time division multiple access
TORA	Temporally ordered routing algorithm
TREP	Topology reply
TREQ	Topology request
UDP	User Datagram Protocol
VANET	Vehicular Ad hoc Network
WCETT	Weighted Cumulative Expected Transmission Time
WMN	Wireless Mesh Network
WRP	Wireless Routing Protocol
WSN	Wireless Sensor Network

Chapter 1

Introduction

This work presents solutions to improve the performance of traffic flows in Multi-hop Wireless Networks, under practical and dynamic scenarios. The protocols that have been designed do not assume a priori knowledge of traffic patterns and can promptly adapt to current traffic conditions, even under highly dynamic conditions. The main goal is to balance load, optimizing the network utilization and facilitating fair access of TCP flows to resources, improving flow throughput and fairness.

This chapter first gives a brief introduction to Multi-hop Wireless Networks and Wireless Mesh Networks, and outlines the problems that are addressed in the thesis. After explaining the main goals that were set out for this work, it discusses the methodology used to reach them and the resulting contributions. Finally, it presents the structure of the dissertation.

1.1 Multi-hop Wireless Networks

In its most general form, the concept of a Multi-hop Wireless Network (MWN) refers to a collection of autonomous devices (called *nodes*) capable of dynamically forming a wireless network without relying on centralized control or preexisting infrastructure. The nodes can be deployed over a large area, and communicate wirelessly using radio interfaces. Nodes are therefore connected by wireless *links*. A node can only communicate directly (establishes links) with nodes within its transmission range, called its *neighbors*. To communicate with non-neighbor nodes, packets must be relayed using intermediate nodes. In this case, packets traverse multiple links (*hops*) to reach the

destination; this is called multi-hop communication. Every node can therefore act as a router, forwarding packets for other nodes. A routing algorithm is needed to dynamically determine routes based on the current network connectivity (and possibly other factors such as load and channel assignment). The nature of the network permits nodes to move randomly. The movement of nodes, along with new nodes entering the network or nodes switching off leads to dynamic topologies.

One of the main interests in using MWNs is that they permit deploying a network easily and quickly over a relatively large area. Another desirable property common to MWNs is fault-tolerance. Their decentralized nature implies that no node is essential, and if a node fails or switches off data can be routed through alternate paths.

Depending on the particular characteristics and capabilities of the nodes, the technologies employed, and the application of the network, many types of MWN are possible. The three main types are:

- Mobile ad hoc networks (MANET) where nodes are expected to be mobile (free to move in any direction) and battery powered. A result of node movement is that the topology can change frequently. Application scenarios for MANETs include emergency situations like natural or human-induced disasters or military conflicts. These networks also include Vehicular ad-hoc networks (VANET). One of the main issues in these networks is coping with mobility.

- Wireless sensor networks (WSN), composed of sensor devices, typically small in size and with limited processing power and energy. Sensors are distributed in a region to monitor physical or environmental conditions, and they cooperatively forward the data to a main location. In these networks, one of the main problems is minimizing energy consumption.

- Wireless mesh networks (WMN), usually formed by stationary mesh routers, mobile clients, and a set of gateway nodes providing access to external networks. The mesh routers form a high-speed wireless backbone across a wide coverage area. One of their main applications is providing broadband Internet access. The main focus of these networks is usually to maximize capacity.

Wireless networks can be built using omnidirectional antennas, directional antennas, or a combination of both. An omnidirectional antenna transmits and receives radio

signals equally in all directions, forming links with other nodes in every direction. A directional antenna transmits and receives radio signals in a single direction, only forming links with nodes in that direction.

Directional antennas can be used to build point-to-point links, and as such decrease or even eliminate interference between links. However, most often, omnidirectional antennas are used in MWNs, due to the high complexity and cost of using directional antennas. Their use requires specialized hardware, and introduces many challenges at the MAC level (more hidden nodes, deafness problem) due to the fact that two nodes only "hear" each other if their antennas are aimed in the direction of the other node. This work assumes that nodes are equipped with omnidirectional antennas.

In MWNs nodes share the wireless channel, and as a result one of their major issues is limited capacity, due to interference between links. Wireless interference can greatly decrease the performance of the network and can be specially devastating to the performance of multi-hop flows. One of the main challenges in these networks is effectively utilizing their capacity, which is the principal object of study of this work.

1.1.1 Wireless Mesh Networks

Wireless Mesh Networks are multi-hop wireless networks often composed of mesh clients, mesh routers and gateways. Mesh routers are stationary nodes, interconnected by wireless links, forming a wireless backbone to relay the traffic of mesh clients. Mesh clients are end-user devices (e.g. laptops, smartphones) that associate with a router (e.g. their nearest router) to access the network and can be mobile. The gateways are a subset of routers that have direct connectivity to a fixed infrastructure (e.g. a wired network such as the Internet), providing access to other networks. WMNs are attractive because they provide an easy and cost-effective way of deploying a wide-area network and offer services such as broadband Internet access. In addition, the organization of multiple nodes into a mesh topology provides redundancy and multiple alternative routes inside the network. This, along with the self-configuring nature of the network provides robustness against link and node failures, and congestion. They can be deployed at the scale of home, enterprise, neighborhood or even metropolitan networks. Fig. 1.1 shows an example of a typical WMN. The gateway access scenario in WMNs is a main area of focus of this thesis.

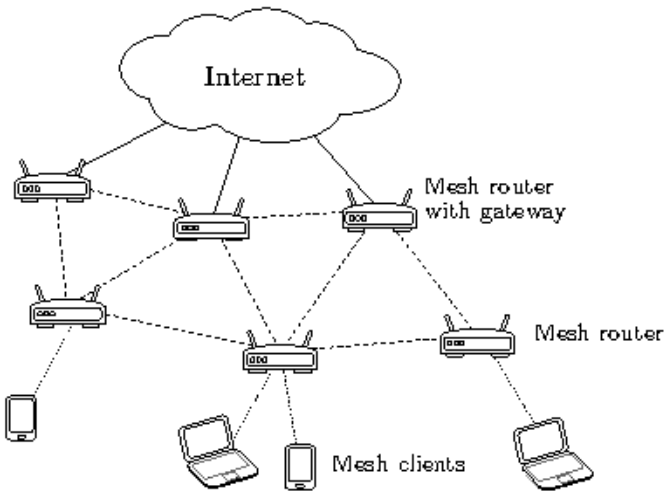


Figure 1.1: Wireless Mesh Network - The wireless mesh backbone formed by routers provides Internet access to clients. A subset of routers are gateways connected to the Internet.

In contrast to other multi-hop wireless networks, in WMNs routers are static and are not resource-limited. The fact that they are stationary and are intended to provide a ubiquitous high-speed backbone means that they can be connected to a permanent power source, are not limited in size or processing power, and can be equipped with multiple radio interfaces and/or antennas.

In MWNs links can transmit in the same frequency channel (*single-channel* networks) or use several non-overlapping channels (*multi-channel* networks). For example, IEEE 802.11b/g specifies 3 orthogonal channels while IEEE 802.11a specifies 12. The use of multiple channels reduces interference, permitting more concurrent transmissions and significantly increasing capacity. Equipping routers with multiple radio interfaces permits taking advantage of multiple channels. While single-radio multi-channel solutions have been proposed, they require nodes to dynamically switch between channels, while coordinating with neighboring nodes to ensure communication over a common channel for some period [12, 111]. However, such coordination is usually based on tight time synchronization among nodes, which is difficult to realize in a MWN, and/or very fast channel switching capability that is not yet available with commodity hardware. Additionally, a single interface does not permit a node to send and receive simultaneously. With multiple interfaces tuned to different channels it is possible for a node to

send and receive simultaneously, and radios can be tuned to a channel for a relatively long period of time. The fact that routers in WMNs are not energy-constrained, together with ongoing reduction in hardware costs turns multi-radio WMNs into a viable option. In this context, a key issue is the assignment of channels to radio interfaces to minimize interference.

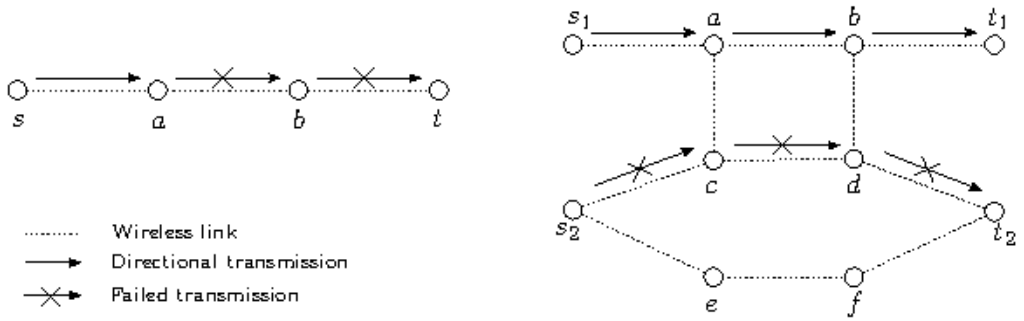
1.2 Problem and motivation

In MWNs nodes share access to the wireless medium. If multiple nodes close to a receiver R transmit simultaneously (whether to R or to other nodes), collisions at R can prevent successful decoding of a transmission directed to it. A wireless link is given by a transmitter-receiver pair. A link will be said to interfere with another if a transmission by the first can disrupt the transmission of the second. In this context, we will refer to a *collision domain* as a set of links that cannot be active simultaneously (the transmission of one link can collide with the transmission of the others). Simultaneous transmission by links in one collision domain is not possible, which severely limits the capacity of MWNs.

Spatial frequency reuse permits two links separated by a distance d in space to simultaneously transmit on the same frequency channel without interference. Several factors influence this distance and the degree of spatial reuse achievable, such as the transmission power of nodes, or MAC protocol characteristics (physical carrier sensing, virtual carrier sensing). The geographical distribution of nodes over the coverage area of the network divides the network in multiple different collision domains. In multi-channel networks, links can transmit using one of several non-overlapping frequency channels. Links only interfere with others using the same channel, dividing the network into smaller collision domains. This means that more simultaneous transmissions are possible, notably increasing capacity.

Wireless interference in MWNs has a profound effect on the performance of multi-hop transmissions. We can distinguish two fundamental types of interference affecting multi-hop flows (illustrated in Fig. 1.2):

- Intra-flow interference, which occurs when links in a path used by a traffic flow interfere with each other. This is a case of self-interference where the traffic along a path interferes with itself. In Fig. 1.2 (a), if all links transmit in the



(a) Intra-flow interference

(b) Inter-flow interference

Figure 1.2: Intra-flow and inter-flow interference - For simplicity, in this example carrier sense and interference range are equal to the transmission range, i.e. a node will not transmit if it hears the transmission of a neighbor, and a reception can fail if a neighbor of the receiver transmits. Figure (a) shows that no link in flow path $s \rightarrow t$ can transmit simultaneously. Figure (b) shows the links in path $s_2 \rightarrow t_2$ affected by transmissions of links of path $s_1 \rightarrow t_1$.

same channel, no two links in the path can be active simultaneously. If node s is transmitting, a cannot transmit because it is receiving. Node b cannot transmit because its transmission would collide with the transmission of s. The maximum throughput that can be achieved in this path is $\frac{C}{3}$, where C is the maximum link capacity. Throughput degradation in a chain topology [70] is known to follow a function $O(\frac{1}{n})$ where n is the number of hops, when $n < 5$.

- Inter-flow interference refers to interference between flows using different paths. Fig. 1.2 (b) shows links of path $s_2 \rightarrow t_2$ affected by transmissions on path $s_1 \rightarrow t_1$. Similarly, transmissions on path $s_2 \rightarrow t_2$ will also affect path $s_1 \rightarrow t_1$. As a result, both paths have to share the available capacity.

In a single-channel network, intra-flow interference cannot be avoided because it is not possible to schedule consecutive links of a path simultaneously. Inter-flow interference, on the other hand, can be mitigated if it is possible to use paths spatially separated with no mutual interference (in Fig. 1.2 (b) note that there is no inter-flow interference between paths $s_1 \rightarrow a \rightarrow b \rightarrow t_1$ and $s_2 \rightarrow e \rightarrow f \rightarrow t_2$). In multi-channel networks, both types of interference can be eliminated. Intra-flow interference can be avoided if consecutive links within a path transmit on different channels. Similarly,

inter-flow interference is removed if links of different flow paths which would otherwise interfere transmit on different channels.

To effectively make use of the capacity of the network and improve the performance of traffic flows, it is necessary to balance load between collision domains to improve resource utilization and maximize throughput [52]. The goal of load-balancing is to reduce the number of transmissions in a collision domain. This permits sharing the capacity of the domain among fewer transmissions, increasing throughput. The problem is notably different in single-channel and multi-channel networks. In single-channel networks, it essentially requires balancing load between non-interfering regions and paths, which is a routing problem. In multi-channel networks, there are two mechanisms to balance load: routing and channel assignment. Routing seeks paths with low intra-flow and inter-flow interference to distribute the load, because their use leads to more lightly loaded domains. Channel assignment, on the other hand, involves allocating channels to radio interfaces, which determines the channel used by a link to transmit. Because the set of radios and channels is finite, interference cannot be completely avoided and a channel assignment decision must carefully choose which channels to assign to which radios. Channel assignment does not simply increase the capacity of the network uniformly. Instead, by altering the set of collision domains in the network, it can grant different capacity to different links. There is a mutual dependency between routing and channel assignment. Routing seeks paths that provide more capacity (low intra-flow and inter-flow interference), and therefore depends on channel assignment. Channel assignment, on the other hand, needs to assign more capacity to regions with more traffic, and this is determined by routing. To solve optimally, both problems have to be considered jointly.

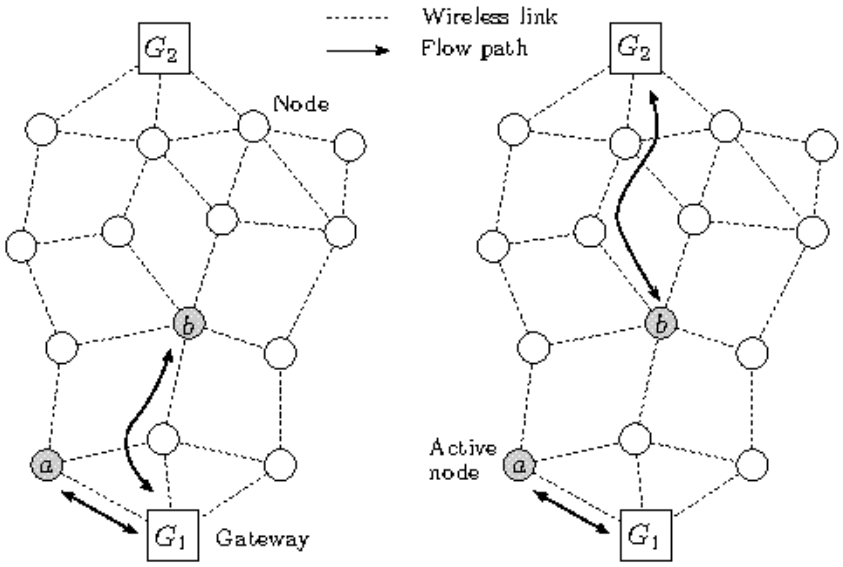
To avoid inter-flow interference, it is necessary to select paths that do not interfere. However, measuring interference is not trivial. In many cases, it is not possible for a node to identify the source of interfering signals which prevent it from accessing the medium or from successfully decoding a transmission, because the interfering source is outside its transmission range. Elaborate schemes to measure interference between pairs of links in a network have been proposed [90, 91, 104]. However, they are costly in terms of the time required to execute them and the control overhead required, and cannot be employed under dynamic conditions (e.g. node mobility). An easier alternative to estimate interference in single-channel networks is to measure the distance between

paths. Transmissions by nodes outside of interference range will not interfere. An obvious way to measure distance between nodes is based on their positional information (e.g. calculating the Euclidean distance between two nodes' geographic coordinates). Examples of previous work that find spatially disjoint paths based on positional information include EFR [89], 3DMRP [110] and the work in [72, 117]. The drawback of location based protocols is that obtaining this information generally requires specialized hardware and may not be readily available at all nodes. For this reason, an important problem addressed in this work consists in measuring spatial separation of a path to a set of nodes, based only on hop distance between nodes. Although metrics based on hop distance have been proposed to avoid interference, such as *coupling* [93] and *correlation* [119], in practice they are not designed to measure distance between any two nodes. These metrics essentially measure based on the number of direct links between nodes of different paths, which severely limits the degree of separation they can measure. It is important to note that spatially separated paths can provide other important benefits in MWNs besides interference avoidance. Examples include increased reliability when using multipath routing by gaining fault tolerance to situations of regional failures, and increased security by distributing a transmission over spatially disjoint paths to avoid interception. To take advantage of the above benefits, this thesis will develop a multipath routing protocol for MWNs capable of finding spatially disjoint paths between two nodes without location information.

In WMNs that provide Internet access, most traffic is directed to/from the gateways. Traffic aggregation in the paths leading to the gateways means that the majority of load is concentrated in the regions surrounding the gateways.

In a single-channel network there is little to no benefit in attempting to balance the traffic served by a gateway inside the network, because all of its flows ultimately contend for access in the region surrounding the gateway. This region constitutes an unavoidable bottleneck for flows served by the gateway. Consider the example in Fig. 1.3 (a), where the traffic of nodes a and b is served by gateway G_1. In this situation, it is not possible to find paths from the nodes to G_1 with low inter-flow interference. However, there is a way to balance the load in the network, and consists in consciously choosing the gateway through which the traffic of a node enters and exits the network (as shown in Fig. 1.3 (b)). It is important to note that load-balancing through gateway selection cannot be done in an arbitrary fashion. Association of nodes

to gateways must be chosen carefully, as it is important to maintain traffic locality and avoid harmful interference. For example, in Fig. 1.3 (b) note how assigning node a to G_2 would result in an overly long path, suffering from high intra-flow interference, and generating inter-flow interference in most of the network. To avoid using harmful



(a) No gateway load-balancing (b) Gateway load-balancing

Figure 1.3: Load-balancing in gateway access scenario - In single-channel networks, load can only be balanced effectively by reassigning traffic between gateways. In multi-channel networks, channel assignment can be used to avoid inter-flow interference between flows served by the same gateway.

paths, it is necessary to estimate the interference of the path in the network. As discussed previously, an effective way to do this is to measure the distance of the path to other paths in the network (note that it is not important to check interference of a path against flows served by the same gateway because inter-flow interference between them is unavoidable). Another important consideration that must be taken into account in this scenario is that most Internet traffic is based on TCP flows. Besides maximizing throughput by load-balancing, it is also important to grant flows fair access to resources. For this reason, traffic should be balanced at the TCP flow level (i.e. gateways should serve a similar number of flows) to allow flows to more fairly share the network capacity. Traffic conditions at the TCP flow level can vary rapidly. A gateway load-balancing solution must be able to promptly adapt to changing traffic conditions. Adapting to the

instantaneous load is desirable but can however lead to network instability, in the form of frequent route oscillations and non-convergence. Prior proposals exist to balance load between gateways in single-channel WMNs. However, they present the following important limitations: many do not take interference into account when balancing load [47, 87, 116, 120], do not implement mechanisms to avoid network instability [73, 87, 98] can have high convergence time [102], and to our knowledge no proposal balances traffic at the TCP flow level or balances elastic traffic.

When extending the above gateway access scenario to multi-channel networks, there is an important difference due to the fact that interference can be dynamically controlled by channel assignment. It is now possible to avoid intra-flow interference as well as inter-flow interference between flows served by the same gateway. In Fig. 1.3 (a), note that if the paths from G_1 to nodes a and b use different channels they will not interfere. To balance load in the network, it is necessary to jointly optimize routing and channel assignment given the current set of TCP flows. As a result, the aggregate throughput, average flow throughput and flow fairness will improve. In addition, such a solution must be able to promptly adapt to changing traffic conditions. Because channel assignment is traffic-aware, and rapid traffic fluctuations are expected at the TCP flow level, this imposes the need for frequent channel re-assignment. A change in channel allocation requires disseminating the new assignment to nodes in the network. The time required for this, along with the interface switching delay, can temporarily disrupt network activity. To effectively support this, the channel assignment algorithm must be able to quickly calculate a solution while minimizing the number of channel re-adjustments needed. Many works have studied the problem of joint routing and channel assignment in WMNs. In many cases, proposals consist of centralized approaches that assume mostly static network and traffic conditions, and as such cannot be considered practical, because they are not designed to adapt to the instantaneous network load. Another limitation of previous centralized approaches is that they make unrealistic and simplifying assumptions regarding the network and interference model, resorting to some combination of the following: time-slotted synchronized medium access model [8, 62], unrealistic binary interference model or throughput estimation [81, 103, 114]. More practical routing and channel assignment solutions have been proposed. However, some do not guarantee convergence [102] and many are not traffic-aware [25, 64, 101]. Furthermore, to our knowledge no prior work has addressed the problem of jointly

balancing traffic at the TCP flow level in a WMN and optimizing channel assignment for a specific set of TCP flows.

This work studies the above problems in detail and proposes practical solutions. The goals set out for the thesis with respect to each problem are described in detail in the next section, while the resulting contributions are explained in a following section.

1.3 Aims of this work

The main goal of this thesis is to present practical methods of improving the performance of traffic flows in dynamic MWNs (where the traffic pattern is not known a priori and can vary at any instant). One of the main ways to achieve this will be through the use of load-balancing. Load-balancing permits distributing the network load across multiple collision domains to reduce network resource utilization, increase throughput and avoid congestion. In addition to maximizing throughput, it is important to grant flows fair access to resources. To achieve this goal, this work sets out to study and propose solutions for the following problems:

1. How to measure spatial separation of paths to avoid inter-flow interference in single-channel MWNs. This problem requires the design of a metric to measure the spatial distance of a path to a set of nodes. To obtain distance information between nodes, one obvious way is to use location information (e.g. geographical coordinates). However, it is desirable to determine if hop-count distance, which is more easily obtainable, can correlate with Euclidean distance. Once such a metric is obtained, the design, implementation and evaluation of a multi-path routing protocol that discovers spatially disjoint paths will serve as proof of concept.

2. On-line gateway load-balancing in single-channel WMNs. Focusing on the gateway access scenario in WMNs, this problem centers on balancing traffic at the TCP flow level between gateways while avoiding paths with high inter-flow interference inside the network (using the previously developed path separation metric). Traffic at the TCP flow level can vary frequently and unpredictably, and so a responsive on-line protocol is needed for this purpose, capable of adapting to the current set of flows, without introducing high overhead in the WMN, and avoiding network instability.

3. On-line load-balancing routing and channel assignment in multi-radio multi-channel WMNs. This is a joint routing and channel assignment problem. The objective is to find routes in the WMN for every TCP flow served by a gateway, and allocate channels to links, to reduce network utilization and improve the throughput and fairness of flows. Similar to the previous problem, a responsive on-line protocol is needed. Routes and channel assignment can change frequently as a result of traffic variations, and so the channel assignment problem requires calculating a channel allocation with minimum number of channel re-adjustments with respect to the previous channel assignment. In addition, the solution must introduce minimal overhead in the network and an efficient protocol is needed to disseminate channel assignment to nodes.

1.4 Methodology

The general methodology used to study and solve each of the problems in this thesis is the following.

Using a mathematical model of the system, the problem is studied and an optimization problem is formulated (e.g. find the pair of paths with most separation connecting two vertices in a graph, find the flow-gateway association that minimizes maximum gateway utilization and path cost in the graph). The optimization problems considered in this work prove to be $\mathcal{NP}$-hard, meaning that it is unlikely that an optimal solution is obtainable in polynomial time. The algorithms developed are heuristic and provide an approximate solution. The main design goal of optimization algorithms in this work is low time complexity. This is necessary to design and support responsive on-line protocols that can promptly adapt to frequently changing network conditions (e.g. dynamic topology and traffic patterns).

Network protocols are developed to permit nodes to collect and distribute the necessary network state variables (e.g. current topology graph) to calculate a solution using the proposed optimization algorithms. The protocols are also responsible for communicating and applying the solution in the network.

The performance of the protocols and algorithms is studied using a combination of the following methods. In some cases, it is studied analytically. In others, evaluations are carried out by generating random problem instances under the mathematical model

used to study the problem, and evaluating the solution obtained. To generate "realistic" problem instances, the Yans physical model of the *ns-3* simulator is used. Based on the chosen positions of nodes in space and radio physical parameters (e.g. node transmission power, modulation used, path loss model), it determines graph connectivity (set of links) and the degree of interference between links. In all cases, the performance has been evaluated using the *ns-2* and *ns-3* network simulators, and compared against other relevant existing proposals.

1.5 Contributions

The main contributions of this work are:

- **PSD**: Path Spatial Distance metric. Metrics are proposed to measure the spatial distance of a node to a set of nodes, and the distance of a path to a set of nodes. The latter is called the Path Spatial Distance (PSD) metric. Depending on the problem, a set of nodes can refer to a path or a group of nodes in a region of the network. The primary goal of the PSD metric is to permit a routing algorithm to select paths that avoid or mitigate inter-flow interference in single-channel networks. PSD is easy to calculate and is based on the minimum hop distance between nodes, which is shown to correlate strongly with Euclidean distance in most practical scenarios. Therefore, it is not necessary to obtain location information (e.g. geographical coordinates) of nodes. The PSD metric is studied and presented in Chapter 3 (section 3.3).

- **SD-PCA**: Spatially disjoint path calculation algorithm. A heuristic algorithm is developed to find a pair of maximally separated paths connecting two nodes (s, t) in a graph, using the PSD metric. Its main advantage is that it can efficiently obtain a reduced set of paths from which a pair of paths with most spatial separation can be selected. This algorithm can be used by a node in the network with knowledge of the topology graph to find spatially disjoint paths to a destination node. The algorithm is implemented and tested in a multi-path routing protocol (see below), and evaluations show that it is effective at finding paths with maximal separation. SD-PCA is developed in Chapter 3 (section 3.4).

- **SDMR**: Spatially Disjoint Multipath Routing. This is a reactive multipath routing protocol that permits nodes to discover the topology graph on-demand (i.e. only when routes are needed) and, using the SD-PCA algorithm, calculate a pair of spatially separated paths to a destination. Unlike common on-demand routing protocols for MWNs, one discovery phase can serve to discover paths to multiple destinations. Evaluations show that the protocol requires less control overhead than a standard proactive protocol (OLSR), and can effectively discover the current topology, even in situations of node mobility. Simulations show that the PSD metric and SD-PCA algorithm are effective in separating paths. The SDMR protocol is described in Chapter 3 (section 3.5).

- **GWLB**: Gateway load-balancing protocol. The protocol balances load between gateways in a single-channel WMN, while at the same time avoiding the use of paths inside the network that produce harmful interference. The gateway selection algorithm is heuristic and assigns every TCP Internet flow currently active in the network to a gateway (i.e. traffic of the flow will enter and exit the network through the chosen gateway). This is calculated in a centralized manner by a controller node (typically one of the gateways), with the objective of minimizing the maximum number of flows served by the gateways and the cost of flow paths inside the WMN. This cost is based on the estimated inter-flow interference of the path (using the PSD distance metric).

  The gateway load-balancing protocol obtains all necessary information from the WMN routing protocol, and the current set of flows is known at all times without introducing any overhead. This, coupled with the efficiency of the gateway selection algorithm, permits a solution to be recalculated periodically with a very high frequency, ensuring high protocol responsiveness. The solution does not suffer from convergence issues because it is calculated centrally. Traffic is balanced at the TCP flow level, thus improving the throughput and fairness of flows. GWLB is studied, developed and evaluated in Chapter 4.

- **LBCA**: Load-Balancing and Channel Assignment. This protocol performs routing and channel assignment to optimize the performance of a set of TCP Internet flows (those currently served by a gateway) inside a WMN. A route is calculated

for every flow and a channel assignment (allocation of channels to links) in order to optimize network utilization and improve flow throughput and fairness.

The joint routing and channel assignment problem is decomposed into a separate routing problem and a separate channel re-assignment problem, solved in sequence only once. The routing problem assumes an interference-free network and calculates flow routes to minimize link utilization. The length of paths is taken into account to avoid using overly long paths. Independent paths can be assigned to flows on an individual basis, meaning that flows of the same mesh router need not follow the same path (permitting increased load-balancing). The channel re-assignment problem, given the link-loads (determined by routing) and previous channel assignment, calculates an edge-channel assignment to minimize interference as well as the number of channel re-adjustments.

Limiting the number of channel re-adjustments is crucial to ensure high responsiveness of the protocol. In addition, the proposed routing and channel assignment algorithms are of low time complexity, permitting frequent recalculation of the solution. The solution is calculated in a centralized manner at the gateway. The gateway obtains knowledge of current Internet flows as traffic passes through it, without introducing any overhead in the WMN. Routing is applied instantaneously by adding the route to a flow's packets (source routing). To disseminate channel assignment information in the network, a channel assignment distribution protocol is developed. This protocol permits disseminating channel information quickly and with low overhead, and does not require assigning one interface of every node to a common channel.

LBCA is studied, developed and evaluated in Chapter 5.

1.6 Organization

The rest of this dissertation is structured as follows:

Chapter 2 provides background information on the most relevant topics in MWNs concerning the problems addressed in this work, including capacity issues of MWNs, routing and channel assignment.

Chapter 3 consists of three main parts. The first part explains and develops the PSD metric to measure path spatial separation. The second presents the design of SD-PCA. The third part presents the design, implementation and evaluation of the SDMR multi-path routing protocol.

Chapter 4 presents the study, design and evaluation of the Gateway Load-balancing protocol for single-channel WMNs.

Chapter 5 presents the design, implementation and evaluation of the joint routing and channel assignment protocol for multi-radio multi-channel WMNs.

Finally, Chapter 6 presents the conclusion of this work, lists the research publications made during its elaboration and discusses future work.

Chapter 2

Background

This chapter presents background on the most relevant topics in MWNs related to the subject of this work. After briefly presenting the history of MWNs, it discusses prominent issues of wireless transmission and interference in multi-hop network, including the topic of modeling interference. This leads to the discussion of capacity in MWNs, where it explains the asymptotic bounds on achievable throughput and its causes. Capacity scaling due to the use of multiple channels is also analyzed.

The chapter provides a description of some of the most relevant routing protocols and metrics proposed for MWNs, discussing their advantages and disadvantages. This includes, among others, single-path and multi-path routing protocols, and routing metrics suitable for multi-channel networks. It also discusses the important problem of network instability faced by load-sensitive routing and other load-balancing strategies.

Finally, the chapter explains the basics of channel assignment and routing in multi-radio multi-channel MWNs, and describes some of the most relevant protocols and algorithms proposed to this date.

2.1 A bit of history

The idea of ad hoc networking and multi-hop wireless networks dates back to the 1970s, when the U.S. Defense Advanced Research Projects Agency (DARPA) researched Packet Radio networks [57]. In the context of the military, two prime requisites were rapid network deployment and survivability of the network. Traditional systems based on central servers or base stations were not a valid option because they required de-

ployment of infrastructure prior to the use of wireless devices. Reliance on existing infrastructure is not possible in enemy territory. It is also unacceptable that the destruction of a host-device or base station lead to the failure of the whole network. That is the reason why a distributed architecture, without critical points of failure and using wireless radio links presented the ideal solution.

The limited transmission range of radio interfaces lead to the design of nodes with the capability of forwarding information using other nearby nodes, leading to the development of wireless multi-hop communication. Multi-hop communication, combined with packet-switching technology and novel medium access protocols [6], formed the initial base for the development of large-scale fault-tolerant ad hoc networks or Mobile Ad hoc Networks (MANETs).

The DARPA agency played a major role in the initial research and development of these networks. The first packet-switched wireless network was the so-called Packet Radio Network, -PRnet- [57]. This network later evolved with the introduction of Survivable Radio Network (SURAN) in 1983 and subsequent developments in networks by the Global Mobile (GloMo) program in 1994 [68, 106]. Late related developments consist of the US Army Tactical Internet (TI) in 1997 and the "Extending the Littoral Battlespace Advanced Concept Technology Demonstration" (ELB ACTD) used by the U.S. Marines in 1999.

The huge growth experimented by the Internet at the start of the 90s, combined with decreasing cost of radio interfaces lead to a notable interest in ad hoc networks by the scientific community. In 1997, the Internet Engineering Task Force (IETF) established the MANET Working Group [2] to create and standardize routing protocols to support dynamic and multi-hop paths in MANETs.

Traditionally, MWNs have consisted of nodes equipped with a single omni-directional radio. In recent years, lower cost hardware, proliferation of 802.11 APs, as well as increasing interest in providing wide-area broadband access using wireless technology, has propelled the development of mesh networks where routers can be equipped with multiple radios, antennas, and have ample processing power. There are numerous standardization efforts underway related to wireless mesh technology, including IEEE 802.11s, 802.15.5 and 802.16j.

2.2 Wireless transmission in MWNs

This section discusses the most relevant issues regarding wireless communication and interference in MWNs that affect the capacity of these networks. It also studies the most extended methods used to model interference between wireless links, which constitutes the basis for interference-aware routing and channel assignment protocols and algorithms.

2.2.1 Physical-layer transmission

Digital signals in MWNs are most commonly transmitted using radio waves. Signal strength attenuates as the radio wave propagates through space due to path loss. Path loss is caused by a number of factors, including the distance between transmitter and receiver, environment, propagation medium and obstacles. The loss exponent is typically used as a means of modeling path loss. Path loss can be calculated using the formula:

$$L = 10\alpha \log_{10}(d) + C \tag{2.1}$$

where L is the path loss in dB, α is the loss exponent, d is the distance between transmitter and receiver, and C is a constant that accounts for system losses. The value of α is normally in the range of 2 to 4 (where 2 represents propagation in free space, and 4 is for relatively lossy environments). In some environments, such as buildings and other indoor environments, the path loss exponent can reach values in the range of 4 to 6.

For a radio transmission, the Signal-to-Interference-and-Noise-Ratio (SINR) is the ratio of the signal's power at the receiver to the combined noise and interference power in the receiver. Interference includes all signals different from the signal to be decoded. More specifically, for a radio transmission from node a to b, the SINR at node b is:

$$SNIR_{ab} = \frac{S_{ab}}{N_b + NI_b} \tag{2.2}$$

where S_{ab} is the average power at node b of the signal sent by a, N_b is the ambient noise power level at b and NI_b is the sum of interfering signals at b.

To decode a transmission with an acceptable bit error probability, the SINR at the receiver must exceed a threshold β. In other words, the Bit Error Rate (BER) during

decoding is a function of the SINR [45]. The lower the SINR, the more difficult it is for the modulation scheme to decode the received signal. The value of β depends on a number of factors, one of the main ones being the modulation scheme used. From equation 2.2, we can conclude that successful reception is heavily influenced by cumulative interference from other transmitters. A number of factors influence interference at a receiver. The main ones include transmit power of interfering transmitters and distance to transmitters.

2.2.2 Medium access

In a MWN nodes share the wireless medium. Nodes in close vicinity of each other cannot transmit simultaneously. More specifically, a node should refrain from transmitting if it will interfere with and disrupt an ongoing transmission from another node. A Medium Access Control (MAC) protocol is necessary to coordinate access of nodes to the shared medium, and allow nodes to share the channel capacity.

In MWNs many practical difficulties prevent the use of collision-free channel access schemes like Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA). These schemes reserve resources and partition the channel so that devices can transmit without collision. Traditionally, they require centralized control, which is difficult to achieve for medium access in a MWN. On the other hand, implementing distributed approaches of these schemes is challenging. Other difficulties include the fact that TDMA requires time synchronization among nodes, which is difficult to achieve in a MWN [48], and CDMA requires code management.

Contention-based protocols assume no central entity to allocate channel resources in the network, and as a result are more suitable for implementation in MWNs. To transmit, each node must contend for the medium. Collisions result when more than one node tries to transmit at the same time. In the context of MWNs, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the most widely used medium access scheme, and is implemented in the Distributed Coordination Function (DCF) of 802.11. However, CSMA/CA results in poor spectrum usage due to very low frequency spatial reuse efficiency [7].

In CSMA/CA nodes employ carrier sensing whereby a node senses the channel before transmitting and if it is busy it backs off (exponential back-off) and remains idle for

a random period of time before retrying. The timer-based exponential back-off mechanism provides contention resolution (it prevents neighboring nodes from transmitting simultaneously). Optionally, the DCF implements virtual carrier sensing by means of an exchange of Request to Send (RTS) / Clear to Send (CTS) messages. The purpose of this is to more effectively handle the hidden node problem.

Although they permit distributed medium access, exponential back-off, carrier sensing and virtual carrier sensing also introduce many inefficiencies. Carrier sensing and virtual carrier sensing are conservative and in many occasions can prevent otherwise harmless transmissions (exposed node problem). The exponential back-off mechanism provides contention resolution at the cost of reduced channel utilization. Forcing senders to back-off on a packet-by-packet basis in the presence of multiple access interference, has negative consequences on throughput and channel access delays.

2.2.3 Modeling interference

This subsection discusses the most relevant models and methods proposed in the literature to characterize interference between wireless links. These models are necessary for the design of interference-aware solutions, and can be characterized mainly based on their accuracy and complexity.

2.2.3.1 Protocol Model of interference

The Protocol Model [43] is frequently used as a way of simplifying the mathematical characterization of the physical layer. In this model, a transmission is successfully received only if the distance between the sender and receiver is less or equal to the transmission range T_r, and if no other node within interference range I_r of the receiver is transmitting at the same time.

If nodes use carrier sensing, an additional requirement is that no node in the carrier sense range CS_r of the sender transmit at the same time. Furthermore, to guarantee a correct transmission in 802.11 DCF, it is also necessary that *both* the transmitter and receiver have no active nodes in carrier sense range and interference range, due to the use of link-layer acknowledgments (i.e. the receiver has to send an acknowledgment packet (ACK) to the sender).

In this model, the transmission range T_r should be chosen based on the transmission power, path loss and the signal-to-noise ratio (SNR) threshold necessary to correctly

decode a transmission. Regarding the choice of interference range I_r, most works that use the protocol model make simplifying assumptions, and assume that $I_r = \Delta T_r$, where $\Delta \geq 1$ is a constant (e.g. a typical value is $\Delta = 2$). In practice, however, the interference range depends on the distance between sender and receiver (which determines the power of the signal), and the distance between the receiver and each interfering node (which determines the power of the interfering signals).

This model only considers the effect of a single interfering node, and does not consider the effect of cumulative interference from several simultaneous non-intended transmitters. Similarly, in real-world operation, when a node does carrier sensing, it detects the combined energy of all transmitters in its vicinity, not just from one at a time. This is not taken into account by the model.

A main limitation of this model derived from the above is that it is a *binary* interference model. This means that, according to the model, two links either can or cannot transmit simultaneously. If a node is in the interference range of a non-intended transmitter, the model assumes that the node cannot correctly receive from its intended transmitter. However, this is not true. In practice, correct reception depends on the bit error probability based on the SINR. On the other hand, if a node is outside the interference range of a non-intended transmitter, the model assumes that there is no interference. This, again, is not necessarily true as interference from different transmitters can aggregate and affect reception.

Many optimization approaches proposed for MWNs rely on this model. Its main advantages are simple mathematical characterization which permit it to easily be applied to analyze and optimize large-scale wireless networks.

2.2.3.2 Physical Model of interference

This model, also called the SINR model of interference, is based on the physical layer characteristics of transceivers. In this model, a transmission is successful only if the SINR at the receiver exceeds a threshold so that the signal can be decoded with acceptable bit error probability (see Equation 2.2). If using carrier sensing, a node will refrain from transmitting if the cumulative energy sensed in the medium is above its carrier sensing threshold CS_{th}.

One of the main drawbacks of this model is that it translates into more complex mathematical relationships than the Protocol Model, resulting in a high computa-

tional complexity to solve optimization problems in a MWN environment, specially if it involves cross-layer optimization. This is because SINR calculation is a non-convex function with respect to the transmission powers [109].

The physical model requires knowledge of the signal power S_{ij} at a receiver j when a node i transmits. This can be predicted based on the transmit power of i and a path loss model. However, such a model may not likely account for all wireless propagation and environmental phenomenon (specially the dynamic nature of the wireless medium) and this prediction may not translate to behavior of real-world networks. Additionally, obtaining the real value of S_{ij} to calculate SINR in a real-word scenario is not trivial. Current hardware cannot differentiate between the power of the intended signal, and the power of noise and interfering signals.

2.2.3.3 Measuring interference

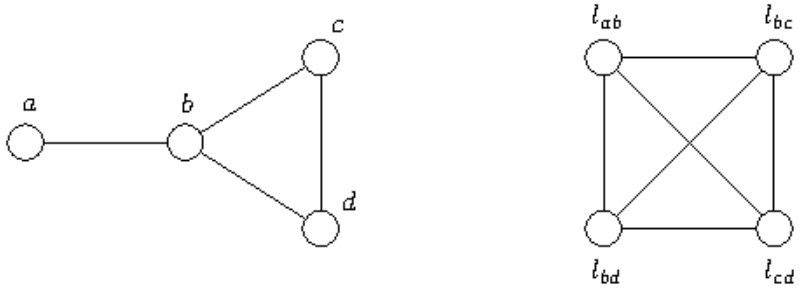
The above models can prove inaccurate in real-world scenarios [91]. There are other factors beside distance (e.g. environment and obstacles) that attenuate signal strength. It is very difficult to account for this in theoretical models. According to Reis et al. [104], "RF propagation in realistic environments is sufficiently complex that the only existing feasible method for estimating packet delivery between two nodes is to measure it".

An alternative to predicting interference from a theoretical model is to empirically measure the interference characteristics of the network. However, measuring interference is not trivial. In many cases, it is not possible for a node to identify the source of interfering signals which prevent it from accessing the medium or from successfully decoding a transmission, because the interfering source is outside its transmission range. Empirical methods have been proposed [90, 91, 104], with the goal of obtaining an estimate of the pair-wise interference of links. More specifically, measuring the packet delivery ratio (and consequently the Packet Error Rate -PER-) of a link when no other link transmits concurrently and measuring its delivery ratio when another transmits simultaneously. These methods require conducting measurements in the network which can be lengthy in time and may require interrupting network traffic. As such, they are only suitable for static networks with no node mobility, such as WMNs.

2.2.3.4 Conflict graph

A MWN can be modeled by a directed graph $G = (V, E)$ where V is the set of nodes in the network and E the set of edges, representing the wireless links. Let us call this the *connectivity graph*. For the construction of this graph it is assumed that every two nodes in the network have a channel in common.

The concept of a *conflict graph* was introduced by Jain et al. in [52] and has since been used in many works to model interference between wireless links. It is an undirected graph derived from the connectivity graph, and is defined as $G_c = (V_c, E_c)$ where $V_c = E$ (i.e. each edge of the connectivity graph is a node of the conflict graph). Let l_{ab} denote an edge between two nodes a and b of the connectivity graph. An edge exists between two nodes l_{ab} and l_{cd} of the conflict graph if the two wireless links cannot transmit simultaneously. This information can be derived from the Protocol Model of interference. Figure 2.1 shows a simple example of a conflict graph.



(a) Connectivity graph (b) Conflict graph

Figure 2.1: Connectivity graph and conflict graph - In this example it is assumed that no two edges of the connectivity graph can transmit simultaneously, and therefore there is a link between every node of the conflict graph.

The above conflict graph, although frequently used, only models binary interference. It is possible to model fractional interference using a weighted conflict graph. In this case, E_c is a set of directed edges. Let $u \xrightarrow{w} v$ denote an edge of G_c between nodes $u, v \in V_c$ with weight w. The weight w indicates the degree of interference that edge u produces in edge v. This could indicate, for example, the PER of v when u transmits concurrently, and as such, can be predicted using measurement techniques or the Physical Model.

Note that, in any case, the above conflict graphs do not model cumulative interference (i.e. the aggregate effect of simultaneous transmissions of several interfering links).

2.3 Capacity of MWNs

Previous studies have shown that the per-node share of aggregate throughput in networks with random source-destination nodes is bounded asymptotically by $O(\frac{1}{n^{\alpha}})$, where n is the number of nodes and α depends on topology and traffic characteristics [43, 63]. Depending on the value of α, as the network size increases, the per-node throughput can decrease drastically. The analytical study by Gupta et al. [43] suggests that $\alpha = 0.5$ for random node location and random source-destination pairs. Experimental studies [63] are far more pessimistic and show that $\alpha = 1.68$. This behavior also holds true in multi-channel networks. Although C non-overlapping channels can provide C times more raw bandwidth, the upper limit of the capacity is unaffected by the raw bandwidth or the way the bandwidth is split among multiple interfaces. Note that this does not mean that capacity scaling with the use of multiple channels is negligible. In fact, it can scale linearly with the number of channels. We will develop this further in the next subsection. In the case of a chain topology in single-channel networks, the throughput is known to be $O(\frac{1}{n})$ [70].

From the above results, it is clear that interference and multi-hop communication are key issues affecting the capacity of MWNs. These results, however, are only true when two nodes communicate with equal probability. They do not hold if nodes communicate more frequently with nearby nodes. The above suggests that to maintain adequate performance it is necessary to limit network size (number of nodes) and/or to maintain traffic locality. If traffic patterns are random (random source-destination pairs) it is not possible to scale a network in size without considerably degrading throughput. On the other hand, if traffic is mostly local, it is possible to scale the network up to any desired size (specially when using multiple channels).

Although maintaining traffic locality is important to achieve high aggregate throughput, depending on the network and application, it can be difficult to achieve. There may be no way to avoid the case where two nodes which are far apart need to communicate. On the other hand, WMNs where most nodes access external networks (e.g.

the Internet) present a favorable situation. In this scenario, there is a unique traffic pattern where nodes send/receive most of their data to gateways. By deploying and assigning one gateway to serve the nodes in a particular area, it is possible to maintain traffic locality. As the network (and coverage area) increase in size, more gateways can be added. This particular architecture can be seen as a design similar to the one used in cellular networks. Figure 2.2 illustrates an example.

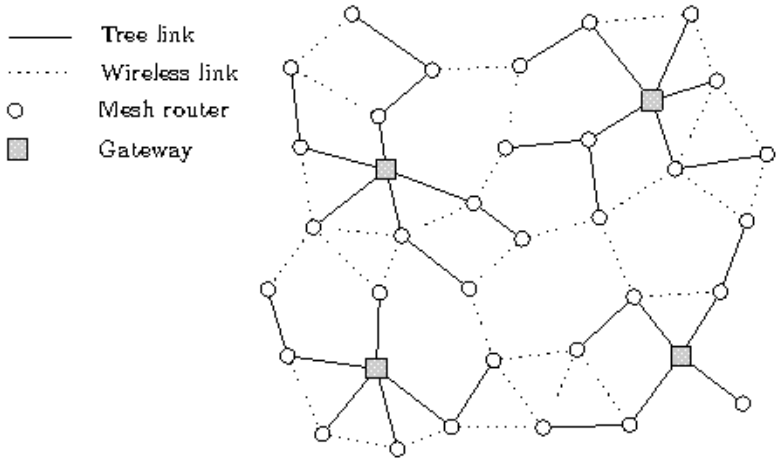


Figure 2.2: Internet access WMN backbone - In this example, mesh routers connect to their nearest gateway in terms of hop count, forming a tree structure. The maximum number of hops to reach a gateway is three, and in most cases is one or two.

Another important consideration is that, in practice, not all nodes in the network will be active simultaneously, which is the assumption of the capacity studies discussed above. In this situation, an increased number of nodes can provide more alternate routes in the network, which can prove particularly useful to balance load and avoid congested areas [52]. Taking this into account, it may prove beneficial to break traffic locality to some extent to achieve better load-balancing. As an example, in the case of WMNs with multiple gateways, it may not be necessary in some cases for every node to send traffic to its nearest router (as is the case in Fig. 2.2). If some gateway regions are specially overloaded while other gateways have many inactive nodes in their neighborhood, it can be beneficial if some nodes send their traffic to another (more distant) gateway. This particular problem is addressed in chapter 4.

This discussion on the capacity limits of MWNs points out the importance and necessity of mechanisms to increase and effectively use the capacity. There are many

possible solutions affecting most layers of the system, and an optimal solution will involve a cross-layer optimization. Solutions include load-balancing routing, use of multiple channels, directional antennas, power control and improved MAC protocols. This work focuses on the use of load-balancing and multi-channel networks. These topics will be discussed in the rest of the chapter.

Regarding the use of load-balancing in MWNs, best utilization can be achieved with dynamic load-balancing, adapting to the *current* traffic conditions. This, however, introduces important challenges in routing protocols (discussed later in section 2.4.6). Previous studies show that there is an opportunity of achieving throughput gains by load-balancing, but interference aware routing protocols are needed [22, 27, 52]. Section 2.4.5 contains a description of routing metrics designed on this principle.

2.3.1 Multi-channel networks

In theory, the capacity of the network can scale linearly with the number of channels C, because the network is decomposed into C parallel channels. In practice, limitations in the number of radios available at each node can make achieving this gain difficult.

2.3.1.1 Single-radio networks

Commodity single-radio wireless transceivers operate in a half-duplex mode, i.e. they cannot transmit and receive data simultaneously. If nodes are equipped with only one radio (single-radio networks) they cannot transmit and receive simultaneously, even if multiple channels are available. Additionally, because a radio can only be tuned to one channel at any time, parallel use of multiple channels at the same node is not possible. Using multiple channels in a single-radio network requires nodes to dynamically switch between channels, and to coordinate with neighbors to ensure communication over a common channel for some period [12, 111]. However, such coordination is usually based on tight time synchronization among nodes, which is difficult to realize in a MWN, and/or necessitates very fast channel switching capability that is not yet available with commodity hardware. The switching delay for current 802.11 hardware ranges from a few milliseconds to a few hundred microseconds [103]. This also leads to the fact that switching delay along a multi-hop route effectively lengthens the route (the delay manifests as virtual hops).

2.3.1.2 Multi-radio networks

Nodes equipped with multiple radio interfaces do not have the limitations of single-radio nodes and can simultaneously receive on one interface and transmit on another (using different channels). In addition, a node can make parallel use of multiple channels at the same time.

Multi-radio networks have the capability of achieving the theoretical gains of using multiple channels. If nodes are equipped with C radios, linear capacity scaling can be obtained simply by tuning each interface of a node to one of the available channels. In practice, nodes have a limited number of radios, and the assignment of channels to radio interfaces becomes a determining factor to achieve all possible gains. It is necessary to carefully assign channels to radio interfaces to minimize overall network interference. Figure 2.3 shows a simple example where the use of multiple radios and channels permits increasing the throughout by a factor of 3 and eliminate intra-flow interference. Note that the throughput achievable in single-channel networks in this example is $R/3$, where R is the maximum link rate.

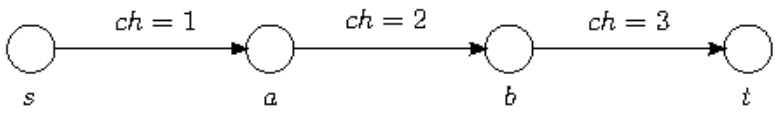


Figure 2.3: Multi-radio multi-channel example - If nodes are equipped with two radios, and each link transmits in a different channel, it is possible for the source s to transmit to t at the maximum link rate R.

It is important to note that although channel assignment increases network capacity, the routing protocol must be able to select paths that minimize intra-flow and inter-flow interference (i.e. routing must be interference and channel-aware). However, the routes used determine the load in every link and collision domain. This in turn influences channel assignment, i.e. channels should be allocated to assign more capacity to regions with more load. This leads to a mutual dependency between routing and channel assignment. In summary, the limited number of radios leads to the fact that to optimize network performance it is necessary to jointly optimize routing and channel assignment. Furthermore, if the traffic profile is known, additional performance can be achieved. The topic of channel assignment and routing will be discussed in section 2.5.

2.4 Routing in MWNs

2.4.1 A taxonomy of routing protocols

Routing protocols in MWNs can be classified based on three fundamental characteristics:

- Hop-by-hop vs source routing
- Reactive vs proactive
- Single-path vs multi-path

2.4.1.1 Hop-by-hop vs source routing

In hop-by-hop routing protocols, nodes only store in their routing table the address of the next device to reach a destination, called *next hop*. A source sends a packet to a destination by passing the packet to the next hop along the route. Successive nodes in the route repeat the same step and forward the packet to the next hop in the chain until the packet reaches the destination.

When a source routing protocol is used, the source node inserts the (complete or partial) route to reach the destination (called *source route*) in the header of every packet. Nodes are visited in the order specified by the source route, the last node being the destination. Using this technique, it is the source node who selects the route to be followed by its packets, while in hop-by-hop routing the route is determined by intermediate nodes. One important advantage of source routing is that it allows a source to directly manage network performance, e.g. by forcing packets to travel over one path to prevent congestion on another. This offers a great deal of flexibility, as a source can even classify its traffic into flows and send each flow through a different path. The main limitation of source routing is the need to include the route in the header of every data packet, which, depending on path length, may introduce substantial overhead. This overhead is directly proportional to the path length.

2.4.1.2 Proactive vs reactive

Proactive routing protocols, also called "table-driven" protocols, are designed so that nodes always try to maintain up-to-date routes in the network to all other reachable

nodes. The goal is for nodes to always have an up-to-date routing table specifying the route to every node in the network. In this way, a source node can immediately obtain a route to a destination when needed. Nodes adapt to changes in the topology by periodically propagating updates through the network. In this way, routes are proactively maintained regardless of whether there is data traffic or not. The overhead of maintaining an up-to-date network topology can be high, depending on the frequency of updates. The frequency of updates will mainly depend on the frequency with which the topology can change. This in turn depends mainly on the mobility of nodes, and as such, the frequency can be much higher in MANETs than in WMNs. Most proactive routing protocols proposed for MWNs have inherited properties from algorithms used in wired networks. To adapt to the dynamic features of MWNs, modifications have been made to traditional wired network routing protocols.

With reactive (also called "on-demand") routing protocols for MWNs, routes are only searched by a node when needed. This contrasts with proactive protocols which continuously maintain all routes inside the network up-to-date. As such, reactive protocols can substantially decrease control overhead, specially in highly dynamic networks. Thus, reactive protocols have better scalability than proactive routing protocols in dynamic networks (e.g. MANETs). A main limitation, however, is that nodes may not have routes immediately available when needed and, as a result, may suffer from long delays before they can start forwarding data.

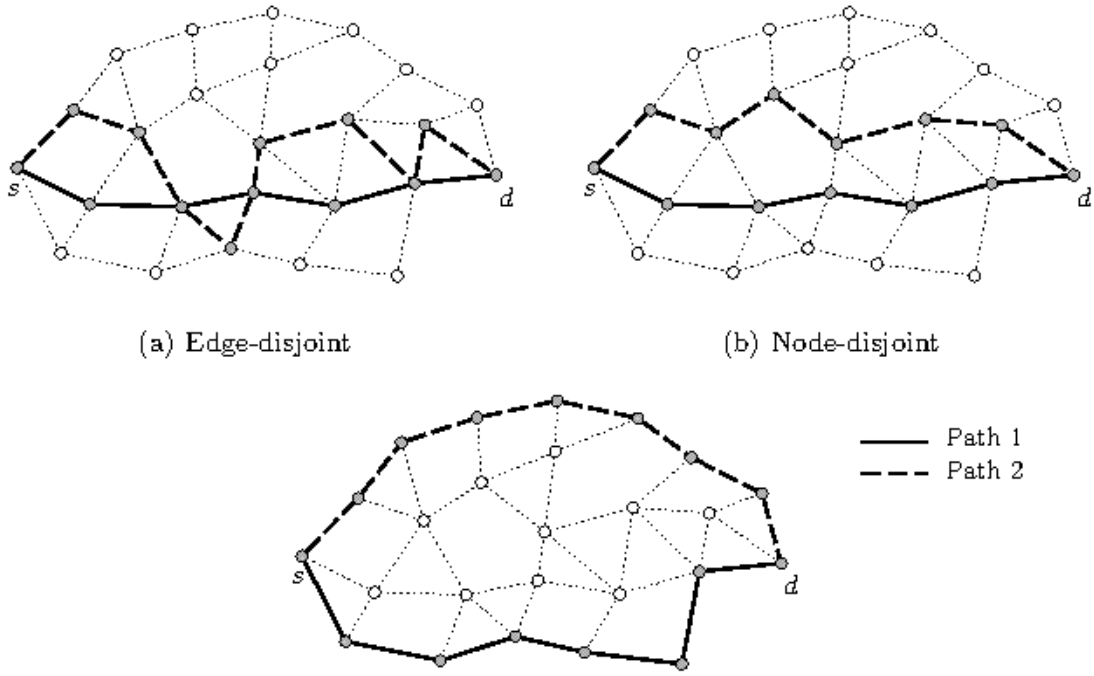
When using a reactive protocol, if a node needs a route to a destination and no path is known, it invokes a route discovery operation. This operation terminates when either a route is found or no route is found after a specified time. Because active routes can break in MWNs (e.g. due to node mobility), route maintenance is an important operation of reactive routing protocols, by which the nodes try to repair the route by finding alternate paths, or inform the source, which has to invoke a new route discovery.

2.4.1.3 Single-path vs multi-path

Single-path routing is the most prominent routing scheme. In this approach, a source node only has one route available to reach a destination.

Multi-path routing is the routing technique whereby a source node can use multiple alternative paths through a network to reach a destination node. This can provide

a variety of benefits in MWNs such as fault tolerance, increased bandwidth, load-balancing or improved security. The multiple paths found might be overlapped, edge-disjoint, node-disjoint or even spatially disjoint with respect to each other. Figure 2.4 illustrates this. Edge-disjoint paths are those with no links in common. Node-disjoint paths are those with no nodes in common (with the exception of the source and destination). Note that every node-disjoint path is an edge-disjoint path. Spatially-disjoint paths are those whose nodes are separated by a specified distance in space.



(a) Edge-disjoint

(b) Node-disjoint

(c) Spatially-disjoint

Figure 2.4: Taxonomy of path disjointness - This figure shows examples of different classes of path disjointness, where the paths connect two nodes s and d.

Multi-path routing can provide fault-tolerance in situations where a path currently in use fails. The source node can simply switch to an alternative path, without having to perform a new route discovery (specially useful in reactive routing protocols). In this context, edge-disjoint paths provide robustness to link failures, whereas node-disjoint provide additional robustness to link and node failures. Spatially disjoint paths can provide robustness to situations of regional failures (i.e. when all the nodes in a region of space fail).

Multi-path routing can also provide increased bandwidth, by offering a source node the aggregate capacity of multiple independent paths. In single-channel networks, however, it may not be possible to achieve this gain, due to interference between the paths. Even if the paths are spatially separated, if they originate and terminate at the same nodes, they will inevitably contend in the areas around the source and destination. For this reason, a mechanism to reduce interference is necessary to effectively utilize multi-path routing to increase throughput (e.g. multi-channel networks).

Related to the issue of increased bandwidth is the fact that multiple alternate paths can also prove helpful to achieve load-balancing, permitting nodes to distribute their load through various paths, to avoid overloading a path. Same as above, avoiding interference is necessary to effectively balance load.

An additional benefit of multi-path routing is increased security, specially important in wireless network communications. The idea is that, for a communication to be compromised, the multiple routes it traverses have to be compromised. Spatially-disjoint paths can prove particularly useful in this scenario, because it reduces the probability of a node intercepting the communication of every path.

2.4.2 Single-path proactive protocols

We briefly discuss some of the most relevant proactive routing protocols proposed for MWNs.

2.4.2.1 Destination-Sequenced Distance Vector (DSDV) routing

DSDV is a proactive routing scheme developed by C. Perkins [96] for MANETs, based on the Bellman-Ford algorithm [15]. It eliminates the problem of routing loops resulting from the use of the distributed Bellman-Ford algorithm and the count-to-infinity problem, by using sequence number on route updates.

The protocol disseminates routing tables by flooding. To reduce overhead, complete routing tables are exchanged infrequently, while small incremental updates are sent more frequently. The tables can be exchanged periodically or only when changes in the topology are detected. Each table entry contains the distance and next-hop to the destination, and a sequence number. When a router receives new information, it uses it if the sequence number is greater. If the sequence number is the same, it uses it only if the distance is less.

Nodes initially flood entries where they are the destination and the sequence number is even and monotonically increasing. When a node B detects that the route to a destination D is broken, it updates the entry by incrementing the sequence number (which is now odd) and setting the distance to infinity. It then floods the entry. When a node A receives the message, it incorporates the entry in its table, only if it does not have a better entry to reach D. This entry will be replaced when the node receives a positive update with a greater sequence number.

The main limitation of DSDV is the high control overhead it requires. However, it is suitable for small ad hoc networks. Since no formal specification of this algorithm is present there is no commercial implementation. DSDV, however, has influenced the design of the well-known AODV [94] protocol, which is another sequence-number-based distance vector protocol, with the main difference that AODV is reactive.

2.4.2.2 Wireless Routing Protocol (WRP)

WRP is a proactive unicast protocol for MANETs developed by Murthy and Garcia-Luna-Aceves [86]. It uses an enhanced version of the distance-vector routing protocol, which uses the Bellman-Ford algorithm [15] to calculate paths. To adapt to the dynamic nature of MANETs, the protocol introduces mechanisms to avoid route loops and to permit reliable exchange of route update messages.

The protocol, much like DSDV, inherits the properties of the distributed Bellman-Ford algorithm. It can achieve faster convergence than DSDV and requires fewer table updates (less control overhead), at the cost of more memory to maintain more tables. It is designed to be executed on the link-layer of a MWN. Because table update messages can be lost or corrupted due to the nature of wireless communication, WRP introduces the use of Acknowledgment Messages (ACK) to achieve reliable updates by retransmission.

Messages are broadcasted and so a single message can be received by all neighbors. However, every neighbor must send an ACK. Besides the use of ACK, every node maintains the state of links to neighbors using periodic HELLO messages when there is no traffic. When a certain number of HELLO messages are not received from a neighbor, the node assumes the link has broken.

Each node maintains four different tables. A distance table, a route table, link-cost and another to maintain retransmission and ACKs. The latter, called Message Retrans-

mission List Table (MRL), contains the sequence number of received update messages, a retransmission counter and a list of updates sent. As soon as a node detects a change in the topology (broken or new link) it sends the tuple $< destination, distance >$ to advertise the change. To avoid the count-to-infinity problem, each node must check the consistency of data received in every route update message.

2.4.2.3 Optimized Link State Routing (OLSR) protocol

The Optimized Link State Routing (OLSR) protocol [20, 51] is a proactive link-state routing protocol for MWNs. Nodes use HELLO messages to discovery their one-hop neighbors (link-state) and then disseminate link-state information through the network using Topology Control (TC) messages. Individual nodes, upon receiving TC messages from every other node in the network, can build and maintain an up-to-date connectivity graph of the topology, which is used to compute next hop destinations for all nodes in the network based on the calculation of shortest paths. The resulting paths depend on the link-cost metric used (e.g. hop-count or more advanced metrics such as those described in section 2.4.5).

OLSR represents an optimization of a link-state protocol for dynamic networks. This optimization is based primarily on the reduction of the size of link-state tables exchanged in TC messages, as well as the number of transmissions necessary to flood the messages. The key to reducing the number of transmissions is the use of Multipoint Relays (MPR). These are nodes selected to distribute TC messages. That is, only MPR nodes participate in flooding and, as such, the number of message transmissions can be substantially reduced, specially in dense networks. MPRs have to be selected in such a way that TC messages can reach every node in the network.

The exchange of HELLO messages between nodes permits them to discover their one-hop and two-hop neighbors. It also permits a node to derive the quality of the links connecting it to its one-hop neighbors, which can be used to compute the link-cost. Based on the information obtained from HELLO messages, every individual node selects a subset of nodes from its set of one-hop neighbors and declares them as MPRs. These are the nodes responsible for forwarding TCs messages broadcasted by the node. The MPR set of a node is a minimum set of nodes that have connectivity with every two-hop neighbor of the node.

The exchange of TC messages can be done periodically or can also be triggered by changes in the topology (e.g. link break). Nodes only include the links to their MPRs in the tables, thus reducing the size of TC messages and saving bandwidth.

2.4.3 Single-path reactive routing

2.4.3.1 Ad hoc On-Demand Distance Vector (AODV) Routing

AODV is a reactive routing protocol for MWNs developed by Perkins, Royer and Das [94, 95]. As the name indicates, it is a distance-vector routing protocol. To avoid the count-to-infinity problem of other distance-vector protocols, it uses sequence numbers on route updates, a technique introduced by DSDV.

When a node needs to establish a connection and has no route to the destination, it broadcasts a request (Route Request -RREQ- message). Neighboring nodes receiving this request forward the message, and record the node that they heard it from, thus effectively creating an "inverse" route to the source of the message. Other nodes which receive the RREQ continue forwarding the message and recording the route (next hop) to reach the source. In this manner, the request is flooded through the whole network.

When the destination, or any node that already has a route to the destination, receives the RREQ, it no longer propagates the request and instead sends a Route Reply (RREP) message via unicast back to the source using the inverse route. The propagation of the RREP message serves to establish routes to the destination in all nodes along the inverse route. The source node, upon receiving the RREP, can now route packets to the destination. The flooding of the RREQ message can lead to nodes receiving multiple copies of the message, indicating multiple different inverse routes to the source. A node will select the minimum-cost route (e.g. minimum hop count route). Similarly, if the source node receives multiple RREPs, it will select the minimum cost route to the destination. Unused entries in the routing tables are purged after a time. This means that while routes are being used the information remains in the routing tables.

Sequence numbers are used to identify the most recent routes. When a source sends a RREQ, it includes its own sequence number (source sequence number) and the last known sequence number of the destination (destination sequence number) in the message. Nodes update the inverse route to the source with the source sequence

number. Additionally, nodes that receive the RREQ and have a route to the destination with sequence number (destination sequence number) older than the one included in the RREQ cannot send RREPs.

In the case of link failure, the node that detects the failure sends a route error (RERR) back to the source node/s that are using the broken route. The source node will then repeat a route discovery. To detect link failure, nodes can periodically send HELLO messages. If a node does not receive a HELLO message from a neighbor in a specified time it concludes that the link no longer exists.

The advantage of AODV is that it creates no extra traffic once routes are established. A disadvantage is that it requires more time to establish a connection, and the initial communication to establish a route is heavier than other approaches.

2.4.3.2 Dynamic Source Routing (DSR) protocol

DSR is an on-demand routing protocol developed by Johnson, Maltz and Hu [53, 54] for MANETs. Its main characteristic is that it uses source routing instead of relying on the routing table at each intermediate device. That is, a source node includes the route (sequence of node addresses) to reach the destination in data packets, and intermediate nodes forward the packets based on this route.

The protocol has two phases: a discovery phase and a route maintenance phase.

When a node needs to establish a connection and currently has no route to the destination, it initiates route discovery by sending a route request (RREQ) message. This process is similar to the one described for AODV above. When a node n receives a RREQ, it checks if it already has a route to the destination. If it does, it sends a Route Reply (RREP) back to the source containing the route from n to the destination. Intermediate nodes add their address in the RREP on the way back to the source and in this way, upon receiving the reply, the source has the complete route to the destination. In addition, every node that receives or overhears (in promiscuous mode) a routing message can store and use the route contained in that message.

The route maintenance phase is invoked when the topology changes. If a node detects a link in a route is broken, it first attempts to find an alternative route in its routing table, and if no route is found it will send an error message to the source; the source will then invoke a new route discovery.

One of the main disadvantages of DSR is considerable routing overhead due to the source-routing mechanism employed. This is because every data packet needs to include the complete route from the source to the destination in its header. This overhead is directly proportional to the path length. An optional extension to DSR, named the DSR Flow State Extension, was proposed to reduce this overhead. This extension works by allowing the routing of most packets without an explicit source route header in the packet, thus substantially reducing overhead. The technique used is called implicit source routing, first described in [46] and now included in the standard [54]. Flow state extensions introduce a so called flow table for each network node, making DSR a stateful routing protocol. For each flow a node forwards there is one entry in the flow table which records the next hop address.

2.4.3.3 Dynamic MANET On-demand (DYMO) protocol

The Dynamic MANET On-demand (DYMO) [16] is a routing protocol currently under development and intended to be an evolution of AODV. It can be regarded as a simplified combination of previous reactive routing protocols (AODV,DSR) and uses distance vector routing like AODV to maintain loop-free routes.

DYMO defines only the basic elements required for reactive routing: route discovery and route management. It is intended to be extensible. When a source needs a route to a destination it initiates a query flood in the form of a route request (RREQ) dissemination. This message is flooded through the network until it reaches the destination. In the process, reverse routes toward the source are formed. When this message reaches the destination it replies with a route reply (RREP) sent via unicast to the source, forming forward routes to the destination along the path. Nodes receiving routing information in the form of RREQs and RREPs check it's usefulness via update rules. If it is loop-free and better than previous information the route is updated and the routing message propagated, or in the case of the destination a RREP is generated. Only one path is maintained at a node at any given time. Loop-freedom is guaranteed by use of destination sequence numbers. Sequence numbers enable nodes to determine the order of DYMO route discovery messages, thereby avoiding use of stale routing information. A new route is deemed superior if it is loop-free and shorter than a previous route.

2.4.4 Multi-path routing

2.4.4.1 Temporally-ordered routing algorithm (TORA)

TORA [92] is an on-demand hop-by-hop protocol that relies on inter-nodal coordination to compute multiple loop-free paths. Route discovery is realized via an adaptation of LMR [21] which builds a destination-oriented directed acyclic graph (DAG). It uses a modified partial link reversal technique to reorientate the DAG in the presence of link failures, avoiding instability in case of network partitioning that occurs in other link reversal algorithms [29].

TORA [92] is adaptive, efficient and scalable, suitable for highly dynamic MWNs. The main feature of TORA is that control messages are localized to a very small set of nodes near the occurrence of a topological change. To achieve this, nodes maintain routing information about adjacent nodes. The protocol has three basic functions: Route creation, Route maintenance and Route erasure. By maintaining multiple routes to the destination, topological changes do not require any reaction. The protocol reacts only when all routes to the destination are lost. In the event of network partitions the protocol is able to detect the partition and erase all invalid routes.

Limitations of TORA include the requirement of in-order delivery of control packets and the fact that it incurs in high inter-nodal coordination overhead in presence of frequent topology changes. Additionally, path disjointness is not considered.

2.4.4.2 Multipath DSR

The DSR protocol described previously in section 2.4.3.2 can be used to find multiple paths between a source and destination with minimal modifications. The RREQ and RREP messages contain the route traversed by the message. During a route discovery, RREQs can traverse different paths to reach the destination. Any node forwarding or overhearing these packets will thus be able to store multiple different routes to the source of the message, simply by extracting the route from the message. Instead of replying only to the first received RREQ as is the default behavior of DSR, the destination node can send an additional RREP for a RREQ that carries a disjoint route compared with the routes already replied.

In [88], A. Nasipuri and S. R. Das demonstrate that the use of multiple paths in DSR can keep correct end-to-end transmission for a longer time than a single path. In

other words, the frequency of route discoveries is much lower if a node keeps multiple paths to a destination. This is the first extensive study on performance benefits of multipath routing in MANETs.

However, the method used by DSR to forward RREQs severely limits the set of link-disjoint and node-disjoint paths found in many cases because the intermediate nodes drop every duplicate RREQ that may comprise another link or node-disjoint path. Permitting nodes to forward every duplicate RREQ, however, is not a viable option because this will result in substantial flooding overhead.

2.4.4.3 Split Multipath Routing (SMR)

Split Multipath Routing (SMR) [67] is another multi-path variant of DSR. SMR introduces a different RREQ propagation mechanism in order to compute "maximally" disjoint paths that Multipath-DSR cannot find. Maximally disjoint paths are defined as those with the least number of nodes in common (which will be node-disjoint paths if they exist). The protocol discovers multiple routes on demand, one of which is the path with the least delay.

When a source needs a route to a destination but no information is available, it floods a RREQ message over the entire network. In contrast to DSR, nodes can forward duplicate RREQs. In this way, several duplicates that traversed the network over different routes reach the destination. The destination then selects multiple disjoint paths, and sends RREPs back via the chosen routes. The complete route information is contained in the RREQ packets. Furthermore, in contrast to DSR, intermediate nodes are not allowed to send RREPs, even when they have route information to the destination. This allows the destination to receive all the routes so that it can select maximally disjoint paths.

Because nodes forward duplicate RREQs, SMR introduces a mechanism to control flooding overhead. Instead of dropping all duplicate RREQs, intermediate nodes only forward those which used a different incoming link than the first received RREQ, and whose route hop count is not larger than that of the first received RREQ.

The destination node will select the two routes that are maximally disjoint. One of the two routes is the route with the shortest delay: the path taken by the first RREQ the destination receives. This path is sent back to the source as soon as the destination receives the RREQ, to minimize the time the source has to wait to start

sending packets. The RREP includes the entire path so the intermediate nodes can determine where to forward the packet.

After having sent the first RREP, the destination waits a certain amount of time to receive more RREQs and determine all possible routes. The destination can then determine the maximally disjoint route to the already replied route from among all paths received. In the case where there is more than one maximally disjoint route, the shortest one is chosen. If there is more than one shortest path, the one that delivered the RREQ the fastest is selected. The destination then sends a second RREP along the chosen maximally disjoint path.

An important limitation of SMR is that, while it attempts to reduce the number of flooding initiations, it doesn't reduce routing overhead considerably. It is because each flooding in SMR requires much more control packets than that of DSR.

2.4.4.4 Ad-hoc On-demand Multipath Distance Vector (AOMDV)

AOMDV [75] is a reactive multipath routing protocol based on AODV. In each route discovery it can find multiple (edge-disjoint or node-disjoint) routes between a source and destination. The main goal of the protocol is to improve fault-tolerance, by using alternate routes when a path fails. A new route discovery is only needed when all available paths fail, thus avoiding the latency and high overhead of route discoveries.

The protocol seeks to take advantage of the route discovery process of AODV to find multiple paths, with minimal additional overhead. To form multiple reverse paths, nodes listen to duplicate copies of RREQs. Note that because taking into account duplicate RREQs can induce route loops, a mechanism is developed to avoid this. When nodes advertise routes (by the forwarding of RREQ and RREP messages) they advertise the hop-count of the longest available route to the destination. Nodes will only accept paths with higher sequence number and lower advertised hop-count than one available at the node. This mechanism avoid the formation of loops.

Disjointness is calculated distributively without the use of source-routing based on the *first-hop* field. This field indicates the first hop in the path followed by the RREQ. Nodes in a path know the first-hop and next-hop. Two paths are link-disjoint if they have unique next hops as well as unique last hops. Note that because this condition is maintained at all intermediate nodes in the path during propagation, the resulting paths are link-disjoint.

One of the main limitations of AOMDV is that it fails to catch many opportunities of finding existing disjoint paths. This is because the RREQs are rarely re-broadcasted and due to limitations in the use of the first-hop field.

2.4.5 Routing metrics

Until recently, the *hop-count* metric has been the most extensively used metric by MWN routing protocols. The main reason is that it can be measured very quickly and efficiently, making it particularly suitable for dynamic networks where nodes can frequently move. Traditionally, most development on MWNs focused on mobile ad hoc networks, hence the use of the hop-count metric.

A major limitation of the hop-count metric is that it provides no information on the quality or capacity of links. This can easily lead to the choice of suboptimal routes. In other words, the choice of the route with minimum number of hops will not always be the route that offers more capacity. This is because it does not take into account the following factors:

- **Packet loss rate**. It is likely that routes with less number of hops employ "longer" links, i.e. links where the distance between the transmitter and receiver is larger. Increased distance between a transmitter and receiver will reduce the signal-to-noise ratio, which can increase the link loss rate and make it more prone to suffer from interference.

- **Link rate**. In a MWN links can transmit at different rate, depending on the technology and modulation scheme used. This is normally related to the above factor, where links with lower quality transmit at lower rates. Routing through less hops but slower links can provide less throughput than routing through more (but faster) links.

- **Load**. By not taking into account the load in the network, relying on hop-count can result in heavily congested routes that traverse the same links [97].

- **Interference**. Hop-count is not interference-aware and cannot find paths with low intra-flow or inter-flow interference. For example, in multi-channel networks it is not capable of favoring paths with more channel diversity.

Lately, there has been increasing interest in MWNs where nodes are mostly stationary, such as Wireless Mesh Networks. This has prompted the study and development of more advanced routing metrics, suitable for wireless multi-hop networks. The goal of these metrics is to consider both the path length and the quality of links that compose it. Examples of these metrics will be described in the following subsections. A limitation of more advanced metrics is that, as nodes move, the metrics cannot quickly track changes in link quality, making the hop count metric more suitable for networks with mobile nodes [26].

Although taking into account the current load in the network is desirable to avoid congested areas and balance load, it can also easily lead to route instability in many routing protocols. This will be discussed in further detail in section 2.4.6.

2.4.5.1 Expected Transmission Count (ETX)

The ETX metric, developed by De Couto et al [22], estimates the number of expected transmissions to correctly send a packet over a link. This number varies from one to infinity. An ETX of one indicates a perfect transmission medium, whereas an ETX of infinity represents a completely non-functional link.

Let l_{mn} denote a bidirectional link between nodes m and n. Let p denote the probability that a packet transmission between m and n fails. We will assume that a successful transmission requires link-layer acknowledgment, as in 802.11 DCF. If p_f and p_r denote the packet loss probability in the forward and reverse direction of bidirectional link l, p is calculated as:

$$p = 1 \quad (1 \quad p_f) \times (1 \quad p_r) \tag{2.3}$$

Let the probability that the packet will be successfully delivered from m to n after k attempts be denoted by $s(k)$:

$$s(k) = p^{k \ 1} \times (1 \quad p) \tag{2.4}$$

Finally, the expected number of transmissions required to successfully deliver a packet from m to n is denoted by ETX:

$$ETX = \sum_{k=1}^{\infty} k \times s(k) = \frac{1}{1 \quad p} \tag{2.5}$$

To measure p, De Couto proposes each node to periodically broadcast short probe packets once every T seconds. Every receiver node collects these probe packets for a period P. The packet loss rate is obtained by:

$$p = 1 \quad \frac{probes\_rx(P)}{P/T} \tag{2.6}$$

where $probes\_rx(P)$ is the number of probe packets received during the period P.

The ETX of a path is the sum of the ETX of its links. Advantages of this metric is that it takes into account the packet loss ratio of each link, the asymmetric quality of bidirectional wireless links (in the forward and reverse direction), and the path length. Note that increasing number of hops will increase the ETX metric. Because the ETX metric has the additive property, it can be incorporated into other, traditionally shortest path-based, on-demand or proactive routing schemes with minimal changes to the routing protocol.

A limitation of ETX is that it only considers loss rates of the links and not their data rate. Additionally, it does not take load or interference directly into account. Note, however, that load and interference (specially in single-channel networks) can indirectly influence the metric, causing the ETX of links to increase.

2.4.5.2 Expected Transmission Time (ETT)

ETT is a metric developed from ETX, by Draves et al [27]. It basically extends the ETX metric to take into account the rate of links.

The value of ETT is taken as $ETT = ETX \times \frac{S}{B}$ where ETX is the expected transmission count of the link, S denotes packet length, and B refers to the bandwidth of the link. Essentially, it measures the expected time to transmit a packet on a link. The ETT of a path is the aggregate ETT of the links in the path, and measures the end-to-end delay.

2.4.5.3 Weighted Cumulative ETT (WCETT)

The WCETT metric was proposed jointly with ETT in [27], and is specifically designed for multi-radio multi-channel networks. It shares the benefits of ETT and additionally seeks to find paths with low intra-flow interference, which in the context of multi-channel networks means paths where links have high channel diversity.

The WCETT of a path is measured as:

$$WCETT = (1 \quad \alpha) \sum_{i=1}^{L} ETT_i + \alpha \max_{1 \leq j \leq K} T_j \tag{2.7}$$

where ETT_i is the expected transmission time (ETT) of link i in a path of L links, α is a tunable parameter $0 \leq \alpha \leq 1$, K is the total number of channels available in the system and T_j is the sum of the transmission times of the path's links on a particular channel j. The total transmission time on a channel j is calculated as:

$$T_j = \sum_{\text{link } i \text{ on channel } j} ETT_i \qquad 1 \leq j \leq K \tag{2.8}$$

The first part of Equation 2.7 provides the sum of ETTs of every link on the path and represents the end-to-end delay factor. The second part is the channel diversity factor and gives more weight to paths where many links are assigned to the same channel. In this way, it favors paths with more channel diversity. Note, however, that it does not accurately capture intra-flow interference because it does not take into account the separation between links using the same channel. In other words, it will penalize links transmitting in the same channel even if they do not interfere. The tunable parameter α provides a balance between both factors of the equation.

An important characteristic of WCETT is that it is not *isotonic*. This means that, depending on the routing protocol used, it can cause loop formation within a network.

WCETT does not take load or inter-flow interference directly into account.

2.4.5.4 Metric of Interference and Channel-switching (MIC)

The MIC metric was designed by Yang et al. [123] to achieve the benefits of all the previously discussed metrics, and at the same time avoid intra-flow and inter-flow interference.

The metric is composed of two parts: the Channel Switching Cost (CSC) of links, and the Interference-aware Resource Usage (IRU) factor of links. The MIC of a path p is defined as:

$$MIC(p) = \alpha \sum_{\text{link } l \in p} IRU_l + \sum_{\text{link } i \in p} CSC_i \tag{2.9}$$

$$IRU_l = ETT_l \times N_l \tag{2.10}$$

$$CSC_i = \begin{cases} w_1 & \text{if } ch(i) \neq ch(prev(i)) \\ w_2 & \text{if } ch(i) = ch(prev(i)) \end{cases} \tag{2.11}$$

where $0 < w_1 < w_2$, $ch(i)$ denotes the channel assigned to link i and $prev(i)$ is the link preceding i in path p.

The CSC of a link aims to avoid intra-flow interference. The metric favors paths with more channel diversity, in a similar way to the WCETT metric. This is done by assigning more weight to paths where two consecutive links use the same channel.

The IRU of a link aims to avoid inter-flow interference. N_l denotes the number of nodes affected by the transmission of link l. More specifically, the neighboring nodes which, when using CSMA-based MAC protocols, wait for the transmission to be completed. The IRU factor therefore takes into account the time required for transmission on the link, and how this affects neighboring nodes. Note that it does not measure actual interference with current traffic, because it does not consider if the nodes are transmitting or not. Furthermore, the different degrees of interference with each node (e.g. based on signal strength) are not considered. In summary, the IRU metric favors paths which can potentially interfere with less nodes, independently if the nodes carry traffic or not, and independent of accurate interference predictions.

The parameter α represents the inter-flow interference normalization factor for a network having N number of nodes and is estimated as $\alpha = \frac{1}{N \times \min(ETT)}$. Therefore, the MIC metric provides a routing metric that balances the effects of both the intra-flow and inter-flow interferences.

The MIC metric does not take load directly into account. It is designed purposefully in this way to reduce network instability. The drawback is that it will not adequately balance load. Additionally, like WCETT, MIC is not isotonic.

2.4.5.5 Interference-Aware routing metric (iAWARE)

The iAWARE metric was proposed by Subramanian et al [113] with the purpose of finding paths that take real interference into account, based on the current network

traffic. The iAWARE metric uses the same path metric formula as WCETT, but replaces ETT with the iAWARE metric defined as

$$iAWARE_l = \frac{ETT_l}{IR_l} \tag{2.12}$$

where IR_l is the Interference Ratio for link l_{mn} between nodes m and n, defined as:

$$IR_l = \frac{Noise_m}{Noise_m + \sum_{k \in IS(m) \quad \{n\}} \theta(k) P_m(k)} \tag{2.13}$$

where $Noise_m$ is the background noise at node m, $IS(m)$ is the set of nodes that can interfere with node m, $P_m(k)$ is the signal strength of a transmission from node k at node m, and $\theta(k)$ is the normalized rate at which node k generates traffic averaged over a period of time. Note that $0 < IR_l \leq 1$, and that a higher interference results in a smaller value of IR_l. The set of interfering links is calculated based on the Physical interference model (see section 2.2.3.2).

Like WCETT, the iAWARE metric is not isotonic, and therefore it cannot be applied to link-state routing protocols. In addition, the inclusion of the normalized rate parameter $\theta(k)$ makes the metric load-dependent, which also means that careful use of the metric is necessary to avoid routing instability. Finally, it may not be possible to measure $P_m(k)$ due to practical issues (see section 2.2.3.2).

2.4.6 Route stability and load-sensitivity

An important requirement of routing protocols and metrics is that they must ensure stability of the network, by avoiding frequent route changes. Frequently changing (*unstable*) path weights can be very harmful to the performance of any network. On the one hand, frequent changes can lead to a high volume of route update messages. On the other hand, it is also likely that routing protocols may not be able to converge on time under frequent updates, disrupting network operation.

The stability of a path weight depends on the type of path characteristics that are captured by the routing metrics, which can be either *load-sensitive* or *topology-dependent*. Load-sensitive metrics assign weights based on the current load that affects a link. Examples of load-sensitive metrics include Degree of Nodal Activity [44], and Number of Congested Nodes [56]. When using load-sensitive metrics, the weight of a route may change frequently as new flows arrive or existing flows end, leading to

network instability [108]. On the other hand, topology-dependent metrics assign a weight to a path based on the topological characteristics of the path, such as the hop count and link rate in the path. Therefore, topological-dependent metrics are more stable, specially in static MWNs.

Of the routing metrics discussed in section 2.4.5, hop-count is the only metric which, in practice, only depends on the topology. As a result, it can guarantee route stability, specially in static networks or those with low mobility. However, it cannot be expected to balance load or to find high throughput paths. The routing metrics ETX, ETT, WCETT and MIC, although carefully designed to avoid sensitivity to load, in practice are affected by load. This is because more traffic generates more interference and contention, which can influence the packet loss rate of links. This is specially true in single-channel networks. The study by Ramachandran et al. [100] shows route instability in a static wireless mesh network when the ETT metric is used.

The use of a load-sensitive routing metric can affect different routing protocols differently, depending on their characteristics. In other words, the stability of routes when using a load-sensitive metric will depend on the type of routing protocol used. For example, reactive routing protocols can usually be designed to ignore frequent changes in path weights. When using DSR (see section 2.4.3.2), the route to a destination is only found by a node when it needs to send traffic, and does not get updated as long as the route still exists. Therefore, metric changes do not generally cause flows to change their routes, and the stability of routes is not affected by frequently changing route metrics. In contrast, the use of load-sensitive metrics in proactive protocols may cause frequent route updates that affect the paths of many flows. As a result, these metrics have a high chance of creating network instability if traffic variations are frequent and large. Note that due to the small scale of MWNs, such variations are possible.

While it may seem that topology-dependent metrics are more suitable for MWNs, the truth is that load-sensitive (or *load-aware*) routing protocols are necessary to be able to balance load in the network and effectively utilize its capacity. However, they must overcome the important challenge of maintaining route stability in the network while being able to adapt and converge rapidly to changes in traffic conditions. Achieving these goals is a major challenge for distributed routing protocols where, in general, to maintain route stability it is necessary to give up on adaptivity. While on the other

hand, achieving high adaptivity is very difficult due to high route update overhead and the impossibility of fast convergence.

2.5 Multi-radio multi-channel networks

The use of multi-radio multi-channel networks is seen as a promising strategy to substantially increase the capacity of Wireless Mesh Networks. As explained in section 1.1.1, the fact that mesh routers are stationary, resource-rich and intended to provide a ubiquitous high-speed backbone means that they can be connected to a permanent power source, are not limited in size or processing power, and can be equipped with multiple radio interfaces and/or antennas.

Intelligent use of radios, channel assignment (CA) and routing is crucial to effectively use multiple channels and exploit available capacity in multi-radio multi-channel networks. As explained in section 2.3.1.2, there is a close relationship between CA and routing. To maximize performance, the two problems should be treated jointly rather than separately. However, in practice the joint problem is too hard to solve optimally. One frequently used approach to the joint problem is to solve the CA and routing problems separately, in some instances iterating over the two phases to improve the overall performance. We will first discuss each problem in isolation, finally presenting some work on the joint problem.

2.5.1 Radio usage policies

The radio usage policy determines how radios are used in multi-radio multi-channel networks. In particular, it affects: (a) the choice of interface to communicate with a particular neighbor (interface-to-neighbor binding) and (b) the choice of when and for how long to bind a channel to an interface. Channel-assignment algorithms (responsible of determining which channel to assign to an interface), will be discussed in the following sections.

2.5.1.1 Interface-to-neighbor binding

The interface-to-neighbor binding determines the interface/s a node uses to communicate with a particular neighbor. This affects routing, i.e. when a node needs to send a packet to a neighbor the routing layer must know through which interface to send

the packet. It is possible for a node to employ multiple interfaces to communicate with a neighbor. In this situation, the interfaces must be tuned to different channels, and the nodes have to share more than one common channel. This will essentially provide increased bandwidth (via multiple non-interfering links) to communicate with the neighbor. Note that exploiting this strategy complicates routing. One approach to use multiple interfaces simultaneously to communicate with one neighbor is to use them in a round-robin manner on a packet-by-packet basis. This, however, can result in packets arriving out of order at the receiver, and more so at the destination node, which can severely degrade the throughput of higher layer protocols such as TCP.

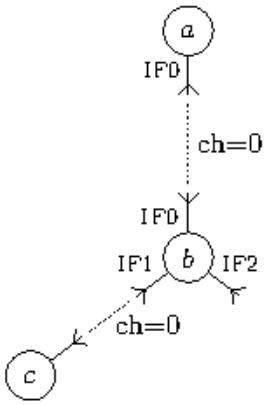


Figure 2.5: Static interface-to-neighbor binding - In this example, interface 0 (IF0) of node b is used exclusively for communication with a. CA decides that channel 0 is to be assigned to edges (a, b) and (b, c), requiring two interfaces due to static binding.

Interface-to-neighbor bindings can be static or dynamic. Static bindings permanently or semi-permanently assign one interface or set of interfaces to communicate with the same neighbor. Decentralized channel allocation strategies may rely on static bindings to prevent a local channel re-assignment decision from causing channel re-assignments across the whole network, also known as *ripple effect* (see [80, 102]). For example, in the Hyacinth [102] architecture, the set of radios that a node uses to communicate with its parent node, termed UP-NICs, is disjoint from the set of radios the node uses to communicate with its children nodes, called DOWN-NICs. The drawback of this approach is that static bindings can restrict channel assignment. Consider the example of Fig. 2.5. Suppose the channel assignment process requires allocating channel 0 to edges (a, b) and (b, c). Interface 0 of node b is used exclusively for commu-

nication with a. This means that a second interface of b must be assigned to channel 0 (e.g. interface 1) for communication with c, leaving only one free interface on b. In this situation, only two distinct channels at the most can be assigned to edges incident on b, when the interface constraint permits assigning three channels. Dynamic interface-to-neighbor bindings, on the other hand, can change between channel assignments and offer more flexibility in channel assignment. Note that a change in bindings will require updating the routing table.

2.5.1.2 Interface-channel usage policy

One common way of binding channels to interfaces is to use *static* binding. In this approach, a channel is assigned to an interface for a long period of time relative to packet transmission duration. Note that although static bindings generally have a duration of hours or days, channel re-assignment in the order of minutes or seconds is also possible. The key advantage of using static binding is that it requires no change to the existing IEEE 802.11 standard.

More complex schemes involving the *dynamic* switching of channels on a per-packet basis require some level of coordination among nodes (to ensure communication on a common channel), usually in the form of a modified MAC protocol. Another potential problem with dynamic bindings is that simultaneous communication with multiple neighbors using different channels may require frequently switching channels for each packet transmission, which can incur a significant latency penalty.

In hybrid (or mixed) bindings, one interface of a node is tuned to a fixed channel, while the others can dynamically switch to other channels. Kyasanur et al [64] propose one such approach, where a fixed interface at each node is tuned to one of the available channels (the least loaded channel in the vicinity of the node). The choice of channel is communicated to neighbors, and represents the channel on which the node wants to receive packets. When a node needs to send a packet to its neighbor, it switches one of its dynamic interfaces to the fixed channel of the neighbor and transmits the packet. Note that the same problems present with dynamic switching are present here.

2.5.2 Channel assignment preliminaries

The CA problem is usually stated in graph-theoretic terms. Let $G = (V, E)$ be a graph in which V is the set of nodes and E the set of bidirectional wireless links. A link exists

between two nodes if they can communicate directly (exact definition depends on the model used, see discussion of interference models in section 2.2.3). The graph G is the *connectivity* graph of the network.

The CA problem can be formulated in terms of assigning channels to the nodes in V, or to the edges in E. In the vertex formulation, the goal is to assign a channel to every radio of every node $v \in V$, while ensuring that two neighboring nodes (i.e. nodes connected by an edge) can communicate by sharing at least one common channel. In the edge formulation, the goal is to assign a channel to every edge $e \in E$, while ensuring that the number of distinct channels assigned to edges incident on a node does not exceed the number of interfaces of the node (called the interface constraint). Note that both formulations are equivalent, because a solution to the vertex formulation can be translated to a solution for the edge formulation and vice versa.

Any solution to the above problem is considered a feasible CA. The feasible set is clearly nonempty since the simple solution of assigning the same channel to every edge (or radio) is a feasible solution. At the same time, a random assignment of channels to radios or edges may not be feasible. For example, if channels are assigned to the edges incident on a node without considering the interface constraint, it is possible to violate the constraint by assigning more channels than available interfaces. A CA problem is an optimization problem which requires finding the feasible CA that optimizes a chosen performance metric.

In general, the objective of a CA problem requires minimizing the interference between edges. This requires modeling edge interference. In almost all existing work, this involves calculating, for every pair of edges $e, f \in E$, the degree of interference between the edges (when the edges are assigned to the same channel). Let $I(e, f)$ denote the interference between edges e and f. Depending on the interference model used, interference can be *binary*, i.e. $I(e, f) \in \{0, 1\}$, or it can be a real number $I(e, f) \in [0, 1]$ (*fractional* interference). When binary interference is considered, two edges either can or cannot be active simultaneously. Fractional interference, on the other hand, can be used to model the more realistic situation where interference affects the packet error rate (PER) of a link. The above information can be obtained from the protocol model or physical model of interference (see section 2.2.3 for more information). Additionally, many works build a conflict graph (section 2.2.3.4) based on interference information.

2.5.3 Channel assignment problems and algorithms

In this subsection we discuss some approaches to channel assignment that assume equal load on all links.

Some approaches to CA formulate the problem as an integer linear program (ILP). This was adopted by Das et al. [23] where the objective of their problem is to maximize the number of simultaneous transmissions achievable in the network. An equivalent formulation using the conflict graph was proposed by Jain et al. [52]. The problem consists in finding the largest possible independent set in the conflict graph. These solutions are not load-aware, and instead maximize the achievable instantaneous throughput in the network, by assuming that all edges carry equal load. The complexity of obtaining an optimal solution to ILP problems can make these approaches intractable even for moderately-size networks.

Other proposals approach the problem from a graph-theoretic perspective. The CA problem can be viewed as a coloring problem on the connectivity graph. Given a set of colors (i.e. the channels), the goal is to assign colors to edges, while satisfying the constraint that the number of distinct colors assigned to edges incident on a node is a most the number of interfaces of that node. The coloring found must optimize some objective function. When formulated as a coloring problem, it is easy to prove that the CA problem is NP-hard [74, 76].

Marina and Das [76] formulate the CA problem as a coloring problem. Assuming a binary interference model, let us denote $IE(e) = \{f \in E \mid I(e, f) = 1\}$ as the set of edges that interfere with edge e. The objective of the problem is to minimize the maximum size of IE(e) over all edges $e \in E$. The problem is shown to be NP-complete, and they also provide a heuristic solution. The algorithm proceeds in a single pass over all the nodes in the network and assigns channels while ensuring that the connectivity constraint is met. When there are multiple available choices for the channel to be assigned to a link, the choice is made based on minimizing the size of IE.

All the CA approaches discussed above assume the protocol model of interference (see section 2.2.3.1 and [43]). As explained previously, this is a simplified and inaccurate model, which fails to capture fractional interference as well as cumulative interference from multiple interfering transmitters.

2.5.4 Channel-aware routing

The problem of routing in a multi-channel MWN for a given CA requires finding routes which minimize intra-flow and inter-flow interference. In other words, the routing protocol and metrics must take into account the channels assigned to edges in order to select those paths with less interference. In section 2.4.5 we discussed various routing metrics, among which WCETT, MIC and iAWARE are channel-aware. The use of one of these metrics with a standard routing protocol such as those discussed in section 2.4 will permit finding channel-diverse routes, and in some cases also avoid inter-flow interference.

2.5.5 Joint routing and channel assignment

If the traffic demands in the network and load on each link is known, a CA algorithm can use this information to further optimize the capacity and achieve better utilization. Link load depends on the routing protocol. However, it is also true that the routing protocol needs to select routes based on the channel assigned to links. This mutual dependency between routing and CA imposes the need for joint algorithms.

In the last years there has appeared a large body of work on the subject of joint routing and channel assignment in WMNs. Most existing proposals can be classified broadly in two groups: centralized and offline solutions, and distributive and online solutions. In the following we briefly discuss some of the most relevant proposals.

Most of the centralized solutions described below calculate *flow*, which refers to traffic per unit of time traversing a link or node. These solutions allocate specific *flow* to links, and thus require throughput models to guarantee that such allocations are feasible (are schedulable).

Raniwala et al. propose a centralized solution [103] that repeatedly applies separate routing and channel assignment algorithms until a solution converges. The routing algorithm calculates link-*flow* allocation, while the channel assignment algorithm ensures that such routing is feasible. The solution is calculated offline assuming knowledge of a traffic matrix which specifies long-term average demands between all nodes, and thus intended to be valid for hours or days. Throughput is estimated based on a binary interference model.

In [8], Alicherry et al. propose a centralized approximation algorithm to solve a joint routing and channel assignment formulation. Their model assumes time-slotted synchronized medium access and a binary interference model in order to calculate interference-free scheduling. It requires knowledge of the traffic matrix and calculates splittable *flow*, where the total demand between a source and destination is generally routed through multiple paths. Tang et al. [114] use mathematical optimization to calculate *flow* allocation, routing and channel assignment. The problem does not require a traffic matrix but rather assumes that all nodes are active simultaneously and allocates *flow* to each to achieve max-min fairness. Throughput is estimated based on a binary interference model. Mohsenian-Rad et al. [81] present a MILP formulation to calculate topology, channel assignment and routing. Similar to the above, the problem assumes knowledge of traffic matrix, and calculates *flow*-link allocation. Avallone et al. develop a centralized offline heuristic for the joint routing and channel assignment problem [10]. It calculates link-*flow* allocation and channel assignment to maximize aggregate throughput, assuming all nodes are active simultaneously communicating with the external network. The method of precomputing *flow* calculates splittable *flow*, however, it does not balance load in the network, because maximum *flow* computations are independent of each other (calculated as single-source, single-sink problems). Although their model uses SINR, the interference model is binary (cumulative interference is not taken into account and neither is link error probability (fractional interference)). The authors propose a special forwarding paradigm to ensure that *flow* allocation is schedulable in conventional MACs. In [41], Gardellin et al. propose a method to solve a joint routing and channel assignment problem by decomposing the problem into several ILP optimization subproblems which are solved optimally in sequence and later combining the solution in a post-processing phase. This solution also calculates link-*flow* allocation and knowledge of the traffic matrix is required.

The main drawback of the offline centralized solutions described above is that they cannot be considered practical. Essentially, they are theoretical solutions which focus on algorithms to solve joint routing and channel assignment formulations. These formulations rely on unrealistic throughput and interference models, or require time-synchronized medium access. Moreover, these works lack practical solutions for measuring the network state, applying routing and disseminating channel assignments.

We now briefly present practical routing and channel assignment solutions. Most of these solutions, while practical, avoid load-sensitivity due to the issues discussed in section 2.4.6, and therefore do not balance load in the network. There is thus an important need for practical joint routing and channel assignment solutions capable of calculating routes and CA to optimize the performance given the current load in the network, while at the same time avoiding network instability.

In [102], Raniwala et al. propose a distributed solution based on their previous work [103], which periodically modifies routes and channel assignment distributively based on current network load. Routing forms a tree structure rooted at the gateway. A limitation is that the routing metrics proposed are only suitable for balancing traffic between gateways, but not inside the network, specially when TCP traffic is considered. In addition, routing and load-balancing are decoupled and there is no guarantee of convergence (both are load-aware and one affects the other). Furthermore, the load-sensitivity of the routing metric can lead to instability, because the required convergence time can be higher than the time between traffic variations in a WMN.

In [101], Ramachandran et al. propose a practical multi-radio WMN architecture. The online centralized channel selection algorithm (TIC) computes routes between the gateway and mesh routers and allocates channels to links on these routes. TIC requires that all nodes operate one of its radios on a common default channel in order to coordinate topology measurement and disseminate channel switch commands. Routing and channel assignment, however, are not load-aware.

Dhananjay et al. [25] propose a distributed channel assignment and routing protocol. The channel assigned to a node's interfaces depends on the hop-distance of the node to the selected gateway, with the purpose of assigning different channels to all links in the path to the gateway. Channel assignment in this approach is not load-aware. Kyasanur et al. propose a distributed link-layer protocol for interface-channel assignment and a multi-channel aware routing metric [64]. Each router chooses a channel least used by nodes in its neighborhood without coordination with other routers. Routing metric and channel assignment are not load-aware.

Chapter 3

Spatial separation of paths in MWNs

This chapter addresses the problem of measuring spatial separation of paths in a MWN without the use of positional information and finding paths with maximal spatial separation between two nodes. The first part explains and develops the PSD metric to measure path spatial separation using hop-count information. The second presents the design of the SD-PCA algorithm to find paths with maximal separation based on the PSD metric. Finally, the third part presents the design, implementation and evaluation of the SDMR spatially disjoint multi-path routing protocol.

3.1 Introduction and motivation

The capability of finding and using spatially separated paths can provide important benefits in MWNs. In single-channel networks, it may prove to be the only practical way of exploiting frequency spatial reuse and balancing load. Routing packets through paths separated in space can avoid inter-flow interference, improving network utilization and throughput. The interference between two or more paths which are physically close is sometimes referred to as *route coupling*. Route coupling is known to severely affect the performance of single-channel MWNs. The effects of load balancing and route coupling have been studied in [40, 55, 67, 93, 118, 119]. Measuring interference in MWNs is not trivial, specially in dynamic networks with mobility (this is explained in more detail in section 2.2.3.3). In the absence of interference information, a simple and effective

mechanism is to estimate interference based on the distance between nodes. In other words, separating paths based on the distance between their nodes will avoid the route coupling effect. This strategy will be employed in chapter 4 as part of a load-balancing solution for single-channel WMNs with multiple gateways, using techniques described in this chapter.

Another benefit of spatial separation is fault tolerance to situations of regional failures. This refers to the case where all paths that cross a specific area fail (e.g. all nodes in a region of space fail due to a blackout). If nodes have knowledge of multiple spatially disjoint routes to reach a destination, they can promptly switch to an alternate unaffected route. The use of multiple independent paths as a means of providing fault tolerance has been frequently proposed for MWNs in the form of multipath routing protocols. Most frequently, however, multipath protocols only focus on finding link-disjoint or node-disjoint routes (see section 2.4 for more information). The use of multipath routing has also been shown to reduce route discovery overhead in on-demand protocols, because nodes can switch to alternate paths when the current path fails, thus avoiding a new route discovery which accounts for the majority of protocol overhead [75].

Increased communications security, specially important in wireless networks, is another benefit of spatial separation. The idea consists in dividing a (possibly encrypted) message and distributing it through multiple independent paths. For the communication to be compromised, the multiple routes it traverses need to be compromised. Spatially-disjoint paths provide added security, because it reduces the probability of a device intercepting the communication of every path. Other situations that threaten the security of a communication include man-induced geographically localized failures or black-hole routers. The use of spatially disjoint routes can also help in these situations.

A straightforward way of measuring distance is with knowledge of the positional information of nodes and a method to calculate distance between node positions. A usual method is the calculation of Euclidean distance based on the geographical coordinates of nodes (latitude, longitude, height). However, location information may not be present at all nodes. As such, it is desirable to be able to estimate the distance between nodes based only on connectivity information. This chapter first demonstrates that the minimum hop distance between nodes correlates strongly with Euclidean distance in

many network scenarios. It then develops the Path Spatial Distance (PSD) metric, based on hop distance between nodes, to measure the distance of a path to a set of nodes. This metric can be used to measure distance between paths.

To use spatially disjoint paths in MWNs, it is important not only to be able to measure distance between paths in space, but it is also important to be able to find such paths. Most routing protocols rely on a partial view of the network and only find the shortest routes to a destination. In most cases, these routes are in close proximity of each other. For example, most proposed multipath protocols concentrate on finding link or node-disjoint paths. In practice, the spatial separation of paths in these cases is negligible.

Even with a global view of the network (i.e. the complete connectivity graph), it is not trivial to find spatially disjoint paths in the absence of positional information. More specifically, without positional information to guide the calculation of paths, it is difficult to calculate an optimal set of disjoint paths without exploring every possible path in the graph. For example, given a source-destination pair, it is not trivial to find two maximally spatially disjoint paths connecting the source-destination without resorting to measuring the distance between all possible paths connecting the endpoints. This can be prohibitively expensive. This chapter develops an efficient heuristic algorithm to find a pair of spatially disjoint paths connecting two nodes, when the topology graph of the network is known. This algorithm is called Spatially Disjoint Path Calculation (SD-PCA) and uses the proposed PSD metric.

Finally, this chapter also studies the design of a spatially disjoint multi-path routing protocol. Multipath routing can benefit notably from spatial disjointness. As explained in chapter 2, this routing scheme permits the establishment of multiple paths between a source and destination, and can provide various benefits over single path protocols, such as: increased reliability, load balancing, bandwidth aggregation, and secure communications. As explained above, each of these benefits can be enhanced by the use of spatially separated paths.

With global knowledge of the topology and the proposed SD-PCA algorithm, it is possible for a node to calculate spatially separated routes to a destination. With this in mind, routing protocols such as OLSR (described in [20] and section 2.4.2.3), where nodes have knowledge of the connectivity graph, can be modified easily to support spatially disjoint routing. However, routing protocols where nodes have global

knowledge of the network are usually proactive, known to incur in high overhead in dynamic MWNs. This chapter proposes a new reactive (on-demand) multipath protocol based on source routing called Spatially Disjoint Multipath Routing (SDMR), capable of finding spatially disjoint paths.

In the SDMR protocol, a node can discover the complete topology graph on-demand. In contrast to other reactive protocols, this permits a node to discover paths to multiple destinations with only one route discovery. Another important difference with common reactive protocols is that the route discovery mechanism is designed to efficiently obtain information from all geographical regions of the network. This information is used by a source node to build the topology graph, which is then used by the SD-PCA algorithm to calculate spatially disjoint paths.

The main contributions [34, 35, 37] of this chapter are as follows: (a) the PSD metric to measure path separation in space, which is shown to be congruent with the Euclidean distance; (b) a heuristic algorithm for computing a pair of maximally spatially disjoint paths between two nodes in a graph using the PSD metric; (c) the SDMR protocol being capable of finding in a reactive way spatially disjoint routes without location information; (d) analytical study of the overhead of the SDMR protocol in comparison with the well-known OLSR.

The rest of the chapter is organized as follows. Section 3.2 reviews related work. Section 3.3 develops and explains the PSD metric. Section 3.4 develops the Spatially Disjoint Path Calculation Algorithm (SD-PCA) and analyzes its complexity. Section 3.5 explains the SDMR protocol. In section 3.6 the overhead of SDMR is studied analytically. Section 3.7 shows and analyzes SDMR protocol performance based on simulations with *ns-2*. Finally, section 3.8 concludes the chapter.

3.2 Related work

Prior work has defined a few metrics to measure the relative degree of interference between paths, namely *coupling* [93] and *correlation* [119]. From the point of view of spatial disjointness, one immediate drawback of these metrics is that they are restricted to avoiding route coupling and as such cannot be used in cases where more separation between paths is needed other than the one necessary to avoid interference. For example, in some applications it may be desired that paths be further apart than interference

range (e.g. to avoid eavesdropping from a node between the two paths). These metrics also have practical limitations that difficult their use. The *correlation* metric implicitly assumes that the interference range is equal to the transmission range, which generally isn't true. Also, *correlation* is only valid for node-disjoint routes. *Coupling* is difficult to calculate in practice.

Other methods of avoiding interference include the use of multiple channels or directional antennas. The studies in [105, 107] are examples proposing the use of directional antennas to improve the performance of multipath routing and avoid route coupling. These proposals require additional equipment, and the technique only seeks paths which don't interfere with each other, and as such don't seek spatially separated paths.

To our knowledge few protocols explicitly attempt to find spatially disjoint paths. Most existing ones are location based and use geographic coordinates to measure the distance between paths. Examples are EFR [89], which applies the concept of electric field lines to find spatially disjoint routes, 3DMRP [110] and the work in [72, 117]. The drawback of location based protocols is that they rely on geographic information which is not always available.

Most proposed multipath protocols for MWNs focus on providing fault tolerance, i.e. the capability of switching to alternate routes when a route fails. Examples based on AODV [94] are AODVM [124], AOMDV [75] and the proposal in [83]. Other protocols use source routing such as a multipath extension of DSR [88] and SMR [67]. These protocols only consider link or node disjointness of paths (i.e. paths share no links or nodes, respectively), as this is generally sufficient from a fault tolerance perspective. Therefore they don't find spatially disjoint routes, and in practice routes found are very close to each other. More recent examples of multipath routing are the proposals for sensor networks in [17, 115]. Similar to the above protocols, they only consider node disjointness of paths.

AODVM/PD [85] is an on-demand multipath protocol based on AODVM [124] that attempts to minimize *correlation*. Besides the limitations of using the *correlation* metric, another drawback of AODVM/PD is that due to the route discovery mechanism of this protocol, it first chooses the shortest path and then attempts to find additional shortest paths with low correlation. This also limits the degree of separation obtainable. A similar protocol is proposed in [84].

The PSD metric proposed in this chapter does not require location information, provides more information regarding the degree of separation compared to the above mentioned metrics, and can be used for both disjoint and non-disjoint routes. Also, the SDMR protocol can discover and choose from paths that traverse all geographical regions of the network (not only k shortest paths).

3.3 Path distance metric

This work proposes a metric to measure the distance in space of a path to a set of network nodes. This set of nodes can refer to a group of nodes in a region of the network or can refer to another path. One of the main goals of this metric is to permit finding paths that do not interfere with other nodes or paths in the network, avoiding inter-flow interference in single-channel networks.

Even though other metrics have been proposed to measure separation between paths, such as *coupling* [93] and *correlation* [119], they present severe limitations in the degree of separation they can measure. These metrics measure based on the number of direct (1-hop) links between paths, whereas the metric proposed in this work is based on a close estimate of the distance in space between nodes.

The distance between nodes in space is given by their Euclidean distance. Accurate Euclidean distance can be calculated with precise knowledge of the geographical coordinates of nodes (latitude, longitude and elevation). However, this information may not be available at all nodes. Instead, this work proposes estimating Euclidean distance based on the minimum hop distance between nodes. This section first studies the correlation between both metrics and then develops the Path Spatial Distance metric.

3.3.1 Definitions and assumptions

- The network is modeled by an undirected graph $G = (V, E)$, where V is the set of nodes in the network and an edge $e \in E$, where $e = (u, v)$ represents a bidirectional wireless link between nodes u and v.
- A path p is a sequence of nodes such that there exists an edge in the graph connecting each node to the next node in the sequence.
- $|p|$ denotes the number of nodes in path p.

- p_i refers to the i^{th} node in path p, where $0 \leq i \leq |p| \quad 1$.
- Let $\text{hops}(p) = |p| \quad 1$ denote the number of hops in a path p.
- $d_h(m, n)$ is the distance in hops between nodes m and n, defined as the number of hops in a shortest path connecting them.
- $d_E(m, n)$ denotes the Euclidean distance between two nodes m and n.

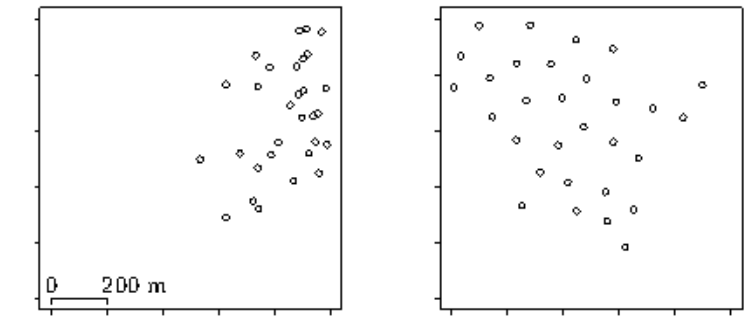
3.3.2 Correlation between hop distance and Euclidean distance

Experiments have been conducted to evaluate the correlation between hop distance and Euclidean distance in random MWN topologies. Given a random network topology, and based on the measured hop distance d_h and Euclidean distance d_E of every pair of nodes in the topology, the Pearson correlation coefficient of d_h and d_E is calculated.

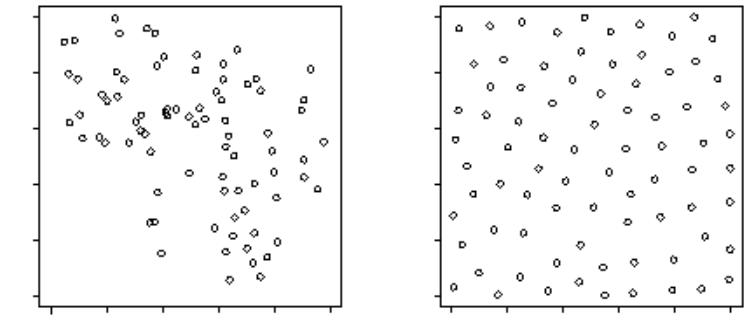
In every topology generated, nodes are distributed randomly in a square area of 1000 m $\times$ 1000 m. To generate connected topologies (i.e. where a path exists connecting every pair of nodes), nodes are placed in sequence in the following manner. For each node, a position is repeatedly selected at random, until a position is found such that the node is within transmission range of an already placed node. This rule does not apply to the first node placed, which can be located anywhere. The maximum transmission range of nodes in these scenarios is 150 m. In other words, the transmission power of nodes is assumed to be set such that links with an acceptable PER are established when the distance between transmitter and receiver is at most 150 m.

Two types of topologies have been generated: "sparse" and "dense". In the former, the minimum distance between two nodes is 100 m, while in the latter the minimum distance is 10 m. Dense networks admit more nodes in a given area of space than sparse networks. In each case, topologies with varying number of nodes have been generated. Note that the maximum number of nodes that can be placed in a sparse network is less than a dense network (sparse topologies have been generated with a maximum of 80 nodes while dense topologies have been generated with a maximum of 160 nodes). For each type of topology and number of nodes, 100 different random topologies have been generated. Figure 3.1 shows examples of the topologies produced in this manner.

The results of the experiments are shown in Fig. 3.2. Each point in the graph shows the average correlation coefficient of the distance metrics in the scenarios that



(a) Dense topology with 30 nodes (b) Sparse topology with 30 nodes



(c) Dense topology with 80 nodes (d) Sparse topology with 80 nodes

Figure 3.1: Example of sparse and dense topologies generated - Nodes are placed randomly in a 1000 × 1000 m area while ensuring the topology is connected. In dense topologies the minimum distance between nodes is 10 m while in sparse topologies it is 100 m. The maximum transmission range is 150 m.

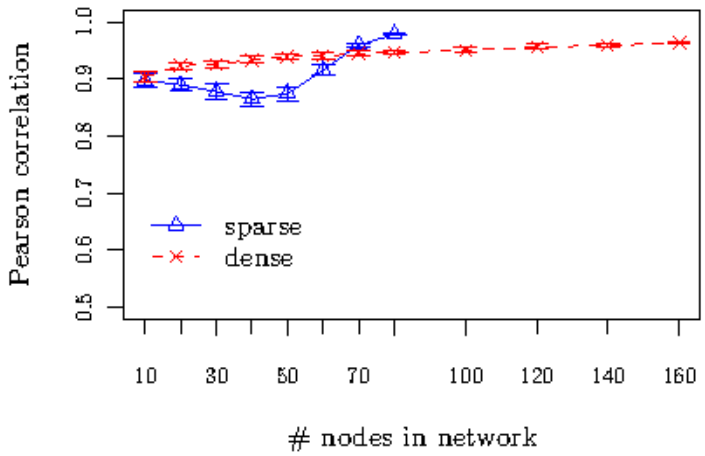


Figure 3.2: Correlation results between d_h and d_E - The figure shows the average Pearson correlation coefficient of the d_h and d_E metrics in the scenarios tested.

fall in that class. Confidence intervals are shown at the 95% level. As we can see, the correlation coefficient of d_h and d_E is notably high, and has a tendency to grow with more density and number of nodes.

3.3.3 Path Spatial Distance (PSD) metric

To define the path metric, it is first necessary to define the distance of a node n to a set of nodes N. This is defined as the minimum distance d_h from n to nodes in N, as specified by equation 3.1. This is illustrated in Figures 3.3 (a) and (c). Note that a set of nodes N can also refer to a path, as shown in Fig. 3.3 (c).

The Path Spatial Distance (PSD) of path p to a set of nodes N is defined as the arithmetic mean of the distance of nodes of p to N, as per equation 3.2. This is illustrated in Figures 3.3 (b) and (d).

$$\text{node\_dist}(n, N) = \min_{m \in N} \{d_h(n, m)\} \tag{3.1}$$

$$\text{PSD}(p, N) = \frac{1}{|p|} \sum_{n \in p} \text{node\_dist}(n, N) \tag{3.2}$$

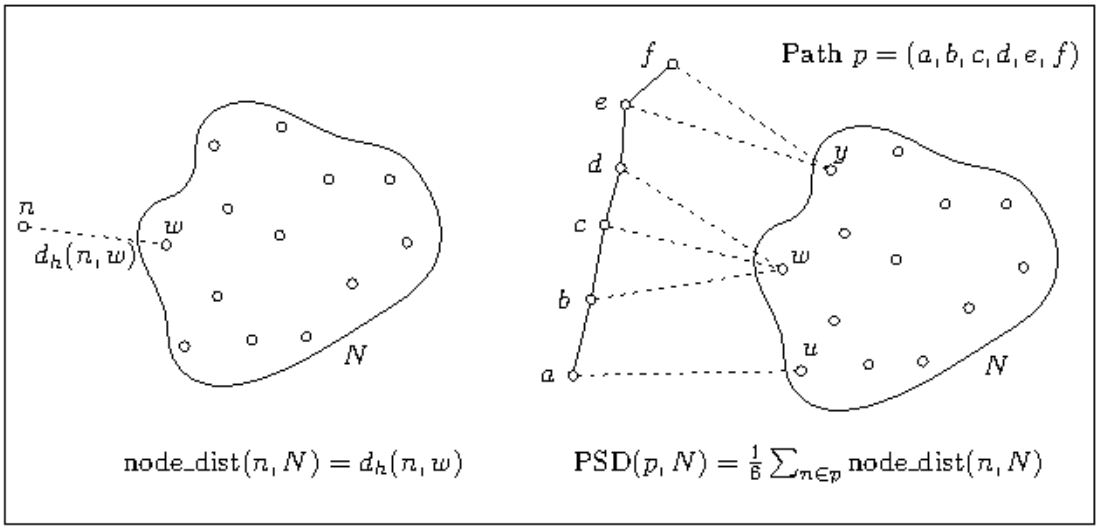
The PSD metric measures the distance of nodes of a path to a chosen set of network nodes. This metric can be used to measure the distance between two paths, or the distance of a path to a region of nodes. In the previous section it was shown that distance in hops between nodes can correlate highly with Euclidean distance. This means that the PSD metric is suitable for measuring spatial separation. Simulation experiments in section 3.7 show additional results regarding the correlation with Euclidean distance.

3.4 Spatially disjoint path calculation

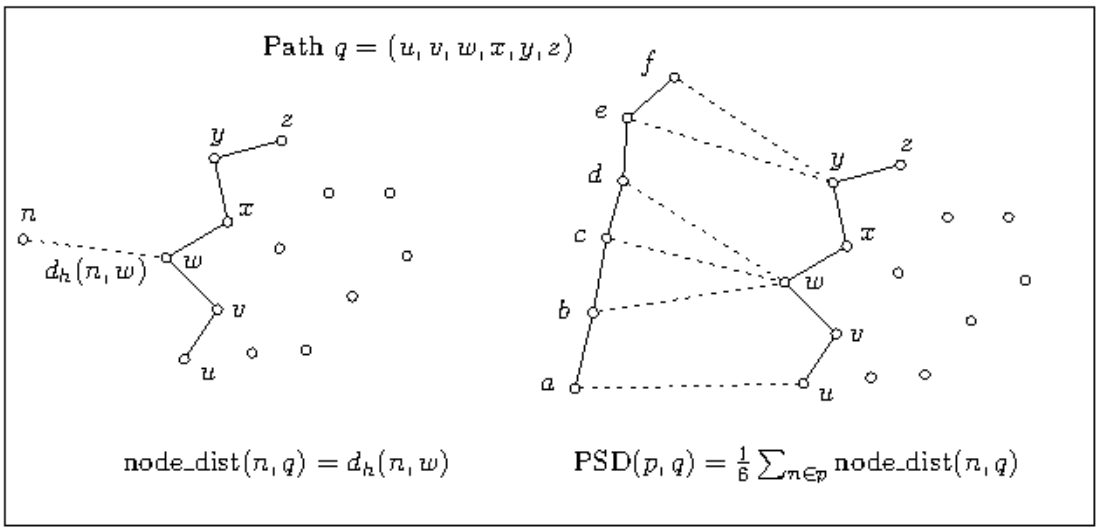
This section develops an algorithm to solve the problem of finding a pair of paths between a source and destination with maximal spatial separation.

3.4.1 Maximal spatially disjoint path problem

Given a graph $G = (V, E)$, a source $s \in V$ and destination $t \in V$, the problem consists in finding the pair of paths connecting s and t with maximum spatial separation (based



(a) Distance of node n to set of nodes N (b) Distance of path p to set of nodes N



(c) Distance of node n to path q (d) Distance of path p to path q

Figure 3.3: Path spatial distance metric - Dashed line segments connect a node with its nearest node in the set N or path q. The length of this segment represents the distance of the node to N or q.

on the PSD metric). Because the PSD metric (equation 3.2) is not symmetric, to calculate the distance between two paths, we use the following equation:

$$\Gamma(p, q) = \frac{\mathrm{PSD}(p, q) + \mathrm{PSD}(q, p)}{2} \tag{3.3}$$

Let P denote every path in G connecting s and t. The maximal spatially disjoint path problem ($\mathbf{P_{MSD}}$) can be formulated as:

$$\mathbf{P_{MSD}}: \max_{p,q \in P} \Gamma(p, q) \tag{3.4}$$

The objective of the proposed problem is to find the pair of paths in P with maximum separation, disregarding other factors such as the length of the paths. This is achieved because the PSD metric will favor separation of all nodes of p to nodes of q, and vice versa. In practice, the particular problem to solve in a network scenario will depend on the type of application, considering factors like: minimum required path separation, the path length, or trade-off between path separation and path length. This work focuses on the distance maximization problem, mainly to illustrate the capability of the proposed multipath routing protocol (SDMR) of finding spatially disjoint paths, up to any desired degree of separation. An example of an alternative problem which can be considered in more practical cases is the following:

$$\min_{p,q \in P} (|p| + |q|)$$

subject to:

$$\Gamma(p, q) \geq \Delta$$

which consists in finding the pair of shortest paths that satisfy a minimum separation Δ (e.g. three-hop separation to avoid interference or eavesdropping).

3.4.2 Spatially Disjoint Path Calculation algorithm (SD-PCA)

Given a set of paths P in G, equation 3.5 returns the pair of paths which maximizes $\Gamma(p, q)$:

$$\text{bestPair}_{\text{a}}(P) = \underset{(p,q)\in P\times P}{\arg\max}\ \Gamma(p,q) \tag{3.5}$$

$$\text{bestPair}_{\text{b}}(P_1,P_2) = \underset{(p,q)\ p\in P_1,q\in P_2}{\arg\max}\ \Gamma(p,q) \tag{3.6}$$

A brute force approach to solve $\mathbf{P_{MSD}}$ would be to apply $\text{bestPair}_{\text{a}}$ to all the paths from s to t. There are two problems with this approach:

- Equation 3.5 calculates $\Gamma(p,q)$ and compares $O(P^2)$ times, and
- the number $|P|$ of all possible paths between two nodes in a graph can grow exponentially with the number of nodes.

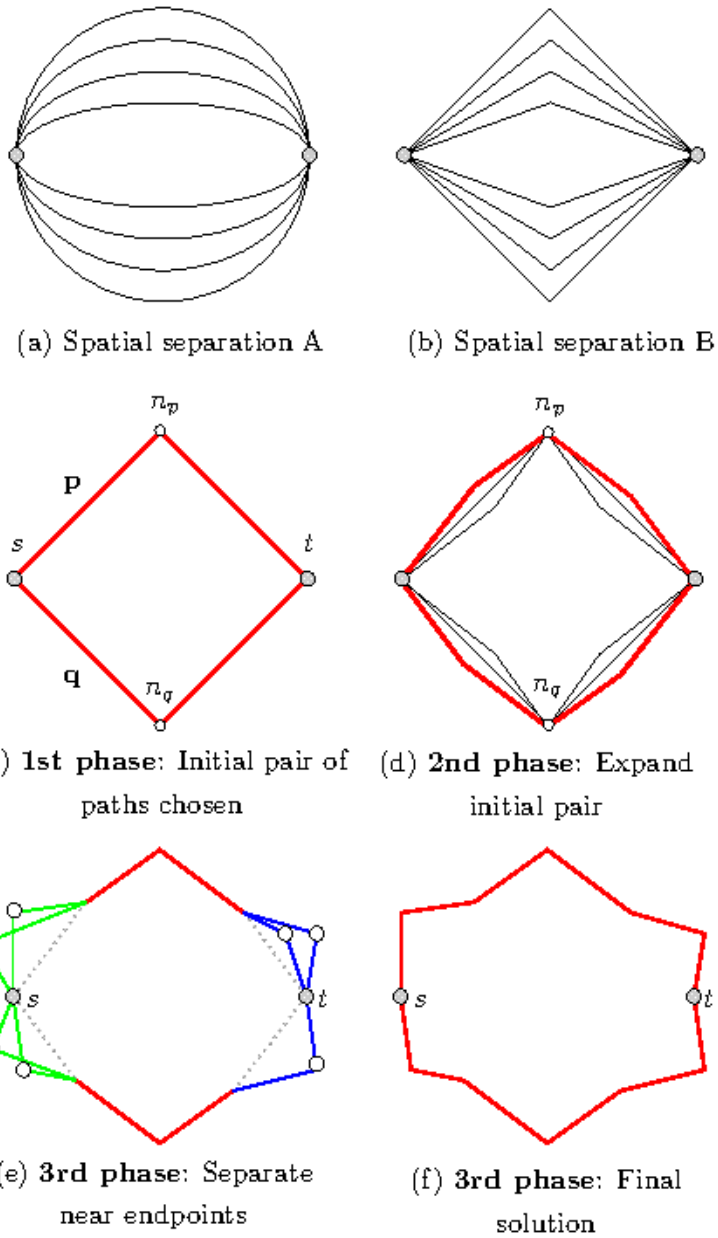
To make the problem tractable, this work proposes a heuristic algorithm to calculate an approximate solution, where the main idea is to find a reduced set of candidate paths between s and t from which to choose a pair of maximally spatially separated paths. The SD-PCA algorithm is developed and explained in the next subsections. Later we will analyze its run time.

3.4.2.1 SD-PCA overview

First we give a simplified explanation of the algorithm to convey the idea behind the heuristic, and illustrate with an example.

To obtain two or more spatially disjoint paths between a source s and destination t, two possible approaches are to choose from paths that resemble those in Fig. 3.4 (a) or (b). The first group offers more separation between paths. However, generating this set of paths requires positional information. The work in [89] is an example of this. On the other hand, generating a set of paths like those in Fig. 3.4 (b) without positional information is straightforward. We can do so by taking the nodes at roughly the same distance d_h to the source and destination, and for each of these nodes, generate a path by concatenating a shortest hop path from s to the node with a shortest path from the node to t. In Fig. 3.4 (b), we can consider the middle point of each path to be one of these nodes. Each one is called a *generating* node of the resultant path.

The SD-PCA heuristic uses this idea as a first phase. The set of paths obtained in this manner is very small (equal to the number of nodes at the same hop distance to s and t), which contributes to the efficiency of the algorithm. From this set, the pair



(a) Spatial separation A

(b) Spatial separation B

(c) **1st phase**: Initial pair of paths chosen

(d) **2nd phase**: Expand initial pair

(e) **3rd phase**: Separate near endpoints

(f) **3rd phase**: Final solution

Figure 3.4: Phases of the SD-PCA algorithm - This figure illustrates the steps performed by the SD-PCA algorithm to find a pair of spatially disjoint paths between a source s and destination t.

of most disjoint paths (p, q) is selected by applying eq. 3.5. Of the set of paths in Fig. 3.4 (b), the pair of most disjoint (p, q) is shown in Fig. 3.4 (c). We call n_p and n_q the generating nodes of p and q, respectively. This is the most important step of SD-PCA because the final solution will be based on this pair of paths. The next steps attempt to find paths which are similar or close to this pair and improve the solution. As we can see, the heuristic can offer a pair of paths with considerable separation, even though it doesn't guarantee the pair of most disjoint paths of **all** possible paths between s and t.

In the previous step, paths were generated with only one shortest path from the endpoints to the generating nodes. This helps limit the number of paths generated. But in a graph there can be multiple shortest paths between two nodes. In the second phase, one obvious way of expanding upon the current pair of disjoint paths is to generate new paths from their generating nodes, this time with all combinations of shortest paths, i.e. concatenate each shortest path from s to n_p with every shortest path from n_p to t, and the same with n_q. An example is shown in Fig. 3.4 (d). The best pair of paths (p, q) is taken from this new set of paths. In the example, the selected pair is highlighted in red.

The last step of SD-PCA will attempt to improve the solution by increasing the separation of the current pair of paths in proximity of the endpoints (s and t), so that the solution can more closely resemble the paths of Fig. 3.4 (a). To do so, SD-PCA generates new paths from this pair. This is done by keeping the inner part of the pair (which are the parts most distant from each other), and connecting it to the source through previously unused two-hop neighbors of the source, and to the destination through unused two-hop neighbors of the destination. An example of paths generated in this manner is shown in Fig. 3.4 (e). The inner part of the pair is highlighted in red. Note that these new paths can offer more separation near the endpoints. Finally, from this set, the pair of most disjoint paths is selected (Fig. 3.4 (f)).

3.4.2.2 SD-PCA algorithm

Now we explain in detail the full operation of SD-PCA. Because the algorithm uses shortest paths between arbitrary nodes, it first solves all-pairs shortest path problem by applying Dijkstra's algorithm (single-source shortest paths problem) on all nodes in the graph G. Thus, given every pair of nodes $(m, n) \in V \times V$, a list of shortest paths between them is stored. SD-PCA limits the number of paths stored between two nodes

to a maximum of $C \log V$, where C is a constant. As a way of accessing the shortest paths stored, SD-PCA uses the following function:

Definition 1. The shortestPaths(a, b, *one_path*) function returns a list of shortest paths between nodes a and b. If *one_path* is true the list will contain only one shortest path, otherwise it will contain all paths previously generated and stored by Dijkstra.

First we must explain the operation of the helper function pathGenerator.

Algorithm 1 Path generator helper function.

```
1: function pathGenerator(N, a, b, cond, postProcess, one_p, one_q) do
2:   P := {} // Empty set
3:   for n in N do
4:     for p in shortestPaths(a, n, one_p) do
5:       for q in shortestPaths(n, b, one_q) do
6:         if cond(p, q) then
7:           p := concatenate(p, q)
8:           if (hops(p) ≤ maxPLen) and ¬loops(p) then
9:             Insert p into P
10:          end if
11:          postProcess(p, n)
12:        end if
13:      end for
14:    end for
15:  end for
16:  return P
17: end function
```

The pathGenerator function (Algorithm 1) generates a set of paths in the following manner: for every node $n \in N$, one or all (depending on argument *one_p*) shortest paths from a to n are concatenated with one or all (depending on argument *one_q*) shortest paths from n to b if both paths satisfy *cond*. The concatenated path is stored if it has no loops and its length is less than the maximum allowed path length. A limit on path length is enforced to prevent SD-PCA from finding overly long paths, which can prove detrimental for routing. For each concatenated path, a post-processing function *postProcess* is called with the path and its generating node.

Definition 2. n is a *generating* node of path p if p is produced by pathGenerator from node n. Specifically, if p is the result of a concatenation of a path *to* n and a path *from* n.

Definition 3. genNodes(p) returns a data structure storing the list of generating nodes of path p.

The main body of SD-PCA is shown in Algorithm 2. In the case where the hop distance between s and t is less than or equal to 2, the best pair of paths is taken from all the shortest paths between s and t and the algorithm finishes (lines 1-2, 15). Otherwise, the main body executes (lines 3-14). In lines 4 and 5 the sets U_s and U_t are initialized, containing the 2-hop neighbors of s and t, respectively. These sets will keep track of the two-hop neighbors of the source and destination which are not used, i.e. not part of any of the paths generated in the first two phases of the algorithm.

The first phase of the algorithm involves generating a set of paths from the nodes at roughly the same distance to s and t, and choosing the pair of most disjoint paths from this set, as shown in Figs. 3.4 (b) and (c). This phase is executed in lines 6-7. The result of line 6 is that, for each node $v \in V$ at roughly the same distance to s and t (condition f_1) it stores a path p which is the concatenation of *one* shortest path from s to v and *one* shortest path from v to t. This is done by applying pathGenerator with $cond = \mathrm{f}_1$ and last two arguments (*one_p* and *one_q*) set to **true**. The post-processing function f_3 first calls f_2, which removes the two-hop neighbors of s and t that are part of the path from U_s and U_t, and then inserts the generating node into genNodes(p). Note that more than one node could generate the same path. From the initial set of paths, a best pair (p, q) is obtained (line 7).

Given the initial pair of spatially separate paths (p, q), the algorithm proceeds to the second phase, which corresponds to Fig. 3.4 (d). Lines 8 and 9 will generate close alternatives to p and q, respectively. In line 8, pathGenerator is applied using the generating nodes of p; it is now called with *one_p* and *one_q* set to **false**, in order to generate paths using *all* possible combinations of shortest paths. The same is done with q in line 9, and from the resulting sets of paths the best pair is chosen (line 10).

Finally, the algorithm executes the third phase, of Figs. 3.4 (e) and (f), in lines 11-13. The function separateNearEndpoints of Algorithm 3 takes a path and expands it, so that similar paths are generated from it, which go through unused two-hop neighbors of

Algorithm 2 SD-PCA: Find a pair of spatially disjoint paths between s and t in $G = (V, E)$.

```
1: if d_h(s, t) ≤ 2 then
2:    (p, q) := bestPair_a(shortestPaths(s, t, false))
3: else
4:    U_s := {n ∈ V : d_h(s, n) = 2}
5:    U_t := {n ∈ V : d_h(t, n) = 2}
6:    P := pathGenerator(V, s, t, f_1, f_3, true, true)
7:    (p, q) := bestPair_a(P)
8:    P_1 := pathGenerator(genNodes(p), s, t, f_T, f_2, false, false)
9:    P_2 := pathGenerator(genNodes(q), s, t, f_T, f_2, false, false)
10:   (p, q) := bestPair_b(P_1, P_2)
11:   P_1 := separateNearEndpoints(p)
12:   P_2 := separateNearEndpoints(q)
13:   (p, q) := bestPair_b(P_1, P_2)
14: end if
15: return (p, q)
16:
17: function f_1(p, q) do
18:    return max(|p|, |q|) / min(|p|, |q|) ≤ 1.25 or ||p| − |q|| ≤ 1
19: end function
20:
21: function f_2(p, n) do
22:    if (hops(p) ≥ 2) then
23:       U_s.remove(p_2)
24:       U_t.remove(p_{|p|−3})
25:    end if
26: end function
27:
28: function f_3(p, n) do
29:    f_2(p, n)
30:    genNodes(p).insert(n)
31: end function
32:
33: function f_T(p, q) do
34:    return true
35: end function
```

Algorithm 3 Derive additional paths from p that traverse unused nodes of U_s and U_t.

```
1: function separateNearEndpoints(p) do
2:   P := {p}
3:   if hops(p) ≥ 4 then
4:     δ_s := min(4, ⌈hops(p)/2⌉)
5:     δ_t := min(4, ⌊hops(p)/2⌋)
6:     P_1 := pathGenerator(U_s, s, p_{δ_s}, f_T, ∅, false, true)
7:     P_1.insert((p_0, ..., p_{δ_s}))
8:     P_2 := pathGenerator(U_t, p_{p_1 δ_t}, t, f_T, ∅, true, false)
9:     P_2.insert((p_{p_1 δ_t}, ..., p_{p_1}))
10:    for p_1 in P_1 do
11:      for p_2 in P_2 do
12:        u := concatenate(p_1, (p_{δ_s}, ..., p_{p_1 δ_t}), p_2)
13:        if (hops(u) ≤ maxPLen) and ¬loops(u) then
14:          Insert u into P
15:        end if
16:      end for
17:    end for
18:  end if
19:  return P
20: end function
```

s and t. Given a path p, separateNearEndpoints does the following: lines 4 and 5 choose the nodes which define the boundary of the inner part of p which will be preserved, shown in Fig. 3.4 (e) as the part marked in red. These nodes are p_{δ_s} and $p_{|p| - \delta_t}$. Next, paths are generated in line 6 connecting s to p_{δ_s}, passing through nodes in U_s. These paths are stored in P_1 and correspond with the paths of Fig. 3.4 (e) marked in green. In line 8 the same is done to connect $p_{|p| - \delta_t}$ to t through the nodes in U_t, and the resultant paths are stored in P_2 and correspond with the paths of Fig. 3.4 (e) marked in blue. Finally, paths are generated by concatenating all combinations of paths in P_1 (green paths), the inner part of p (red), and paths in P_2 (blue).

The main body of SD-PCA, in lines 11 and 12 applies separateNearEndpoints to the current pair of paths. From the new set of paths generated, the best pair is chosen in line 13, and the algorithm finishes.

3.4.3 Time complexity of SD-PCA

This subsection studies the run time of the algorithm in a worst case scenario.

Definition 4. Let d be the average distance between two nodes in G.

Proposition 1. *Solving the all-pairs shortest paths problem requires* $O(V^2 \log V + VE)$.

Proof. We can assume the graph is sparse (i.e. average node degree is low). Even in dense networks, is is possible to derive a sparse graph that maintains connectivity with all nodes (see sections 2.4.2.3 and 3.5.5 for more information). Therefore the Fibonacci heaps implementation of Dijkstra's algorithm can be used. Thus, SD-PCA requires $O(V \log V + E)$ for each of V instantiations of Dijkstra's algorithm. □

Proposition 2. *Functions* shortestPaths(a, b, one_path), $|p|$, hops(p) *and* $d_h(m, n)$ *run in* $O(1)$.

Proof. The lists of shortest paths between two nodes are stored in a data structure which we assume requires average constant time to access. The length of a path is stored with the path, and access is $O(1)$. The distance d_h between two nodes is the number of hops of a shortest path. □

Lemma 1. *The execution time of* $\text{bestPair}_\text{a}(P)$ *is* $O(P^2 d^2)$ *and the execution time of* $\text{bestPair}_\text{b}(P_1, P_2)$ *is* $O(P_1 P_2 d^2)$.

Proof. The execution time of node_dist(n, p) (equation 3.1) is $O(d)$. The distance d_h between two nodes is calculated for each node of p, i.e. an average of d times. Distance between two nodes is obtained in $O(1)$ from Proposition 2.

The execution time of PSD (equation 3.2) is $O(d^2)$: node_dist is calculated for each node in p, i.e. an average of d times.

$\Gamma(p, q)$ (equation 3.3) runs in $O(d^2)$, derived from the above.

$\text{bestPair}_\text{a}(P)$ (equation 3.5) requires calculating $\Gamma(p, q)$ P^2 times, therefore run time is $O(P^2 d^2)$.

$\text{bestPair}_\text{b}(P_1, P_2)$ (equation 3.6) requires calculating $\Gamma(p, q)$ $P_1 \times P_2$ times, therefore run time is $O(P_1 P_2 d^2)$.

□

Proposition 3. f_1, f_2, f_3 *and* f_T *from Algorithm 2 run in constant time.*

Proof. Functions $|p|$ and hops(p) run in $O(1)$ from Proposition 2. Access and modifying operations on data structures U_s, U_t and genNodes(p) is $O(1)$; the number of elements stored in these structures is very small, and we assume the use of data structures of average constant operations. □

Lemma 2. *The execution time of* separateNearEndpoints(p) *is* $O(log^2 V\, d)$. *The number of paths generated is* $O(log^2 V)$.

Proof. Lines 2-5 run in $O(1)$.

We can consider the size of U_s and U_t to be small constants, which will at most contain all the two-hop neighbors of s and t, respectively. The number of two-hop neighbors depends on the node degree of the graph, which we consider much smaller than V and independent of V.

Lines 6 and 8 execute in $O(\log V\, d)$. For line 6, because *one_p* is **false** and *one_q* is **true**, line 4 of Algorithm 1 will return at most $O(\log V)$ paths, and line 5 only one path. f_T is always **true**, so line 7 of Alg. 1 is executed $O(\log V)$ times, and the total running time is $O(\log V\, d)$, because a concatenation of paths is done in $O(d)$. The number of returned paths is $O(logV)$. The same reasoning applies to line 8.

Because the size of P_1 and P_2 is $O(logV)$, line 12, which runs in $O(d)$, is executed $O(log^2 V)$ times, so the total running time of the loop (lines 10-14) is $O(log^2 V\, d)$, which is the total running time of separateNearEndpoints(p). The maximum number of paths generated is $O(log^2 V)$. □

Theorem 1. *The execution time of SD-PCA is* $O(V^2 \log V + V E)$.

Proof. Prior to executing Algorithm 2, the all-pairs shortest path problem is solved, which requires $O(V^2 \log V + VE)$, from Proposition 1.

Given Algorithm 2:

Line 2 requires $O(\log^2 V d^2)$, derived from Proposition 2 and Lemma 1, because the maximum number of paths returned by shortestPaths is $O(\log V)$.

Lines 4 and 5 run in constant time. Building the sets involves obtaining the list of neighbors of each neighbor of s and t, which depends on node degree.

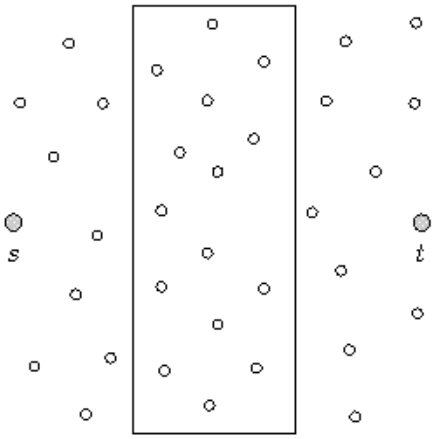


Figure 3.5: Nodes at the same or similar distance to source and destination. - The nodes enclosed in the rectangle represent nodes at roughly the same distance to s and t. The maximum height of the rectangle is the diameter of the graph.

The time to execute line 6 is $O(V + \text{Diameter}(G)d)$. Because both arguments *one_p* and *one_q* to pathGenerator are **true**, lines 4 and 5 of Algorithm 1 will only return one path. Thus, line 6 of Alg. 1 is executed V times. The number of nodes in V for which condition f_1 will be satisfied is $O(\text{Diameter}(G))$. These nodes will be located in an area as shown in Fig. 3.5. The nodes enclosed in the rectangle represent nodes at roughly the same distance to s and t. The maximum height of the rectangle (in hops) is the diameter of the graph. The maximum width in hops will be a small constant c, necessary to satisfy the condition of equidistance to s and t. The density of the graph is assumed to be low, this leads that the number of nodes in the rectangle is $O(\text{Diameter}(G))$. So line 7 of Alg. 1 is executed $O(\text{Diameter}(G))$ times, and therefore the total running time is $O(V + \text{Diameter}(G)d)$. The number of paths returned is $O(\text{Diameter}(G))$.

Line 7 executes in $O(\text{Diameter}(G)^2 d^2)$, from Lemma 1, because the number of paths returned in the above line is $O(\text{Diameter}(G))$.

Lines 8 and 9 execute in $O(\log^2 V d)$. First, the number of generating nodes of path

p will be a small constant c, equal or close to 1. Because both arguments *one_p* and *one_q* to pathGenerator are **false**, lines 4 and 5 of Algorithm 1 will return at most $O(\log V)$ paths. f_T is always **true**, so line 7 of Alg. 1 is executed $O(c \log^2 V)$ times, and the total running time is $O(\log^2 V d)$. The number of returned paths is $O(log^2 V)$. The same reasoning applies to line 9.

Line 10 executes in $O(\log^4 V d^2)$, from Lemma 1, because the number of paths returned in lines 8 and 9 is $O(log^2 V)$.

Lines 11 and 12 run in $O(log^2 V d)$, from Lemma 2.

Line 13 executes in $O(\log^4 V d^2)$, from Lemma 1, because the number of paths returned in lines 11 and 12 is $O(log^2 V)$ (from Lemma 2).

If we approximate a mesh network with a grid graph, we have that Diameter$(G) = O(\sqrt{V})$ and $d = O(\sqrt{V})$. Under this assumption, the time complexity of SD-PCA is dominated by the necessity to previously solve the all-pairs shortest path problem, which requires $O(V^2 \log V + VE)$.

□

3.5 Spatially Disjoint Multipath Routing protocol

This section develops the SDMR routing protocol for the discovery of spatially disjoint paths, explaining the procedures for neighbor discovery, topology discovery, path calculation using SD-PCA and route maintenance.

3.5.1 SDMR overview

SDMR is a reactive multipath protocol capable of finding multiple spatially distant paths between two nodes. For a source node S to calculate disjoint paths to a destination D, it first needs a graph of the connectivity of the network, in a similar way to link-state routing protocols. S produces this topology graph using connectivity information received from other network nodes. The source node then searches the graph for candidate paths between itself and the destination and chooses the set of most disjoint paths according to a distance metric.

The route discovery method is thus similar to other link-state routing protocols like OLSR, but the main difference stems from the fact that in OLSR all nodes periodically broadcast their connectivity information (link-state) to all nodes in the network, whereas in SDMR this information is only sent to a source node on demand. To

achieve this reactively, S *requests* connectivity information to the set of Multipoint Relays (MPR) in the network and to the destination by flooding a Topology Request (TREQ) message, and these nodes send their 1-hop neighbor set back to the source in Topology Reply (TREP) messages, along the reverse paths formed during TREQ propagation.

Data packets are routed along the paths using source routing.

3.5.2 Definitions

- G_n: Undirected topology graph of node n used to find and calculate disjoint paths.
- S: Source node.
- D: Destination node.
- TREQ: Topology Request message.
- TREP: Topology Reply message.
- $nbset_n$: Set of 1-hop neighbors of n.
- $nbset_n^m$: Last known state of $nbset_m$ stored in node n.
- Set of neighbor sets: Set of $(n, nbset_n)$ pairs.
- $nbset\_seqnum_n$: Neighbor set sequence number of n. Incremented each time a change occurs in its 1-hop neighbor set.
- $last\_nbset\_seqnum_n^d$: Value of $nbset\_seqnum_n$ when n last sent its 1-hop neighbor set to d.
- $parent_n^d$: Parent node of n on the spanning tree rooted at d.
- $pending\_trep\_timer_n^d$: Given a pending TREP from n to d, the expiration of this timer triggers the sending of the TREP.
- $pending\_trep\_children_n^d$: Given a pending TREP from n to d, this set contains the known children of n from which it will receive TREPs.

- $pending\_trep\_nbsets_n^d$: Given a pending TREP from n to d, this is a set of neighbor sets which n will append to the TREP sent to d.
- $pendingTopologyDiscovery_n$: True if n is currently doing a topology discovery, false otherwise.
- $pendingDestinations_n$: Given a pending topology discovery, this set contains the destinations for which n is currently waiting to obtain spatially disjoint paths.
- $pending\_treq\_timer_n^1$: Given a pending topology discovery, when this timer expires, n tries to calculate spatially disjoint paths to each destination in $pendingDestinations_n$. If it succeeds, the topology discovery finishes, otherwise n will wait until $pending\_treq\_timer_n^2$ expires.
- $pending\_treq\_timer_n^2$: Given a pending topology discovery, when this timer expires, n tries to calculate spatially disjoint paths to each destination in $pendingDestinations_n$. If it succeeds, the topology discovery finishes, otherwise either a new one commences or all packets to $pendingDestinations_n$ are dropped.

3.5.3 Neighbor detection

In SDMR nodes periodically send HELLO messages. The information exchanged allows nodes to discover their 1-hop and 2-hop neighbors. Based on this information nodes select their Multipoint Relays (MPR). This functionality is the same as in OLSR, refer to this protocol for details (see [20, 51] and section 2.4.2.3).

Whenever the 1-hop neighbor set of a node n changes, n increments $nbset\_seqnum_n$.

3.5.4 Routing packets at the source

SDMR is a reactive protocol and finds routes only when needed. When sending a packet, a source S follows the process shown in Figure 3.6. If the source already has valid spatially disjoint paths stored to the destination D, it uses the paths. Otherwise it must obtain new paths. A source considers stored paths valid as long as they don't break and are fresh. If a path currently used breaks due to link failure (e.g. caused by node mobility or failure), the route maintenance procedure is invoked (see section 3.5.7), to remove the broken links from the topology graph and delete affected paths. The topology graph stored in nodes is used to find paths to any destination, i.e. if the

topology graph is fresh the source can search the graph for spatially disjoint paths to any destination without the need for a new topology discovery, thus reducing overhead. This contrasts with other routing protocols like AODV (see section 2.4.3.1) which require a new route discovery for every destination. If the graph is fresh S searches the graph for spatially disjoint paths by applying the SD-PCA algorithm (see section 3.4). If the graph is not fresh or no suitable paths are found in the graph a new topology discovery must be made.

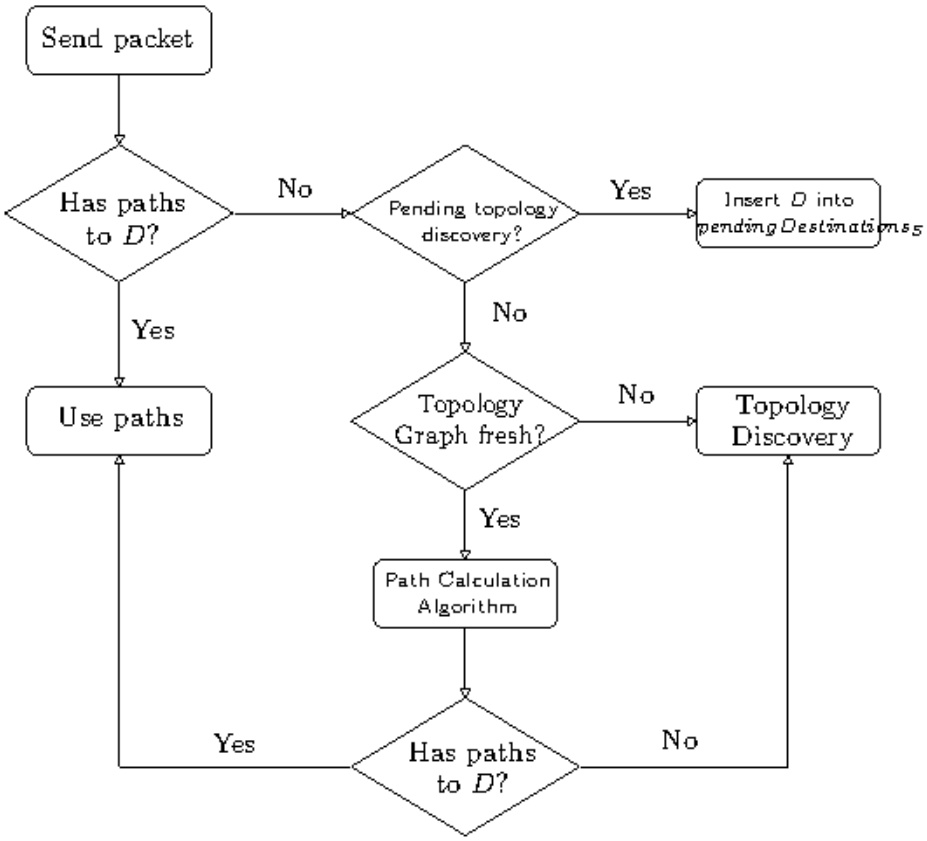


Figure 3.6: SDMR packet routing process at the source node. - This flowchart depicts the process followed by a source node when it needs to send a packet to a destination D.

If a node needs new paths to a destination when a topology discovery is currently in progress, the node inserts the destination into $pendingDestinations_S$ and waits for topology discovery completion (as stated, one topology discovery conveys relevant information to find paths to all destinations).

3.5.5 Topology discovery

In the absence of location information, a node with a local view of the network cannot make decisions which will effectively separate multiple paths which go from an arbitrary source to an arbitrary destination. In other words, to effectively obtain spatially distant paths (with potentially unbounded separation) a global view of the network is necessary. That is why the source node must obtain a topology graph.

A topology discovery attempts to obtain a graph of the connectivity of the network. It is important that the graph span all geographical regions of the network. With this graph, spatially disjoint paths can be found to any other node. To reduce overhead, only the MPRs will send their connectivity information to the source node. MPRs enable connectivity between all network nodes, and their use is specially suitable in large and dense networks. Because MPRs cover all geographical regions of the network, this doesn't prevent SDMR from finding any existing spatially disjoint paths.

A node starts a topology discovery by creating and broadcasting a TREQ message. $pending\_treq\_timer_n^2$ is set to wait for topology discovery completion. The TREQ will be flooded and MPRs will send TREPs back to the source. A topology discovery will conclude after a specified time (determined by SDMR) elapses, unlike other reactive routing protocols where the discovery concludes after receiving a reply from the destination. The reason is that receiving a reply or topology information from the pending destinations may not be sufficient to calculate spatially disjoint paths to them. It is important to receive link-state information from all geographic zones in the network, therefore even if the source receives replies containing information for all destinations, additional time is waited to attempt to receive TREPs from the whole network. Topology discovery resolution is explained later in section 3.5.5.3.

3.5.5.1 Topology Request propagation

A TREQ contains the following fields:

source_addr	dest_addr	broadcast_id	parent

The pair (*source_addr*, *broadcast_id*) uniquely identifies a TREQ. *broadcast_id* is incremented whenever S issues a new TREQ. When a node receives a TREQ from a

neighbor with a ($source\_addr$, $broadcast\_id$) pair not seen before, it records the neighbor as the parent node to reach S. The *parent* field of a TREQ stores the parent of the last node that forwarded the TREQ. The TREQ will be flooded by MPRs, forming reverse paths in the network to S, which results in a spanning tree rooted at S. The use of MPRs reduces the number of redundant retransmissions in the same region while at the same time allowing all nodes in the network to receive the TREQ, thus reducing overhead.

Algorithm 4 shows the process followed by nodes upon receiving a TREQ:

Algorithm 4 A node n receives a TREQ from neighbor m.

```
1: if n ≠ treq.dest_addr then // is not target
2:   if n has not heard (source_addr, broadcast_id) then
3:     Cache (source_addr, broadcast_id)
4:     parent_n^S := m
5:     if n is a Multipoint Relay of m then
6:       Set pending_trep_timer_n^S to δ_τ ms
7:       treq.parent := m
8:       Forward treq
9:     end if
10:  else
11:    if pending_trep_timer_n^S is set and treq.parent = n then
12:      Insert m into pending_trep_children_n^S
13:      if |pending_trep_children_n^S| = 1 then
14:        Set pending_trep_timer_n^S to Δ_τ ms // Wait for TREP from children
15:      end if
16:    end if
17:  end if
18: else // is target
19:  if n has not heard (source_addr, broadcast_id) then
20:    Cache (source_addr, broadcast_id)
21:    parent_n^S := m
22:    Send TREP to S
23:  end if
24: end if
```

When a node n receives the first copy of a TREQ from a neighbor m, it sets a

reverse pointer to S via m (lines 2-4 and 19-21) and forwards the message if it is a MPR of m (lines 5-8). When a node forwards a TREQ it records its parent node to reach S in the message (line 7). The flood of the TREQ by MPRs produces a spanning tree rooted at S.

Every MPR that receives a TREQ has to send a response to S. The response will contain its 1-hop neighbor set if it has changed since the last response sent to S. The neighbor set is carried in Topology Reply messages (TREP). To reduce overhead, only the leafs of the spanning tree should generate TREPs, and remaining nodes will aggregate the TREPs received from their children in one TREP and send it to their parent node. Initially, every MPR that receives a TREQ sets a timer to generate a TREP message (line 6) after δ_τ ms. When a node determines that it is not a leaf, it resets the timer to a higher value ($\Delta_\tau > \delta_\tau$) to wait for TREPs from its children (lines 11-14). A node knows that it is not a leaf when it receives a TREQ from a node that has chosen it as parent (line 11). D immediately sends a TREP when it receives a TREQ (line 22). By only sending the connectivity information of MPRs, TREP overhead and bandwidth is reduced.

3.5.5.2 Topology Reply propagation

A TREP contains the following fields:

source_addr	dest_addr	nbsets

The field nbsets is a set of neighbor sets, as defined in 3.5.2. When a node sends a TREP, it follows the steps of Algorithm 5. The response will contain its 1-hop neighbor set only if it has changed since the last response sent to S (lines 2-4). The rule of line 2 enables a node to determine if its neighbor set has changed. This is possible because a node increments $nbset\_seqnum_n$ with every change in its neighbor set, and records the sequence number when it last sent a response to S (line 4). Different sequence numbers don't guarantee that the neighbor set differs. For example, a node can temporarily lose some neighbors and recover them later. In this case, the $nbset\_seqnum_n$ prior to losing the neighbors and after recovering them is different but refers to the same neighbor set. However, in many cases this rule will avoid sending connectivity information which

hasn't changed. If a node has neighbor sets received from its children, it aggregates them in the TREP (line 8).

Algorithm 5 A node n sends a TREP to S.

1: trep := Create TREP
2: **if** $nbset\_seqnum_n > last\_nbset\_seqnum_n^S$ **then**
3: trep.nbsets := $\{(n, nbset_n)\}$
4: $last\_nbset\_seqnum_n^S := nbset\_seqnum_n$
5: **else**
6: trep.nbsets := $\{(n, \emptyset)\}$
7: **end if**
8: trep.nbsets := trep.nbsets $\cup$ $pending\_trep\_nbsets_n^S$
9: Send $trep$ via $parent_n^S$

Intermediate nodes in the tree can either generate or forward TREPs. When a node receives a TREP, Algorithm 6 is executed. If it receives a TREP before $pending\_trep\_timer_n^S$ expires, and it is still waiting for TREPs from its children, it stores the neighbor sets contained in the TREP in $pending\_trep\_nbsets_n^S$ and returns (lines 2-6). If it has received TREPs from every child, it cancels the timer, appends its 1-hop neighbor set to the TREP (if neighbor set has changed since last TREP to S) and all the neighbor sets in $pending\_trep\_nbsets_n^S$, and forwards the TREP via the parent (lines 7-18). If the node doesn't have a pending TREP to S it simply forwards the TREP to the parent. If the pending TREP timer expires it must send a new TREP (Algorithm 5).

When S receives a TREP it adds the contained connectivity information to G_S (lines 20-32). Note that if a node has included an empty set in the TREP, the source will use the last known neighbor set of that node. If S has received information for all destinations in $pendingDestinations_S$, i.e. if all destinations are in G_S, it sets $pending\_treq\_timer_n^1$ to attempt to complete the topology discovery before $pending\_treq\_timer_n^2$ expires.

3.5.5.3 Topology discovery resolution

A topology discovery will forcefully conclude if $pending\_treq\_timer_S^2$ expires. But it may also conclude earlier if $pending\_treq\_timer_S^1$ was previously set. This timer expires

Algorithm 6 A node n receives a TREP from neighbor m.

```
1: if trep.dest_addr ≠ n then
2:   if pending_trep_timer_n^S then
3:     Remove m from pending_trep_children_n^S
4:     if |pending_trep_children_n^S| > 0 then
5:       pending_trep_nbsets_n^S := pending_trep_nbsets_n^S ∪ trep.nbsets
6:       return
7:     else
8:       if nbset_seqnum_n > last_nbset_seqnum_n^S then
9:         trep.nbsets := trep.nbsets ∪ {(n, nbset_n)}
10:        last_nbset_seqnum_n^S := nbset_seqnum_n
11:      else
12:        trep.nbsets := trep.nbsets ∪ {(n, ∅)}
13:      end if
14:      trep.nbsets := trep.nbsets ∪ pending_trep_nbsets_n^S
15:      Cancel pending_trep_timer_n^S
16:    end if
17:  end if
18:  Forward trep via parent_n^S
19: else
20:  if pendingTopologyDiscovery_n then
21:    for (v, nbset_v) in trep.nbsets do
22:      if nbset_v ≠ ∅ then
23:        nbset_n^v := nbset_v
24:      end if
25:      Integrate nbset_n^v into G_n
26:    end for
27:    if not set pending_treq_timer_n^1 then
28:      if D ∈ G_S ∀D ∈ pendingDestinations_n then
29:        Set pending_treq_timer_n^1
30:      end if
31:    end if
32:  end if
33: end if
```

before $pending\_treq\_timer_S^2$ and is set when S receives topology information from all pending destinations. When $pending\_treq\_timer_S^1$ expires, SDMR tries to find spatially distant paths to all pending destinations in the current topology graph. If all pending requests are satisfied (i.e. valid spatially disjoint paths are found), the topology discovery is completed. Otherwise SDMR waits until $pending\_treq\_timer_S^2$ expires to attempt to complete all pending requests. This gives additional time for S to receive more TREPs which can provide necessary information to calculate disjoint paths to the pending destinations.

When $pending\_treq\_timer_S^2$ expires, SDMR tries to find spatially distant paths to all pending destinations, and if there are still pending requests, SDMR reissues a topology discovery if the maximum number of tries have not been attempted, otherwise it drops all buffered packets to the pending destinations.

3.5.6 Path calculation

After completing a topology discovery the source S has a fresh topology graph of the network which is used to find spatially disjoint paths to any destination. The Spatially Disjoint Path Calculation Algorithm (SD-PCA) detailed in section 3.4 returns a pair of spatially disjoint paths to a destination given a topology graph, which are used to route packets using source routing.

3.5.7 Route maintenance

Route maintenance refers to the mechanisms used by nodes when a link in an active path fails. The sources which are using the link as part of a path are informed, the link is removed from topology graphs and affected paths are deleted.

It is assumed that every node n has a data structure called *remoteActivePaths* which stores, for every source using n as an intermediate node of an active path, the next and previous hop in the path. The next hop leads to the destination, and the previous hop leads to the source. Note that nodes build this information locally without extra control overhead during the course of routing data packets (the source route header contains the previous and next hops).

When a link breaks, the node which detects link failure removes the link from its topology graph, and if the node currently has active paths using the link it deletes the paths. The node then obtains a list of sources which are affected by the link

failure (from *remoteActivePaths*). A Route Error (RERR) message contains the list of affected sources which have to be notified and the broken link. If there is only one source affected, the RERR can be sent by unicast. Note that a node knows the previous hop to reach a source from *remoteActivePaths*. If there are multiple sources affected, the RERR is sent by broadcast to its neighbors, so that those neighbors in paths leading to the sources will forward the message (this mechanism is also used, for example, in AODV).

When a node receives a RERR, it removes the broken link from its topology graph. Then it checks if it is one of the affected sources, in which case the broken link information is processed (by deleting the affected paths) and the source is removed from the RERR. If the RERR has been received by broadcast, the node will only resend the RERR if at least one of the sources in the RERR has an active path through this node. Every other source is removed from the RERR.

Finally, if after processing the RERR there are still sources remaining in the message, the node forwards the RERR, by broadcast if there are multiple sources or by unicast if there is only one source.

After deleting a path which was in use, the source follows the normal routing procedure shown in Fig. 3.6 to continue routing packets to the destination, i.e. if the topology graph is fresh it will attempt to find an alternative path in the graph. If the graph is not fresh or no paths are found, it will issue a new topology discovery.

3.6 SDMR protocol overhead study

This section analyzes the overhead of SDMR in terms of the number of control messages, and compares with OLSR. In both protocols, sources obtain the same topological information (the link state of other nodes in the network), meaning that OLSR can be modified to use the SD-PCA algorithm of section 3.4, and as such achieve the same functionality of SDMR. In OLSR all nodes obtain the complete topology from Topology Control (TC) messages, and therefore by applying SD-PCA a node can obtain spatially disjoint paths to any destination. As we will see, the main difference between SDMR and OLSR stems from the fact that the former is reactive (finds routes only when needed) while the latter is proactive (periodically exchanges control messages to maintain routes). It is therefore important to prove the conditions under which the

overhead of SDMR is better than OLSR. To this end, a theoretical analysis of overhead is carried out, deriving an analytical expression for comparison of both protocols.

Because the overhead of SDMR largely depends on the successful transmission and reception of TREQs broadcasted during flooding, the study considers both an ideal case and a worst case scenario.

The study excludes HELLO messages. The reason is that neighbor discovery is the same in both protocols, with all nodes periodically sending HELLO messages to their neighbors.

3.6.1 Best case scenario

3.6.1.1 OLSR

The number of control packets generated is:

$$\tau_O N R_{mpr} \quad (pkts/sec) \tag{3.7}$$

where τ_O is the rate of TC generation (per sec), N is the number of nodes in the network and R_{mpr} is the average number of forwards in a flood (the number of MPRs which forward TCs).

The reason behind equation 3.7 is that every $1/\tau_O$ seconds, all nodes (N) flood a TC. The TC is retransmitted R_{mpr} times (by MPRs).

3.6.1.2 SDMR

The number of control packets is:

$$N_S \tau_S (R_{mpr} + R_{mpr} \quad 1) \quad (pkts/sec) \tag{3.8}$$

where N_S is the number of sources using the SDMR protocol and τ_S is the average rate at which sources initiate new topology discoveries (per sec).

The reason behind equation 3.8 is that each source (N_S) starts a route discovery every $1/\tau_S$ seconds (on average). The route discovery implies a TREQ flood, which is retransmitted R_{mpr} times (by MPRs). This process forms a spanning tree of R_{mpr} nodes. The nodes which have retransmitted the TREQ reply with a TREP. In the best case, nodes only send one TREP and TREPs are aggregated in parent nodes in

the tree, meaning that only one TREP is sent per edge of the spanning tree. So the number of TREPs is R_{mpr} 1.

3.6.1.3 Comparison of SDMR and OLSR

To be able to compare both protocols, we have:

$$\alpha = \tau_O/\tau_S \tag{3.9}$$

The overhead of SDMR will be the same as OLSR when:

$$N_S\tau_S(R_{mpr} + R_{mpr} \quad 1) = \tau_O N R_{mpr} \tag{3.10}$$

$$N_S = \alpha\frac{N}{2\ \frac{1}{R_{mpr}}} \approx \alpha\frac{N}{2} \tag{3.11}$$

If for example the rate τ_S is half the rate τ_O ($\alpha = 2$) and $N_S = N$ the overhead of SDMR would be the same as OLSR. But because SDMR is reactive we can safely assume that $\tau_S \ll \tau_O$: SDMR finds routes only when needed, and when a node does a topology discovery the information gained can be used to calculate routes to multiple destinations, which decreases τ_S with respect to common reactive protocols.

3.6.2 Worst case scenario

For the worst case scenario, we assume there is a high chance that a broadcast message is not received.

3.6.2.1 OLSR

The expression for overhead is the same as in the best case. Note, however, that the rate of TC generation (τ_{O_2}) is now greater because the loss of broadcast messages (e.g. HELLOs) leads to topology changes which trigger TC floods. We assume that during a TC flood, all MPRs receive at least one copy of the flooded message, and therefore all MPRs retransmit the message.

$$\tau_{O_2} N R_{mpr} \quad (pkts/sec) \tag{3.12}$$

3.6.2.2 SDMR

$$N_S \tau_S (R_{mpr} + R_{mpr} D) \quad (pkts/sec) \tag{3.13}$$

Where D is the average distance from a node to the source (root) in the spanning tree. Note that τ_S is not affected by the loss of broadcast messages.

Like OLSR, we assume that during a flood all MPRs receive at least one copy of the flooded message (TREQ in this case), and therefore all MPRs retransmit the message. In this scenario, if in the worst case none of the nodes which have forwarded a TREQ (R_{mpr}) receives a TREQ from any of its children, the nodes will send their own TREP. Because this TREP must reach the source, the number of TREPs is $R_{mpr} D$.

3.6.2.3 Comparison of SDMR and OLSR

For comparison, we have:

$$\beta = \tau_{O_2} / \tau_S \tag{3.14}$$

Note that because $\tau_{O_2} > \tau_O \rightarrow \beta > \alpha$.

To compare SDMR and OLSR it is desirable to express D as a function of the number of nodes N in the network. D will largely depend on the network topology, location of the source and destination nodes, and the topology of the spanning tree formed during TREQ propagation. The network diameter can be taken as an upper bound of D. This is true if the spanning tree formed contains shortest paths from each node to the root. Even if the spanning tree formed is not the shortest paths spanning tree but an approximation, the diameter is still valid as an upper bound of the average distance of nodes to the root.

To calculate the diameter, this study considers a model of a mesh network where the graph is a square grid, with nodes connected as shown in Fig. 3.7, i.e. a node will have at most eight links. If $\sqrt{N}$ is not an integer, the remaining nodes ($N \quad \lfloor \sqrt{N} \rfloor^2$) are distributed randomly around the edge of the square, one hop from the edge. In this network the diameter is $\lceil \sqrt{N} \rceil \quad 1$.

In this network, we can conclude that the overhead of SDMR in the worst case will be less than OLSR if:

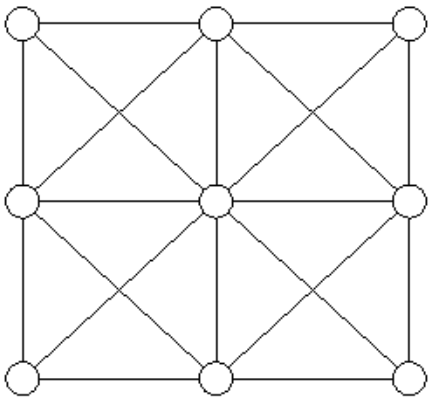


Figure 3.7: Nodes arranged in grid - Network model considered for studying SDMR overhead in a worst case scenario.

$$N_S \tau_S (R_{mpr} + R_{mpr} D) < \tau_{O_2} N R_{mpr} \tag{3.15}$$

$$\text{i.e. } N_S < \beta \frac{N}{1+D} \tag{3.16}$$

$$\text{Because } D < \lceil \sqrt{N} \rceil \quad 1: \tag{3.17}$$

$$N_S \leq \beta \frac{N}{\lceil \sqrt{N} \rceil} < \beta \sqrt{N} \tag{3.18}$$

3.7 SDMR performance evaluation

The goal of the following performance evaluation is to analyze the capability of SDMR of finding spatially disjoint paths. To this end, various tests are performed by simulation with *ns-2* [28], showing the nature and separation of paths found. To contrast the separation of paths found by SDMR with a well-known multipath protocol, the AOMDV protocol is also simulated, which is a reactive multipath protocol capable of finding link or node-disjoint paths (see section 2.4.4.4 for more information). In addition, simulations with its single-path variant (AODV) are carried out to provide additional insight, as it is a well-known reactive protocol (more details in section 2.4.3.1).

This study first measures the distance of paths found by SDMR, with the PSD metric (equation 3.2) and Euclidean distance, showing the spatial separation of the paths found. It then evaluates and compares the performance of SDMR against the other protocols. To demonstrate the effectiveness of spatial separation, it analyzes the

percentage of session packets every node in the network can intercept when routing with each protocol. Both SDMR and AOMDV distribute data packets along every available path to the destination, and AODV uses a unicast route. Results show that in SDMR there are substantially less nodes receiving more than 50% of the session. Finally, tests with node mobility are conducted.

3.7.1 Simulation environment

As a result of the work of this thesis, a simulation model of SDMR for *ns-2* has been developed. The AODV model of *ns-2* is based on [94]. The AOMDV model is based on [75]. Unless otherwise stated, scenarios simulated consist of static networks of 35 nodes placed randomly in a rectangular 1000 meter $\times$ 1000 meter area and 120 nodes placed randomly in 2000 m $\times$ 2000 m area. These topologies are connected and with a minimum distance of 160 m between nodes. The traffic pattern consists of CBR/UDP connections with one source and one or multiple destinations. Each simulation lasts 500 seconds, of which the first 50 seconds represent a warm-up period. The source commences transmission to the first destination after the first 50 seconds of simulation and stops transmitting 5 seconds before the simulation ends, to give time for packets on flight to reach their destinations. Traffic rate of each connection is 4 packets/s and size of data packets is 512 bytes. SDMR parameters are $\delta_\tau = 50$ ms and $\Delta_\tau = 500$ ms.

Confidence intervals are shown at the 95% level. Each point in graphs shown represents an average of ten runs with different topology.

3.7.2 Performance metrics

Protocol performance is studied using the following metrics: (i) *Path distance* - PSD distance between paths as defined in equation 3.2; (ii) *Euclidean path distance* - Euclidean distance between paths, obtained from eq. 3.2 by substituting d_h in eq. 3.1 for Euclidean distance d_E; (iii) *Data packet delivery ratio (PDR)* - calculated as the ratio of received data packets to those transmitted by the sources; (iv) *End-to-end delay* - this is the average end-to-end delay of all delivered data packets. Includes all buffering and queuing delays; (v) *Normalized Control Overhead (NCO)* - is the number of routing control packets transmitted per data packet correctly received at the destination.

3.7.3 Distance between paths found by SDMR

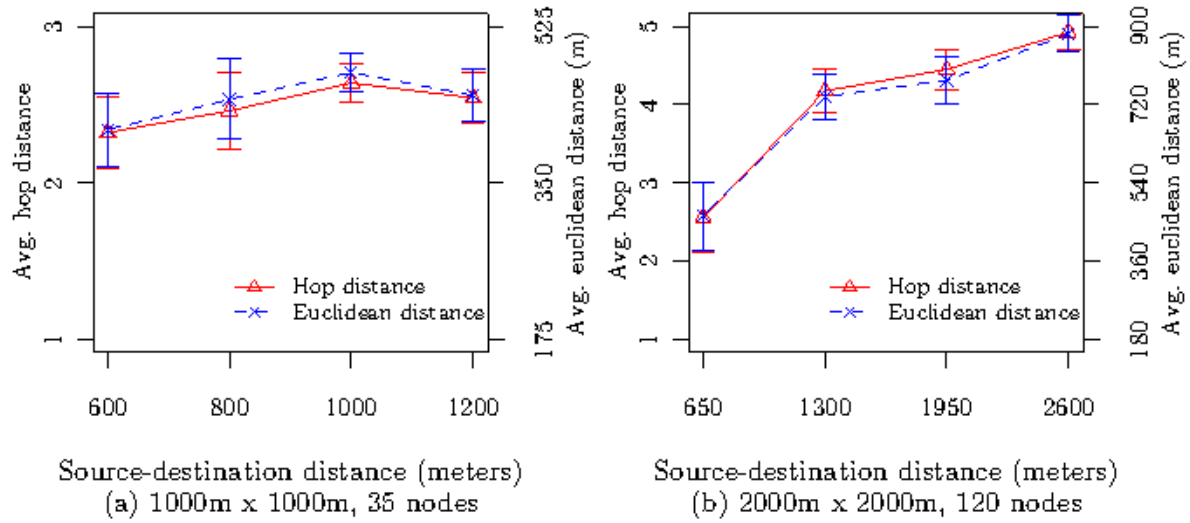


Figure 3.8: Distance between paths found by SDMR. - The left axis shows the distance in hops as calculated by the PSD metric, while the right axis shows the Euclidean distance of the paths, calculated from the PSD metric usign d_E instead of d_h.

In these tests there is one source-destination pair. In order to prevent SDMR from using the same set of paths found in the first topology discovery, a new topology discovery is forced every 50 seconds.

Figure 3.8 shows the results. The distance of paths is measured with the PSD metric (equation 3.2) and with Euclidean distance. The left axis shows PSD and the right axis shows Euclidean distance. Note that because Euclidean distance represents the average minimum Euclidean distance of nodes of one path to another path, the maximum average value it can reach in a square 1000 m $\times$ 1000 m area is roughly 500 m, i.e. although the maximum distance of a node to the nearest node of another path can be 1000 m, on average it is 500 m. And the maximum value in 2000 m $\times$ 2000 m scenarios is roughly 1000 m. As we can see in Fig. 3.8 (a) the Euclidean distance of paths found by SDMR is close to the maximum possible in these scenarios. We can also observe that PSD and Euclidean distance are highly correlated. In the scenarios depicted in Fig. 3.8 (b), the average Euclidean distance increases with the distance between source and destination, up to a maximum of 900 m, which is also close to the

maximum possible in these scenarios. The high correlation between hop distance and Euclidean distance can also be seen.

3.7.4 Comparison of protocol performance

The following tests compare the performance between SDMR, AODV and AOMDV. In a first set of tests, there is one source-destination pair and a new route discovery is forced every 50 seconds for each protocol. This allows to compare the average cost of route discovery between protocols. Because these are static scenarios with only one data connection, the achieved PDR of all protocols is very high (median value is 1). The average transmission delay is also low due to the existence of only one flow. The median value for AODV and AOMDV is 33.94 and 33.46 ms, respectively. In SDMR it is slightly higher (44.72 ms), because route discoveries take longer. Also, in routing along spatially disjoint paths it is expected that paths will be longer than the unicast shortest path case.

The main difference in performance between SDMR and the other protocols is route discovery overhead. Figures 3.9 (a) and (b) show the results. The figure shows the total number of RREQs (TREQ in SDMR) and RREPs (TREP) sent during the simulation period, averaged over multiple scenarios. All protocols perform the same number of route discoveries. RREQ overhead is less in SDMR because only MPRs forward RREQs, however, RREP overhead is greater. This is because a subset of MPRs (ideally only the leafs of the spanning tree generated during topology discovery) generate TREPs, which are propagated to the source. Additionally, in some cases more nodes other than the leafs will generate TREPs, due to not receiving a TREQ from a child node (due to wireless propagation and collisions).

In Fig. 3.9 (a), the fact that RREQ overhead is less in SDMR results in all three protocols having roughly the same overhead. In the scenarios of 2000 x 2000 m (Fig. 3.9 (b)), the difference between SDMR and the other protocols is higher, due to the fact that many more nodes are generating TREPs.

To illustrate the particular scenario where one topology discovery in SDMR can serve to find paths to any destination, a second set of tests perform simulations with one source and multiple destinations. As explained in section 3.5, this feature is not present in common reactive protocols like AODV which require a new route discovery for every destination. In these tests, source and destinations are randomly located and the

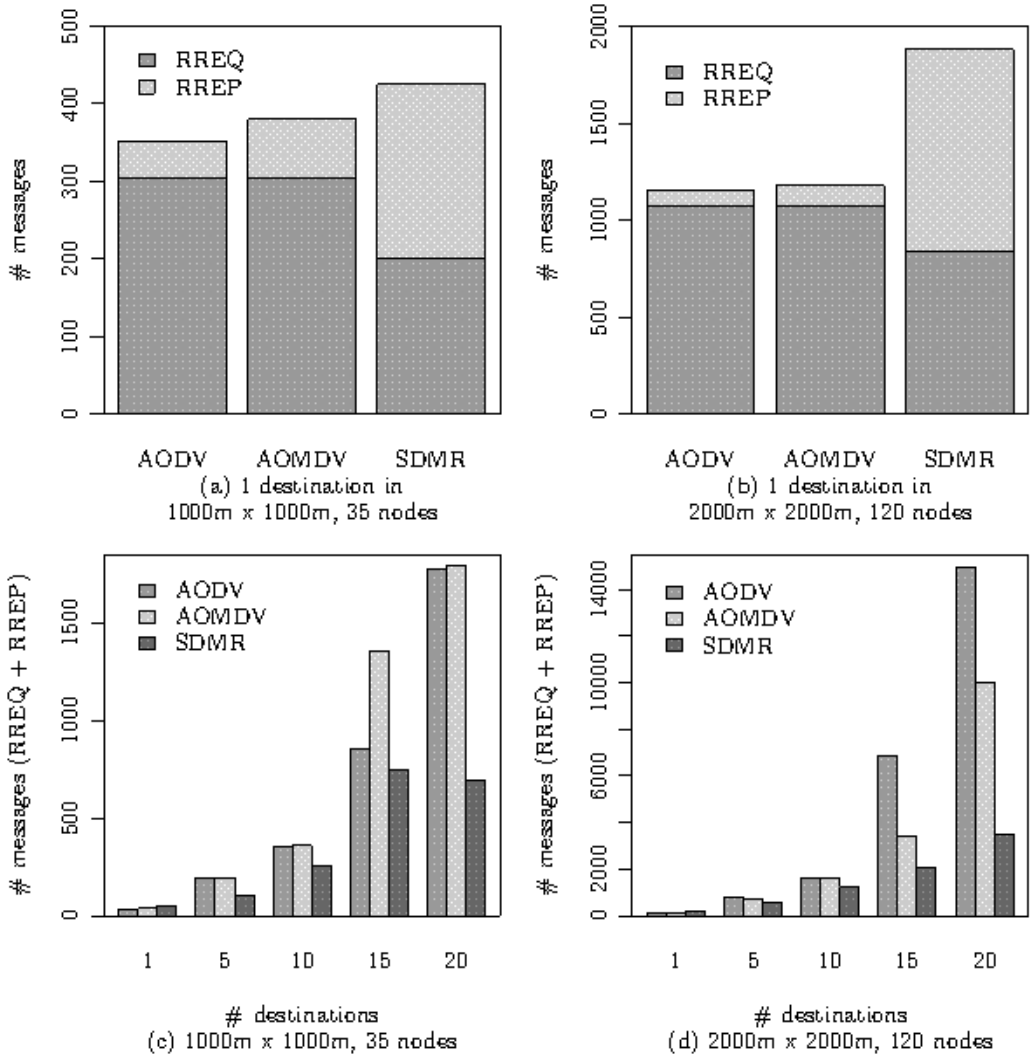


Figure 3.9: Performance comparison between SDMR, AODV and AOMDV - The figure shows route discovery overhead of the protocols with one source and one destination (figures (a) and (b)), and one source and multiple destinations (figures (c) and (d)).

number of destinations varies between 1 and 20. The source commences transmission to the first destination after the first 50 seconds of simulation, and subsequently every 10 seconds for each new destination. All transmissions end 5 seconds before the simulation finishes. Route discoveries are performed only when required by the protocol. The topology graph timeout of SDMR is set to 20 seconds for two reasons: (i) the topology is static, and (ii) even though it is static, a timeout is desired because RERRs received at the source due to congestion change the view of the network.

Figs. 3.9 (c) and (d) show the results. With one destination the overhead of SDMR is higher. But with more destinations it is lower. There are two main reasons for this. First, because in these tests one topology discovery of SDMR can serve to find paths to 3 new destinations. Because a source starts transmitting to a new destination every 10 seconds, in the course of 20 seconds the source starts 3 transmissions. A topology graph expires 20 seconds after completing a topology discovery, so one topology discovery can be used for 3 destinations. Second, in the presence of congestion, routes start to break and the source must recalculate routes. With AODV, a new route discovery must be made. AOMDV has the possibility of using alternate routes (which is why its overhead is lower in the scenarios of Fig. 3.9 (d) with more than 10 destinations). In the case of SDMR, if the graph is fresh, it can recalculate alternate routes without reissuing a topology discovery.

3.7.5 Analysis of session interception

Another set of tests analyzes the interception of data packets by network nodes with all protocols. Besides studying the effectiveness of spatial separation, the goal is to verify if SDMR is suitable for use in scenarios where it is not desired for potentially dangerous nodes in the network to have access to all packets in a session.

In these tests there is one source-destination pair and a new route discovery is forced every 50 seconds, to prevent protocols from using the same set of paths found in the first route discovery. Both SDMR and AOMDV route packets along available paths to destination in round robin fashion. AODV routes along one path. Note that paths can change in all protocols throughout the simulation, since each route discovery can result in different routes. The tests count the number of data packets every node receives, including nodes along routes used.

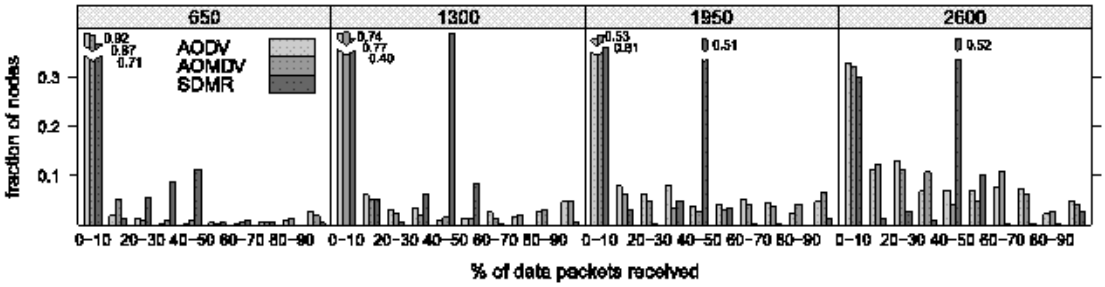


Figure 3.10: Session interception results - Histograms of data packet reception with varying source-destination distance. 2000 m × 2000 m, 120 nodes.

Figure 3.10 shows the fraction of nodes receiving a percentage of session packets. A number of conclusions can be extracted from this figure. In SDMR substantially less nodes receive more than 50% of the session, as compared to AODV and AOMDV. The effect of routing along two distant paths in SDMR is that there are much more nodes receiving between 40-50% of the session: the session is evenly distributed along two paths, and the nodes along these paths and their neighbors can receive these packets. In AODV, since packets are routed along one path, less nodes will be able to overhear data packets, but in SDMR less nodes are able to overhear more than 50% of the packets. AODV and AOMDV behave similarly. This is because in practice AOMDV rarely finds more than one path in the scenarios tested, and paths found are not spatially disjoint (AOMDV only aims to find the shortest link-disjoint paths, which in practice are close to each other).

An important conclusion that can be drawn from the figure is that the advantage of using SDMR to prevent nodes from having access to the entire session decreases with the distance between source and destination. This is logical, since the shorter the route, the less the number of network nodes that can overhear packets. If the distance is too short (e.g. 650 m as shown) it might even prove detrimental to attempt to route along spatially disjoint paths, since this will give more nodes access to data as opposed to routing along one path.

3.7.6 Performance under mobility

SDMR can adapt to node mobility and topological changes. Node movement leads to link breaks and route failures. As explained in section 3.5.7, the route maintenance procedure enables the protocol to update its path information and topology graph when

links in use break, and to search the topology graph for alternate routes or, if none are found, to issue a new topology discovery.

This section analyzes the performance of SDMR with node mobility. For these tests, random mobility scenarios have been generated with one source-destination pair, in topologies of square area 1000×1000 m with a node density of 100 nodes. In contrast to the previous tests performed on topologies of 1000×1000 m where the density was low (35 nodes), the node density in these topologies is high, in order to guarantee graph connectivity in the presence of node movement. The maximum speed of nodes varies in $\{0, 5, 10, 15, 20\}$ m/s. A mobility scenario consists of a random initial location of nodes (including source and destination) and their movement throughout the simulation. Ten random mobility scenarios have been generated for each of the possible node speeds, totaling 50 scenarios. The movement pattern of nodes has been generated with BonnMotion [1] using the GaussMarkov model. Each point shown in the following figures is the average over ten different scenarios.

The tests compare the packet delivery ratio and overhead of SDMR and AODV, as both are reactive protocols. Figure 3.11 shows the results. As can be seen in the figure, the performance of both protocols is similar, specially in regard to protocol overhead. Increasing route failures lead to an increasing number of route discoveries needed to adapt to topological changes, and thus higher overhead. Even though the RREP overhead of SDMR is notably higher, the use of MPRs in these dense topologies contributes to SDMR having a much lower RREQ overhead, which evens the overhead of both protocols. The delivery ratio of SDMR drops faster with mobility, which suggests that AODV has better adaptation capability, specially when topological changes are more frequent. This is expected, as route discoveries take longer in SDMR.

Finally, Figs. 3.11 (c) and (d) show the performance of SDMR in terms of the quality of paths found when there is mobility. Fig. 3.11 (c) shows the average Euclidean distance between paths found per route discovery. This is the distance measured at the instant the paths are found. In previous tests, it was shown how the distance between paths increases with source-destination distance. In these tests, the source-destination distance is random and constantly changing, as well as the network topology, and thus the distance between paths found will change accordingly. It is important to note that the distance achieved is not affected by node speed. Fig. 3.11 (d) illustrates the fact

that not all route discoveries result in finding two paths. Because the topology changes, in rare cases the source only finds one available route to the destination.

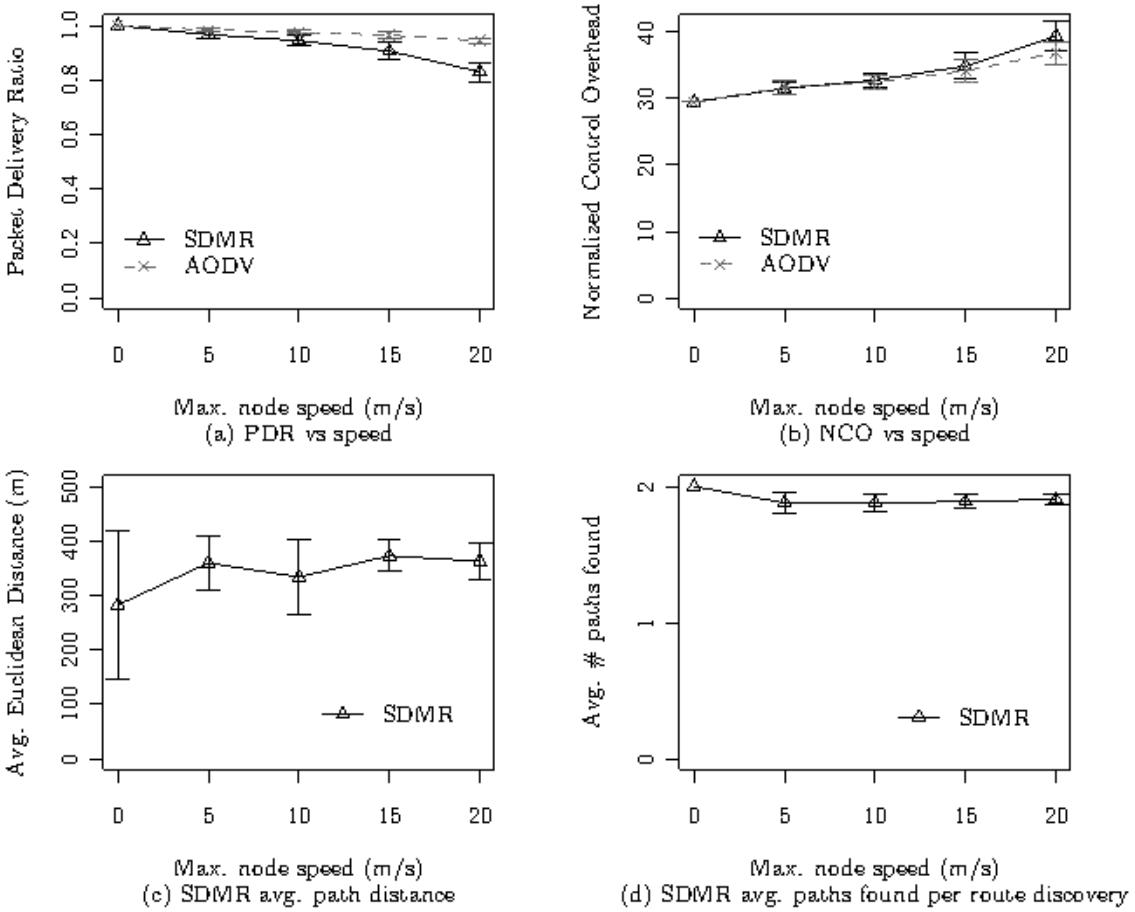


Figure 3.11: Performance of SDMR with mobility - The tests are conducted in topologies in an area of 1000 m × 1000 m with 100 nodes. There is one source-destination pair. All nodes (including the source and destination) move randomly.

3.8 Conclusions

The discovery and use of spatially separated paths can provide various benefits in MWNs, including the possibility of avoiding inter-flow interference, increased fault-tolerance, and increased security. Being able to measure distance between paths and discovering spatially disjoint routes is necessary to achieve these benefits.

This chapter has developed a metric to measure distance of a path to a set of nodes in the network. This metric, called Path Spatial Distance (PSD), does not require the location information of nodes. It only requires connectivity information, and is based on hop distance. On the tests conducted, the metric has been shown to be congruent with Euclidean distance between nodes, and as such is suitable to measure spatial separation.

The Spatially Disjoint Path Calculation algorithm (SD-PCA) has also been proposed. Given the topology graph of the network and using the PSD metric, this heuristic algorithm can find spatially separated between two nodes. The time complexity analysis of SD-PCA shows that routes can be calculated efficiently. In addition, the tests performed with the SDMR protocol prove that the algorithm can approximate paths with maximum separation.

The Spatially Disjoint Multipath Routing (SDMR) protocol developed uses the above distance metric and algorithm. SDMR is an on-demand multipath protocol based on source routing, capable of finding spatially disjoint paths in one route discovery, without the use of location information.

This chapter has studied both the capability of the protocol of finding spatially disjoint paths as well as its performance. The overhead of SDMR has been studied analytically and compared with OLSR. Simulation results have shown that the protocol can discover paths with considerable separation, and the PSD metric can correlate highly with Euclidean distance. The distance between paths found by SDMR can be potentially unbounded, and in the scenarios tested path separation has been close to the maximum achievable. In a 2000 x 2000 m area, we have seen separation of more than four hops and 700 m in most cases. In another set of tests comparing SDMR with AODV and AOMDV, it has been shown that distributing traffic with SDMR along separate paths, less network nodes are likely to receive more than 50% of the session. This means that SDMR is better suited for applications where it is not desired that all packets traverse the same region, e.g. secure communications, or better reliability avoiding regional failures.

The next chapter presents a load-balancing solution for single-channel WMNs with multiple gateways. The goal of the mechanism is to balance load between gateways. To effectively balance load while avoiding harmful inter-flow interference, the solution will use the PSD metric to measure the cost of paths.

Chapter 4

GWLB: Gateway load-balancing in single-channel WMNs

This chapter addresses the problem of load-balancing in the gateway access scenario in single channel Wireless Mesh Networks. In this scenario, gateway selection (whereby the traffic of a node or flow is assigned to be served by a particular gateway) proves to be an effective way of balancing load in the network. The gateway selection problem seeks to balance load between gateways while avoiding harmful interference in the network. The problem is studied in detail and a practical gateway load-balancing solution is proposed and evaluated.

4.1 Introduction and motivation

Wireless Mesh Networks (WMNs) have recently attracted much attention. One of the main reasons is that they provide a cost-effective way of deploying a wide-area network and offer services such as Internet connectivity. Gateway nodes are a key component of WMNs. In many applications of WMNs most traffic will be directed to/from gateways. When the network uses multiple gateways, one important consideration is the strategy employed to associate nodes with a particular gateway, i.e. through which gateway does a node send/receive traffic? We refer to this as the gateway selection problem, and the set of nodes served by a gateway as its *domain*.

In section 2.3 it was discussed that, as a general principle, maintaining traffic locality in multi-hop wireless networks is desirable to sustain adequate network throughput.

In a WMN, this translates into nodes using their nearest gateway to access external networks. However, it was also explained that this is only necessary if all nodes are active simultaneously and their traffic demands are uniform. In practice, not all nodes are active at the same time and, due to the small scale of the network, instantaneous demands can vary frequently and unpredictably.

The selection of gateways at a particular instant influences the traffic distribution and load pattern inside the WMN. It is important to balance load between gateways to avoid overutilized and underutilized regions. Load imbalance can easily occur due to a number of factors, such as heterogeneous traffic demands, time-varying traffic and uneven number of nodes served by gateways. This can lead to inefficient use of network capacity, throughput degradation and unfairness between flows in different domains.

On the other hand, arbitrary load-balancing can hurt performance. In a wireless multi-hop network, interference (namely inter-flow and intra-flow interference) has a major role in network performance. Association of nodes to gateways must be chosen carefully, to avoid flow interactions and to not severely disrupt traffic locality. Solving the problem therefore requires knowledge of the WMN to ensure high performance.

In the previous chapter the PSD metric was developed to measure the spatial separation of a path to a set of nodes. This metric can be used in single-channel networks to perform load-balancing, by finding paths that avoid inter-flow interference. In this chapter, the PSD metric will be used during gateway load-balancing to evaluate the cost of paths and avoid using those that prove harmful.

In addition, because traffic is dynamic and can vary frequently and unpredictably, gateway selection must be able to adapt to the current traffic conditions.

There are two approaches to solve the problem: centralized and distributed. The distributed approach generally involves each router choosing the best gateway based on a routing protocol and metric used in the WMN[1]. To balance traffic, the routing metric must reflect current load. Examples of metrics which directly or indirectly take load into account include ETX [22], ETT [27] and MIC [122]. These metrics were discussed in chapter 2, where it was also explained (in section 2.4.6), that the use of load-sensitive metrics can lead to network instability in many routing protocols. As a result, distributed approaches must deal with several issues. One is gateway

[1] When using this strategy, there must be a mechanism to ensure that traffic from the Internet to a mesh router enters the network through the gateway chosen by the router (e.g. as in [102]).

flapping, where a node continually changes its gateway selection. This occurs, for example, when multiple nodes discover an underutilized gateway at the same time, and switch simultaneously to this gateway, overloading it. Convergence time is also an issue because the time required to converge to a stable gateway selection may be longer than the time between traffic variations. Another issue is the overhead required to broadcast up-to-date load information through the network. For these reasons, in a distributed approach merely relying on a load-aware routing metric is not sufficient. In the next section we examine existing centralized and distributed approaches.

This chapter studies the benefits of centralized gateway selection, where the problem is solved outside the WMN. It develops an online algorithm called Gateway Load-Balancing (GWLB) [33, 36, 38] which, given the current network conditions, efficiently selects gateways for every node or flow. It is assumed that most traffic is directed to/from the Internet. As such, there is no added overhead involved to determine current network load (this information is collected by gateways as traffic passes through them). The solution can be recalculated periodically with a very high frequency to ensure responsiveness. Because it is calculated centrally, it will not suffer from convergence issues. Furthermore, gateway flapping does not occur, and a node can be prevented from switching gateways until a specified time elapses. To prevent harmful load-balancing due to severe interference, the potential interference generated by gateway selection is taken into account based on knowledge of the network topology, using the PSD metric (defined in section 3.3.3). Finally, the solution balances traffic at the TCP flow level, thus improving the throughput and fairness of flows. For these reasons, the proposed solution is suitable for implementation in practical and dynamic WMN scenarios.

The problem studied in this chapter concerns only a node's gateway selection, but not the actual choice of routes used *inside* the WMN. As explained in the introduction of this dissertation, in single-channel networks there is little to no benefit in attempting to balance the traffic assigned to a gateway inside the network, because all of its flows ultimately contend for access in the region surrounding the gateway. In other words, the choice of routes used to reach the chosen gateway is not a determining factor. In the context of multi-channel networks, it is possible to balance the traffic served by a gateway inside the WMN by way of routing and channel assignment. This problem will be studied in chapter 5.

The rest of the chapter is organized as follows. Section 4.2 reviews related work. Section 4.3 describes the network model, defines and formulates the Gateway Load-Balancing problem. Section 4.4 describes the Gateway selection algorithm. Section 4.5 explains the GWLB protocol. In section 4.6 the performance obtained by GWLB is studied against the optimal selection found by solving a MINLP formulation of the problem. Section 4.7 shows and analyzes protocol performance based on simulations with *ns-3*. Finally, section 4.8 concludes the chapter.

4.2 Related work

4.2.1 Centralized gateway selection

Many works [8, 13, 103, 114] study the general problem of centralized routing in WMNs, which involves calculating routes between every active source-destination pair. These solutions can be applied to the gateway selection problem, by assuming the existence of a "virtual" node (representing the Internet) connected to every gateway. This node is the source and destination of all traffic from/to the Internet. The routes calculated by these solutions will therefore determine the gateway used by nodes. Most centralized routing approaches have the following important drawbacks:

- Assume knowledge of an offline traffic matrix, which specifies the demand between all source-destination pairs [8, 103], or allocate rate for every source [13, 114].
- Use throughput models of the network that assume that the capacity of the network (links and collision domains) is known. To achieve this, they rely on simplifying assumptions: interference-free graph model [13], synchronized time-slotted access [8], binary interference models and throughput estimation [103, 114].
- Calculate splittable *flow*, where the demand between a source and destination is split among multiple paths [8, 114].

There are several limitations in the above assumptions when designing a solution for practical scenarios:

Traffic matrices are frequently used for traffic engineering in ISP networks, where capacity is above utilization. In this work, however, it is assumed that the traffic matrix

is unknown and, furthermore, this work contends that it cannot be known in WMNs for two main reasons: (a) because the capacity of a WMN is limited, users will tend to use all available capacity. This means that routing affects resulting demands, and those demands don't necessarily reflect user requirements. As soon as routing changes, demands can change; (b) traffic is dynamic and can constantly change.

Furthermore, this work assumes that network capacity is very difficult to predict for various reasons, including dynamic wireless medium, interference and MAC inefficiencies. For this reason, it is not possible to accurately predict throughput to guarantee that a certain rate allocation is feasible.

Finally, the proposed solution seeks unsplittable flow, where individual flows cannot be split. The problem is closely related to the single-source unsplittable flow problem, which is NP-hard [60].

Tokito et al. [116] propose a centralized gateway selection algorithm with the goal of balancing the load served by gateways. It assumes the current demand of each node is known, but does not specify how this information is obtained, and does not take contention inside the WMN into account when selecting gateways.

We remark that solutions such as the above that require the demand of nodes to be known assume that their demand will remain the same after a change in gateway selection, which is not true when traffic is constituted by TCP flows.

4.2.2 Distributed gateway selection

Raniwala et al. propose the *Hyacinth* architecture for multi-radio multi-channel WMNs in [102]. The routing protocol dynamically maintains trees rooted at each gateway, by periodic exchange of load information in the network. The tree to which a node joins determines its gateway selection. The metrics proposed are based on estimating residual capacity and are not suitable for balancing elastic traffic. Simulation results for a sample scenario show a convergence time of two minutes. Mhatre et al. [79] propose a distributed algorithm to form a delay-optimal routing forest, where a tree is rooted at each gateway. The algorithm, however, is not load-aware, and instead assumes that all nodes and links in the forest will be active simultaneously.

Similar to [116], authors in [47, 120] propose approaches with the goal of evening the load served by gateways. These solutions assume a specific demand for each node

and thus are not designed for TCP traffic. They don't take contention into account, and convergence time is not studied.

Other distributed proposals are aimed at switching nodes to alternate gateways when a gateway becomes congested. These include the protocols WCETT-LB [73] and AODV-ST [98], which extend existing routing solutions. These can suffer from gateway flapping. Another example is the work by Nandiraju et al. [87], in which congested gateways notify nodes which should switch to another domain. This solution doesn't take into account the effects of interference when switching nodes and can also suffer from gateway flapping. Maurina et al. [78] attempt to improve the proposal in [87] by more consciously choosing nodes to switch between domains (i.e. nodes with greater traffic and farther from the gateway). One drawback of these proposals is that they don't balance load in the absence of congestion. With TCP, domains may not be congested but flow unfairness between domains can occur.

4.2.3 Multi-gateway association

Other solutions [47, 50, 66] have proposed associating nodes with multiple gateways simultaneously. For these solutions it can be argued that, by arbitrarily breaking the locality of traffic in the network, they greatly increase contention and can perform worse than single nearest gateway association. For example, in [33] we showed how this occurs with the FP protocol from [47], in which nodes send to all gateways in proportion to gateway capacity.

4.2.4 Contributions

The solution designed in this chapter meets the following goals:

- Efficient traffic-aware centralized gateway selection. Because the solution is calculated centrally it does not suffer from convergence issues and gateway flapping is easily avoided. Furthermore, the solution is calculated frequently with a period of a few seconds to adapt to varying traffic conditions.
- Does not introduce any overhead to determine current traffic conditions.
- Takes into account the potential interference generated by gateway selection to avoid harmful load-balancing, using the PSD metric.

- Balances traffic at the TCP flow level, improving the performance and fairness of flows.

4.3 The Gateway Load-Balancing problem

4.3.1 Network and traffic model

This work considers WMNs which comprise of wireless static or quasi-static mesh *routers*, also called *nodes*. These nodes form a wireless multi-hop network. Mesh *clients*, also called *users*, connect to the mesh routers. A subset of nodes, referred to as *gateways*, are directly connected to a fixed infrastructure, which we will assume is the Internet for the rest of this work. Each router has one radio interface with an omni-directional antenna (e.g. 802.11a/b/g) for communication with other routers, using a single common channel. Communication between nodes and users is done via a separate interface and channel (wired or wireless). An example WMN was shown in Figure 1.1.

This work assumes no accurate knowledge of the capacity and interference model. This is difficult to obtain in practice in a WMN and can be subject to frequent change.

Although intra-WMN communication is possible, it is assumed that most of the traffic will be received from the Internet. Users are randomly located in the network. A user accesses the Internet through one or more links leading from its router to a gateway. To model Internet traffic realistically, this work assumes that all traffic entering the WMN is elastic, and that in any instant a user can initiate a connection to the Internet generating a download flow of any number of bytes. This is safe to assume, because the majority of Internet traffic today uses TCP. Because users are randomly located and can generate connections at any time, network conditions can constantly change.

4.3.2 Gateway Load-balancing overview

An Internet flow is a set of TCP packets exchanged between two endpoints, one in the Internet and one in the WMN. The network operator is responsible for choosing the exact properties that define an endpoint[1].

[1] In this work an endpoint is a pair (address,port).

Given a set of Internet flows, the goal is to find the best gateway for each flow, to optimize their performance (flow throughput and fairness). We call the router in the WMN participating in a flow a *sink*. In general, flows are created after a user in the WMN connects to a server on the Internet.

It is assumed that a routing protocol is executed *inside* the WMN, which establishes routes and performs routing between every pair of nodes in the WMN, including the gateways. Note that this protocol is *not* responsible for gateway selection, but will however route traffic between the chosen gateways and routers. This work does not impose any such protocol or routing metric.

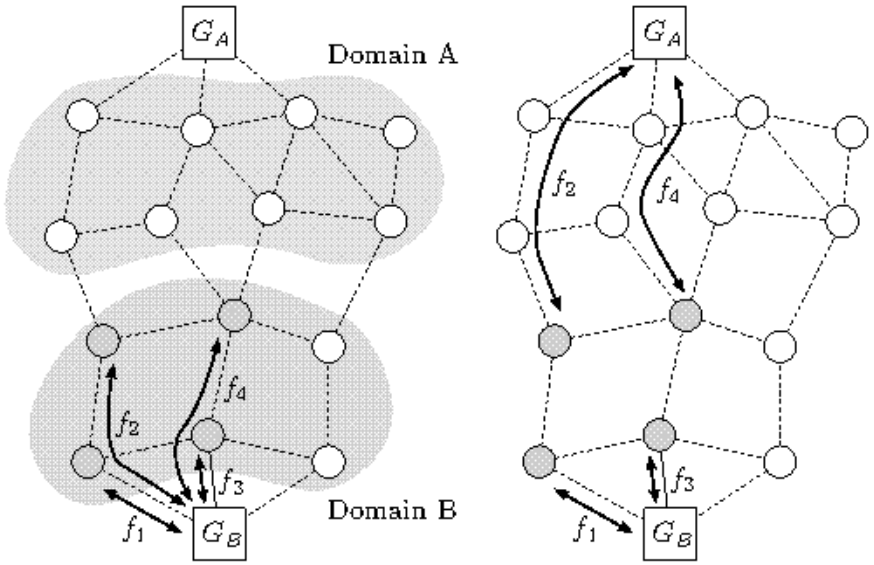
The problem requires that, at any one time, the traffic belonging to a flow be served by only one gateway. This is necessary to simplify routing and to reduce the probability of out of order delivery of packets and Round-trip time variations. This relates the problem to the single-source unsplittable flow problem, which is known to be NP-hard [60, 61].

Because routes inside the WMN are already given, the Gateway Load-balancing problem consists in choosing the serving gateway for each flow. Let $\mathbb{G}$ be the set of gateways in the WMN, $\mathbb{F}$ the set of current flows and $\mathbb{S}$ the set of sinks. Let $G = |\mathbb{G}|$, $F = |\mathbb{F}|$ and $S = |\mathbb{S}|$. A gateway assignment is a function $\alpha_f : \mathbb{F} \mapsto \mathbb{G}$. We denote $\alpha_f(f_i) = g$ if flow f_i is assigned to gateway g. The actual path leading from the gateway g to the flow's sink d is given by $P_{g \to d}$, which denotes the path used by the WMN routing protocol to connect nodes g and d.

Definition 5. A gateway's native domain is the set of nodes closest to the gateway, per the WMN routing protocol metric.

Fig. 4.1 illustrates a simple gateway load-balancing example. This example assumes the use of hop-count as routing metric and shortest path routing inside the WMN. In this case, the topology is unbalanced, i.e. the size of each native domain is different. An unbalanced topology can contribute to load imbalance, however, most frequently, traffic patterns are responsible for load imbalance (as is the case in this example where the most loaded domain is the smallest one). The colored nodes represent active sinks, each participating in one flow. The *Nearest Gateway* (NGW) solution, which uses native domains, can easily lead to load imbalance, as illustrated in Fig. 4.1 (a). In this situation, all flows are routed through G_3, which means that all the load is concentrated

on the region surrounding this gateway. However, the load can be balanced adequately by routing flows f_2 and f_4 through G_A (Fig.4.1 (b)), thus utilizing the capacity of domain A and consequently increasing the bandwidth share of all flows. The example also illustrates that minimizing imbalance usually comes at the expense of increased path cost (longer paths and consequently increased interference). Flows f_2 and f_4 are routed through longer paths in Fig. 4.1 (b). Also note that it would not be desirable to route flows f_1 and f_3 through gateway G_A, because this would create very long paths which would generate high interference.



(a) Without load-balancing (b) With load-balancing

Figure 4.1: Simple gateway load-balancing example - There are two gateways, G_A and G_B, with their corresponding native domains A and B. Colored nodes are participating in one flow (flows labeled f_1 to $f4$). Arrows indicate paths followed by flows.

The NGW solution can be considered a minimum path cost solution but can lead to high imbalance. In contrast, long paths have a high penalty in single-channel multi-hop wireless networks, where the interference factor plays a major role. The capacity of a chain of nodes and the locality of traffic are key factors influencing network capacity [43, 70]. Long paths and increased contention can severely impact performance. It is therefore important to limit path cost, and a trade-off is involved to optimize both load balance and path cost. How this optimization is performed will be explained in detail later.

The approach to gateway load-balancing in this work is to perform gateway selection for Internet flows in a centralized manner. The main reason is that a centralized controller located in one of the gateways or outside the WMN can easily perform load-balancing, providing many benefits in this scenario as explained below.

The information needed by the controller for gateway selection is the current set of flows $\mathbb{F}$ and the current network topology and routes. Gateways collect information on flows at the same time as traffic passes through them and inmediately send this information to the controller (a system such as Netflow [18] or IPFIX [19] can be used for this purpose). Controller and gateway communication is done through the wired network. Network connectivity and routes is assumed to be obtained from the WMN routing protocol (standard link-state protocols like OLSR provide this information). With the information of the current set of flows, the controller executes a fast gateway selection algorithm, calculating α_f and informs the gateways of the new association. Download traffic of a flow will be injected into the WMN by gateways based on this association. The gateway used by download traffic will be recorded in download packets, and the sink will therefore know which gateway to send upload traffic of the flow to.

This approach has the following benefits. Given that, at any instant, the controller knows the *current* set of flows and given the low complexity of the gateway selection algorithm, it is possible to periodically recalculate α_f with a very high frequency (the evaluations in this chapter have used an adaptation frequency as high as once per second). Therefore the controller can react quickly to the changing network conditions (new flows entering the network and flows that have ended). Because the solution is calculated centrally, there are no non-convergence problems which can arise in distributed implementations. That is, given the current state, the algorithm converges to an association α_f, and this solution remains the same until the next calculation. In addition, gateway oscillations can be easily avoided by locking the gateway association of a given flow for a specified time. Finally, the proposed solution does not introduce any control overhead in the WMN besides the routing protocol overhead.

Further details on the GWLB architecture and protocol for controller-gateway coordination are given in section 4.5. The following sections focus on the gateway selection algorithm which is executed periodically at the controller. Table 4.1 lists the symbols used throughout the text to explain the algorithm.

Table 4.1: GWLB symbols.

$\mathbb{G}$	Set of gateways in the network
$\mathbb{F}$	Set of Internet flows
$\mathbb{S}$	Set of sinks in the network
S_u	Set of unlocked sinks
α_f	Per-flow gateway assignment function: $\alpha_f(f_i) = g$ if flow f_i assigned to gateway g
α_s	Per-sink gateway assignment function: $\alpha_s(s_i) = g$ if sink s_i assigned to gateway g
$sink(f_i)$	Endpoint of flow f_i in the WMN
ND_g	Native domain of gateway g
$P_{m \to n}$	Path used by the WMN routing protocol to connect nodes m and n
P_θ	Maximum accepted path cost
$load(s)$	Number of flows of sink s
$load(g)$	Number of flows assigned to gateway g
VG_s	Set of gateways which can be used to reach sink s
VG	$\sum_{s \in S_u} \lvert VG_s \rvert$

4.3.3 Measuring load

This chapter considers dynamic traffic based on TCP. Flows are elastic and do not have a specific demand. Rate control and bandwidth sharing are performed by TCP. In other words, flows have the property of adapting to the available capacity and sharing bandwidth between them.

Taking the above into account, the load of a gateway g is measured as the number of flows served by the gateway, i.e. $load(g) = |\{f_i \mid \alpha_f(f_i) = g\}| \, \forall f_i \in \mathbb{F}$. Imbalance is the variance of the number of flows served by gateways. If gateways serve a similar number of flows, we can expect flows to better share all the network capacity, improving both average flow throughput and fairness. The evaluations in sections 4.6 and 4.7 show that this is a suitable measure to balance load with fairness in mind, better than metrics employed by previous works. The reason is that other metrics don't consider the adaptive nature of TCP flows. For example, when using TCP, congestion is usually temporary due to congestion control, and is not a reliable load indicator. In addition, flows don't have a specific demand but rather tend to use all available capacity (specially

true in a WMN where capacity is limited). This means that a low number of flows can produce the same demand as a large number of flows, and therefore the demand is not a reliable indicator to balance load.

Knowledge of flows is collected by gateways as traffic passes through them, requiring no overhead in the WMN.

4.3.4 Measuring path cost

The WMN routing protocol establishes paths between gateways and sinks. These paths are known by the controller. The controller performs its own estimate of the cost of these paths. This estimate is used when calculating gateway selection, in order to avoid using paths that can cause harmful interference. Note that this cost is unrelated to the cost of routes measured by the routing protocol. This chapter will always refer to the cost of paths as calculated by the controller. We now explain how it is measured.

The controller needs the topology graph and routes used by the routing protocol to connect gateways and sinks. This information is assumed to be obtained from the routing protocol. Note that link-state protocols such as OLSR already provide this information, with no additional overhead other than the default routing overhead (more details on how this information is obtained are given in section 4.5). Given the topology graph, the shortest distance (in hops) between all pairs of nodes is known. This can be calculated by Dijkstra's algorithm and need not be recalculated unless the topology changes.

The path cost metric is designed to meet the following goals:

First of all, it must only depend on the topology. This means that the cost of a particular path only needs to be recalculated if the topology changes.

Second, it is necessary that the metric account for increased network interference when choosing a serving gateway for a sink different from its nearest gateway.

For these reasons, the cost of a path from a gateway g to a sink s is measured as the expected interference of the path to nodes which are not likely to be served by g. That is, we desire the path to minimally interfere with nodes not served by the gateway. The set of nodes likely to be served by a gateway is defined as the sinks closest to it (per the WMN routing metric). Note that this measure also implicitly accounts for path length (longer paths will have increasingly higher cost because they will be farther away from the gateway and closer to nodes of other domains).

A cost metric that meets the above goals can be designed based on the PSD distance metric defined in chapter 3 (equation 3.2). The cost of a path is defined as follows:

$$pcost(P_{g \to s}) = \quad \text{PSD}(P_{g \to s}, \mathbb{S} \quad \text{ND}_g) \tag{4.1}$$

That is, the cost of a path from gateway g to sink s is the negative PSD distance of the path to nodes which are not in the native domain of g. The less distance to nodes which are not in the native domain of g, the higher the cost. The maximum path cost is 0.

In the previous chapter it was shown that the PSD metric is suitable for measuring spatial separation. Increased separation decreases the probability of interference.

4.3.5 Gateway selection problem

The problem consists in finding a flow-gateway association α_f that balances the load of gateways and limits the path cost inside the WMN, consequently increasing flow throughput and fairness. To that end, this work proposes solving the following multi-objective problem:

$$\mathbf{P_{GS}} : \min_{\alpha_f}[\max_{g \in \mathbb{G}}\{load(g)\}, \sum_{f_i \in \mathbb{F}} pcost(P_{\alpha_f(f_i) \to sink(f_i)})] \tag{4.2}$$

The first objective seeks to minimize the maximum number of flows served by a gateway. The second objective seeks to minimize the cost of paths in order to avoid interference in the network.

By minimizing the utilization of a gateway, the capacity of the gateway domain can be shared among fewer flows, increasing average flow throughput. Because load is balanced across collision domains, fairness is improved (e.g. by avoiding cases where there are regions with many contending flows and regions with few). Aggregate throughput can also improve by routing through otherwise unused gateways.

Two modes of association are supported: (a) per-sink association, where all flows of the same sink use the same gateway, and (b) per-flow association where no such restriction exists (i.e. flows of the same sink can arrive through different gateways). The advantage of per-flow association is that it can achieve better load balancing (although this benefit may not be attainable in single-channel WMNs as will be explained later),

while the advantage of per-sink association is that it only needs to store sink-gateway associations and gateway selection is performed faster.

The key to solving $\mathbf{P_{GS}}$ is determining the optimal trade-off between the objectives, establishing bounds on what is considered a valid path. This trade-off can vary from network to network, and depend on its current conditions, and as such can appear rather difficult to determine in the absence of an accurate network model.

The next section develops a method to solve the problem and shows how with basic knowledge and understanding of the network topology and traffic, and using an approximation algorithm, high-performance gateway selection can be efficiently calculated. Sections 4.6 and 4.7 will evaluate this.

4.4 Gateway selection procedure

The gateway selection algorithm (GSA) is executed centrally in the controller. To solve $\mathbf{P_{GS}}$, an upper bound is imposed on the cost of valid paths, i.e. paths with cost higher than this bound are not considered valid. This determines the set of paths which can be used to reach a particular sink, and this in turn translates into the set of gateways which can serve the sink. The procedure then focuses on minimizing the first objective, which constitutes a load-balancing problem equivalent to the Restricted Scheduling problem [31], which in turn is an instance of Scheduling in Unrelated Parallel Processors [49]. These problems are NP-hard, and are closely related to the single-source unsplittable flow problem (restricted scheduling can be expressed as an instance of it [60]). In the restricted scheduling problem, there are n tasks and m machines. A task can only be assigned to a subset of machines. The objective is to assign tasks to machines such that the makespan is minimized. In this case, the gateways are the machines, and the flows (or alternatively sinks) are the tasks. The time to process a task is one (per-flow association) or the number of flows of the sink (per-sink association).

Because the problem is NP-hard, this work proposes a heuristic algorithm to quickly approximate an optimal solution. A fast algorithm is essential in order to be able to calculate flow-gateway associations periodically with a high frequency.

4.4.1 Generating candidate paths

The valid paths (gateways) to reach a sink s are given by:

$$VG_s = \{g \in \mathbb{G} : pcost(P_{g \to s}) \leq P_\theta\} \tag{4.3}$$

This list is ordered in ascending order of cost, and updated when the cost of a path changes. The cost of a path is recalculated if the path to the sink changes, or the topology changes.

Possible values of the threshold P_θ are in the $[-X, 0]$ range, where $-X$ is the highest cost of a path in the NGW solution. This guarantees that every sink has at least one valid gateway. A higher threshold increases the number of alternative paths to reach a sink (and number of balancing options), but higher path costs will decrease performance. Finding a good value for the threshold requires some knowledge of the network characteristics, mainly, the average interference range (spatial reuse), and hop distance correlation with Euclidean distance (which essentially depends on the average link distance). A lower interference range permits a higher path cost, while a very high interference range may even prevent exploiting gateway load-balancing. The value of P_θ should be set close to the average interference range.

4.4.2 Gateway selection algorithm

GSA (Algorithm 7) is a fast greedy algorithm to solve $\mathbf{P_{GS}}$, which focuses on evening the load of gateways, without incurring in high path costs.

As mentioned, the load-balancing problem matches the restricted scheduling problem, which is NP-hard. To our knowledge, the best approximation algorithm to solve this particular problem was recently proposed by Gairing et al. [31]. However, because the system developed in this chapter has a requirement for high responsiveness, this work proposes using a faster algorithm which achieves near optimal results on average (see appendix A). The proposed algorithm is an adaptation of the List Scheduling algorithm commonly used in task scheduling problems. A special order is chosen for the sinks which, for this particular instance of the problem, has proven to offer better results than similar existing heuristics (see appendix A for more details).

This section only shows and explains the per-sink association variant, which is the one used in the simulation tests. The per-flow variant is very similar, only eliminating

the restriction that all flows of a sink use the same gateway. Although per-flow association achieves better balance at a similar path cost, the performance results observed in a single-channel WMN are almost identical to the per-sink algorithm. This is due to increased contention in the network and around the sinks.

GSA iterates over the list of currently unlocked sinks[2], in each iteration choosing the best gateway for the selected sink. The fitness of the solution depends highly on the order of sinks. A heuristic is used to order the sinks, determined by the comparison function of Algorithm 8.

Algorithm 7 GSA per-sink selection algorithm.

1: $S_u := \{s_i \in \mathbb{S} : \neg locked(s_i) \wedge load(s_i) > 0\}$
2: $\alpha_s(s_i) \leftarrow \emptyset \quad \forall s_i \in S_u$
3: Sort S_u // Comparison sort based on Algorithm 8
4: **for** $s_i \in S_u$ **do**
5: $\quad \alpha_s(s_i) \leftarrow \arg\min_{g \in VG_{s_i}} load(g)$
6: $\quad load(\alpha_s(s_i)) \leftarrow load(\alpha_s(s_i)) + load(s_i)$
7: **end for**

Algorithm 8 Sink comparison function used for sorting.

1: **function** less_than(s,t) **do**
2: $\quad$ **if** $(|VG_s| = 1)$ and $(|VG_t| > 1)$ **then**
3: $\quad\quad$ **return true**
4: $\quad$ **end if**
5: $\quad$ **if** $(|VG_s| > 1)$ and $(|VG_t| = 1)$ **then**
6: $\quad\quad$ **return false**
7: $\quad$ **end if**
8: $\quad$ **if** $\frac{load(s)}{VG_s} > \frac{load(t)}{VG_t}$ **then**
9: $\quad\quad$ **return true**
10: $\quad$ **else**
11: $\quad\quad$ **return false**
12: $\quad$ **end if**
13: **end function**

First we explain the greedy algorithm. In line 1, the list of unlocked sinks is initial-

[2]Unlocked sinks are sinks with no locked flows. When a flow is allocated to a gateway, GWLB locks the flow for a specified time (see section 4.5).

ized, containing sinks currently unlocked (with no locked flows). Because these sinks are now free to be routed through any gateway, their current gateway association is cleared (line 2). The list of unlocked sinks is ordered in line 3. For each unlocked sink, GSA chooses the current least loaded gateway (less number of flows) as its current gateway (lines 4-6). Note that if there is more than one least loaded gateway, the one with lowest cost to reach is chosen, because VG_s is ordered in ascending order of path cost. This contributes to generating paths with lower cost.

The comparison function of Algorithm 8 is a less-than operator which determines the order of sinks. Sinks with only one valid gateway are selected first to be assigned (lines 2-7). For all other cases (lines 8-12), a heuristic similar to Longest Processing Time (LPT) first is used. The LPT heuristic specifies that jobs with more load go first (in this case sinks with more flows). For this particular problem, however, we find that it is also important that sinks with more alternatives be picked last, because they offer more possibilities of balancing load, and these possibilities should not be used up too early. Both factors are combined in the decision (line 8).

Theorem 2. *The time complexity of* GSA *in the worst case is* $O(S+F+|S_u|\log|S_u|+VG)$.

Proof. Line 1 requires traversing all sinks and all flows in the worst case: $O(S+F)$.

In line 3, the list of unlocked sinks is sorted by a comparison sort, requiring $O(|S_u|\log|S_u|)$.

The loop of lines 4-7: Line 5 requires traversing the valid gateways of the sink. The loop runs in $O(|S_u|\frac{VG}{S_u}+|S_u|)=O(VG+|S_u|)=O(VG)$.

□

4.5 Gateway Load-balancing architecture and protocol

This section contains details on the GWLB architecture and protocol. This mainly refers to the process by which the controller coordinates with gateways to obtain the necessary network state, communicates the flow-gateway association to gateways and how they apply the solution.

The controller can be co-located with any one of the gateways or be an independent device. Controller and gateways communicate through the wired network. In case of

failure a new controller is elected from the remaining gateways. A simple election protocol can be used, e.g. choose the gateway with lowest ID (e.g. IP address). Gateways collect network state and send it to the controller. Based on the current state, the controller calculates the serving gateway of each flow. The gateway selection algorithm (explained in section 4.4) executes periodically in the controller every G_T seconds (in the evaluations $G_T = 1$). Each time the flow-gateway association α_f is calculated, the controller sends α_f to gateways in order to enforce the solution.

4.5.1 Routing download traffic

Download traffic is routed based on the current association α_f, i.e. a gateway is responsible for injecting the traffic of its assigned flows inside the WMN:

- If a gateway g receives traffic of flow f_i and the flow is *not* assigned to g, i.e. $\alpha_f(f_i) \neq g$, the gateway forwards the traffic to its serving gateway $\alpha_f(f_i)$ (directly or via a tunnel through the wired network if the gateways are in different subnetworks).
- If the traffic belongs to one of its served flows, i.e. $\alpha_f(f_i) = g$, it routes it through the WMN using the configured routing protocol. The gateway records its address in the packets' header.

4.5.2 Routing upload traffic

When a sink has to send upload traffic of a flow, it sends it to the gateway last used by download traffic of the flow (this information is recorded in download packets). If the gateway is currently unknown, it sends it to the nearest gateway (per the WMN routing protocol metric).

4.5.3 Network state monitoring

To calculate a flow-gateway association α_f, the controller needs the following information:

- WMN topology graph.
- Routes from every gateway to every sink: $P_{g \to s}\ \forall g \in \mathbb{G}, \forall s \in \mathbb{S}$.

- Knowledge of active flows $\mathbb{F}$.

Topological information and routes are obtained by gateways from the WMN routing protocol and this information is sent to the controller. The frequency used by the routing protocol to send this information is independent of G_T.

TCP traffic is classified into flows. Traffic classification is performed by the gateways when traffic passes through them, and this information is sent to the controller. A system such as IPFIX [19] can be used for this. Note that these systems employ two basic mechanisms to determine if a flow has finished: flow aging (no packets have been observed for a specified time) and detection of TCP session termination messages.

The controller will have knowledge of all flows but, obviously, flows with a duration shorter than G_T are not guaranteed to be taken into account by the GSA. This does not pose a problem when G_T is very small, because the flows which are not taken into account will be very small-sized flows, which have negligible repercussions on the imbalance of network traffic.

4.5.4 Gateway maintenance

The first time a controller is elected, and every time a new gateway appears or disappears, the controller obtains the current network state, recalculates α_f and informs all gateways. This is done to react to the presence of gateways in the network, adapting to the gateways present at any time. If a gateway fails, its flows will be re-routed through the remaining gateways. If the controller shuts down or fails, a new one is elected.

4.5.5 Load-balanced gateway selection

The load-balancing algorithm executes every G_T seconds, choosing the serving gateway of every flow. The algorithm was explained in section 4.4.

When a serving gateway is chosen for a flow, the flow is locked for $Lock_t$ seconds. The algorithm will not re-route locked flows. A sink is considered locked if one of its flows is locked.

4.5.6 Benefits of GWLB strategy

There are important benefits to the proposed strategy:

- The only required overhead is the normal routing protocol overhead, and the recording of the gateway address in download packets. Flow classification and book-keeping is done at the controller and gateways. Distributed protocols require periodic advertisement of load (e.g. gateway load) through the whole network. Frequent advertisements can introduce too much overhead, degrading performance, while a low frequency will limit adaptation to changing conditions.

- Low-complexity load-balancing algorithm which can be executed periodically with a high frequency, adapting promptly to changing conditions.

- Prevents gateway flapping and non-convergent behavior, because the solution is calculated centrally. Gateway flapping can occur when multiple nodes discover an underutilized gateway at the same time. The nodes then switch simultaneously to this gateway, overloading it. In a distributed protocol (e.g. Hyacinth), a mechanism is needed to prevent this situation. However, this mechanism normally results in arbitrary decisions as to which nodes switch and which don't, leading to suboptimal solutions.

- Frequent gateway oscillation of a given flow is also prevented by locking a flow for a specified time when a gateway is chosen.

4.6 Analysis of GSA routes

Gateway selection calculated by GSA determines the set of routes which will be used to transfer Internet traffic (we call these GSA routes). This section studies how these routes approximate routes found by optimally solving a rate allocation and routing problem which assumes perfect knowledge of the capacity and interference model.

4.6.1 Gateway Load-balancing MINLP formulation

This formulation involves routing and rate allocation in order to optimize a throughput and fairness criteria.

Let $\mathbb{F}$ be the set of flows, and r_i the rate of flow i (we assume that upload traffic is negligible and optimization involves the download direction). Let $F = |\mathbb{F}|$ and $G = |\mathbb{G}|$. A reasonable objective for WMNs is proportional fairness of flows, which can

be achieved by maximizing the aggregate utility of flows, when the utility of a flow is given by $\ln(r_i)$ [59].

A flow can be routed through G different paths, corresponding to the paths from each gateway to the sink. A solution must choose the serving gateway of each flow or, in other words, the route the flow uses. To represent this, the formulation defines G flows (called *routed flows*) for every original flow, each served by a different gateway. The number of routed flows is thus $F \times G$. Let t_{ij} be the routed flow of flow i assigned to gateway j, with rate x_{ij}. The formulation assigns a binary integer variable b_{ij} to each routed flow, indicating whether it is active or not. Only one routed flow of a flow can be active: $\sum_{j=0}^{G} b_{ij} = 1, \forall i \in \mathbb{F}$.

We thus have $r_i = \sum_{j=0}^{G} b_{ij} x_{ij}$. The network and interference model determines the remaining problem constraints. A model which has been frequently used defines contention regions of the network as maximal cliques of the conflict graph [121]. As explained in chapter 2, in the conflict graph, each vertex is a link of the connectivity graph, and there is a link between vertices when they contend. Interference is determined by the Protocol Model (see [43] and section 2.2.3.1).

A solution is given by the vector $\boldsymbol{x} = (b_{ij} x_{ij}),\ \forall i \in \mathbb{F}, \forall j \in \mathbb{G}$. The clique-flow matrix [121] $\mathbf{R} = \{R_{qt}\}$ represents the number of links the routed flow t is using in clique q. Let the vector $\mathbf{C} = (C_q)$ be the vector of achievable channel capacities in each of the cliques. Given the above considerations, the problem can be formulated as:

$$\mathbf{P_{opt}}: \text{maximize} \quad \sum_{i=0}^{F} \ln\left(\sum_{j=0}^{G} b_{ij} x_{ij}\right) \tag{4.4}$$

subject to:

$$\mathbf{R}\boldsymbol{x} \leq \mathbf{C} \tag{4.5}$$

$$\sum_{j=0}^{G} b_{ij} = 1 \qquad \forall i \in \mathbb{F} \tag{4.6}$$

$$b_{aj} \quad b_{bj} = 0 \qquad \forall (a,b) \in \mathbb{F} \times \mathbb{F} : sink(a) = sink(b)$$
$$\forall j \in \mathbb{G} \tag{4.7}$$

Constraint 4.7 is optional, used to enforce per-sink routing. The constraint forces every pair of flows of the same sink to have the same value of binary variables, thus

guaranteeing that they are routed through the same gateway.

4.6.2 Routes comparison of $\mathbf{P_{GS}}$ and $\mathbf{P_{opt}}$

To study how GSA routes can approximate routes found by $\mathbf{P_{opt}}$ the following methodology is used:

- Solve random traffic scenarios in a WMN topology with $\mathbf{P_{opt}}$. Given a set of flows, this obtains the optimal routes and rate allocation which maximize the aggregate utility of the flows.
- Solve the same scenarios with GSA (Algorithm 7). This determines the GSA routes for the given flows. To evaluate the quality of these routes and compare with $\mathbf{P_{opt}}$, the maximum aggregate utility of flows which can be achieved using these routes is determined, and compared with the aggregate utility calculated by $\mathbf{P_{opt}}$. To this end, the paths determined by GSA are input into the NLP version of $\mathbf{P_{opt}}$, which only performs rate allocation. In this version, there are no integer variables because routes are fixed.

Because routes found by $\mathbf{P_{opt}}$ depend on a particular network and interference model (eq. 4.5) (based on the protocol model of interference), this methodology allows to compare both solutions under the same model.

The topology used consists of a static network of 100 nodes placed randomly in a square 2000 m $\times$ 2000 m area. The topology is connected, with a minimum distance of 160 m between nodes, and maximum distance of 250 m. There are 4 gateways, one in each corner of the square. The protocol model of interference is used to generate the conflict graph and cliques, which determine constraints of the optimization problems. Maximum transmission range is 250 m and interference range is 550 m. Clique capacity is set to 280 KB/s. GWLB is configured with $P_\theta =$ 2.5 (represents a cost value close to interference range).

To generate random traffic with and without imbalance, the tests vary the number of flows in each gateway's native domain. The number of flows per native domain is a value in $\{0, 5, 10, 15, 20\}$. Scenarios are generated using all possible combinations of flows. Because there are four domains and five possible quantities of flows per domain, the total number of different scenarios is 5^4 1 = 624 (eliminating the case with zero

flows). For every unique scenario, flows are randomly assigned to a sink in the native domain.

To solve the MINLP problems, the Couenne solver [5, 14] has been used with the following parameters: a time limit of three hours, absolute termination tolerance of $1e \quad 9$ and relative termination tolerance of 0.005. All problems were solved before the time limit. Note that the level of approximation to the optimal solution depends on the tolerance thresholds, i.e. the solver may not reach the optimal solution, but rather a solution close to the optimal.

For comparison, results include routes used by NGW solution. In addition, to show the importance of taking path cost into account when selecting gateways, two gateway selection solutions are defined which perform load-balancing in the same way as GWLB, but with little or no regard to path cost. They are called *arb_lb*1 and *arb_lb*2, respectively. In both solutions, all gateways can be used to reach every sink, i.e. $|VG_s| = G, \ \forall s \in \mathbb{S}$. Both solutions use Algorithm 7, with the following differences:

- In *arb_lb*1, if there is more than one least loaded gateway in one iteration, one is chosen at random (line 5 of Alg. 7).
- In *arb_lb*2, if there is more than one least loaded gateway in one iteration, the least cost gateway is chosen (default behavior of Alg. 7).

Figure 4.2 shows the results. Each point is the average result of the scenarios that fall in that category. Confidence intervals are shown at the 95% level. Figs. 4.2 (a-c) present the results under varying total number of flows in the network, while Figs. 4.2 (d-f) present the results under varying flow imbalance between native domains. Figs. 4.2 (a,d) show the objective maximized by the MINLP and NLP problems (aggregate utility of flows). Figs. 4.2 (b,e) show the minimum rate of flows, and Figs. 4.2 (c,f) show the average rate of flows. As we can see, the performance of GSA routes is very close to the performance of routes found by $\mathbf{P_{opt}}$. The performance of *arb_lb*1 is noticeably worse than NGW in most cases. The *arb_lb*2 solution is also seen to perform worse than NGW in some cases, and rarely performs better.

The solutions found in this section seek the optimal rate allocation to achieve proportional fairness, given a protocol model of interference. The next section evaluates GWLB by simulation using TCP flows.

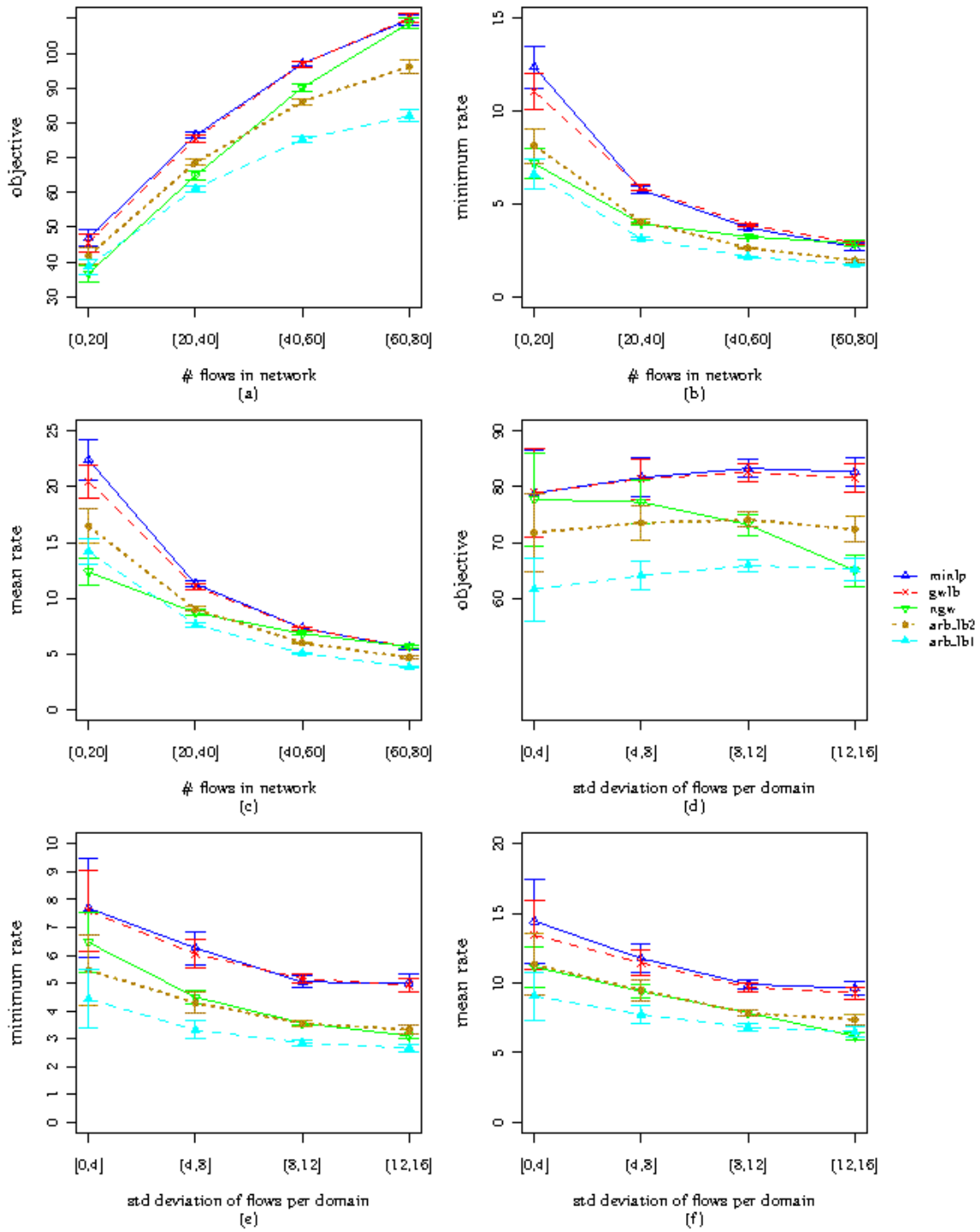


Figure 4.2: Comparison results of GWLB and optimal MINLP solution - This figure shows the performance comparison between MINLP, GWLB, NGW and arbitrary load-balancing solutions, under varying number of flows and varying load imbalance between native domains.

4.7 Performance evaluation

In this section, the performance of GWLB is evaluated in WMN scenarios with multiple gateways and a varying number of users in each native domain. GWLB is simulated with *ns-3* [3] and compared against three different gateway selection solutions: NGW, MLI [47] and MAU [78]. We choose to compare with MLI and MAU because, from our understanding of different proposed load-balancing protocols, they represent one of the best performing solutions of two different approaches used in the literature to execute load-balancing (explained in section 4.2). MLI seeks to even the load of gateways and MAU seeks to alleviate congestion by switching sinks from congested to uncongested gateways. In [33], we found MLI to be the best performing of the protocols in [47]. MLI indirectly takes contention into account, which enables it to perform better than solutions with similar goals.

MLI and MAU are distributed protocols, but their implementation in this work is centralized with no control overhead. Therefore, this implementation achieves an upper bound on their real performance.

For the MLI implementation, decisions are made centrally with complete knowledge of the topology, in each execution switching border sinks (sinks which neighbor nodes assigned to a different gateway) to a less loaded gateway. Border sinks are the first to receive advertisements from gateways different from their current gateway. In addition, the results of MLI illustrate the performance of its load metric, which is based on the current residual load of a gateway. Hyacinth [102], which uses a routing solution similar to MLI, also uses this metric. As we will see, this metric is not suitable for scenarios with TCP flows, where the flows generally use all available gateway capacity, i.e. the residual capacity of the gateways is always close to zero, independent of flow imbalance between gateways.

In the MAU implementation, a decision is made centrally to switch the best candidate sink of a congested gateway to its nearest uncongested gateway. Load is measured as the average queue length of the gateway during a time period. It is measured at the gateway because they are the likely bottlenecks. As we will show, congestion is not a reliable measure for balancing TCP flows. Flows perform congestion control, therefore congestion is usually temporary and does not correlate with the number of flows served by a gateway.

Table 4.2: ns-3 wireless parameters.

Wifi standard	802.11b @ 11mbps constant rate
Path loss model	Log-distance, exponent = 2.7
RTS/CTS	Disabled
Transmit power	$\min\{p : \text{PER}(p, l) \leq 5\%\} \quad \forall l \in L$
Energy detection threshold (EDth) (W)	RxPower(max neighbor distance)
Carrier sense threshold (CSth) (W)	EDth $\times$ 10 $^{9/10}$

4.7.1 Simulation environment

This work has implemented the protocols in *ns-3* in the following manner (illustrated in Figure 4.3). Wired links are 100 Mbps and wireless links are 11 Mbps. There is a server $\mathcal{A}$ outside the WMN to which users connect, and sends data to them. $\mathcal{A}$ is connected by a wired link to another machine $\mathcal{B}$, which implements and runs the different solutions. $\mathcal{B}$ is connected by wired links to every WMN gateway. When $\mathcal{B}$ receives traffic directed to the WMN, it forwards it to the appropriate gateway based on the current gateway selection (which is algorithm-dependent). $\mathcal{B}$ knows the complete topology, active flows, and can read the state of gateway queues. Routing inside the WMN is static shortest-path based on hop-count.

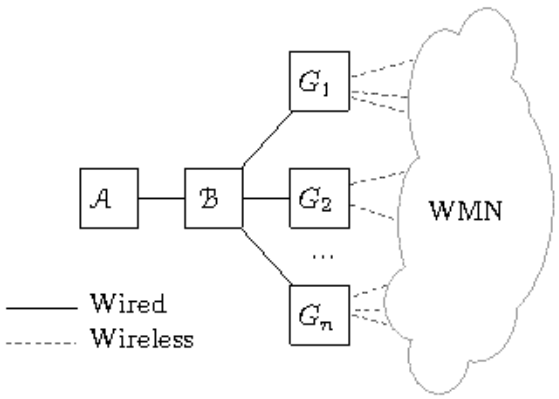


Figure 4.3: Gateway load-balancing simulation architecture implemented in *ns-3* - Nodes in the mesh network connect to the Internet server $\mathcal{A}$, generating TCP download flows. The gateway load-balancing controller runs in node $\mathcal{B}$ and is connected to every gateway.

The parameters used to configure wireless interfaces and propagation in *ns-3* are

shown in Table 4.2. The transmit power chosen is the minimum power that guarantees a Packet Error Rate (PER) of at most 5% for all links in the network (with no concurrent transmissions). The energy detection threshold is the received power at the maximum link distance. Carrier sense threshold is derived based on a ratio of $CSth/EDth$ of 9 dB. This ratio is chosen because it produced maximum aggregate throughput in numerous tests.

The WMN topologies consist of static networks of 100 nodes placed randomly in a square 2000 m $\times$ 2000 m area. All topologies are connected, with a minimum distance of 160 m between nodes. There are 4 gateways, one in each corner of the square, resulting in 4 domains.

Traffic is TCP. Users connect to $\mathcal{A}$, generating flows. For each user, the time between start of connections follows an exponential distribution with $\lambda = 1/10000$ ms, and the size of the download traffic of a flow follows an exponential distribution with $\lambda = 1/65000$ bytes. Simulations last until all flows finish, with users starting connections during the first 100 sec.

The number of users per native domain varies in $\{0, 5, 10, 15, 20, 25\}$. Given a topology, scenarios are generated using all combinations of users on every native domain. Because there are four domains and six possible quantities of users per domain, the total number of different scenarios given a topology is $6^4 \quad 1 = 1295$ (excluding the case where all domains have zero users). Five topologies have been generated, which gives a total of 6475 scenarios. For every unique scenario, users are randomly assigned to a node in their native domain.

The parameters of GWLB are set to $G_{\mathcal{T}} = 1s$, $Lock_t = 1s$ and $P_\theta = \quad 2.5$. A value of $G_{\mathcal{T}} = 1s$ provides high responsiveness to changing conditions. For P_θ, a value close to the average interference range is chosen. Both MLI and MAU are also applied every second. The congestion threshold value of MAU is set to 50%. The parameters for the MLI scheme are $\Delta_l = 1.3$ and $P_{MLI} = 0.7$, as in [47].

Confidence intervals are shown at the 95% level.

4.7.2 Performance metrics

The following metrics are used to study protocol performance. Flows refer to TCP download flows: (i) *Total network throughput* - total data bytes delivered by the network divided by the time elapsed since the first packet was sent and the last packet was

received; (ii) *Flow duration* - time elapsed since the first packet of a flow was sent and the last packet was received; (iii) *Flow throughput* - the number of bytes delivered divided by the duration of the flow; (iv) *Flow throughput fairness* - rates the fairness of the throughput achieved by flows in one scenario using Jain's fairness index.

The flow duration and throughput per scenario is taken as the median value of all flows. The median is chosen due to the frequent presence of outliers and skewed distribution of the flow metrics. For example, there can be scenarios where one flow benefits from a much higher throughput than other flows.

4.7.3 Results and analysis

Fig. 4.4 shows results under varying number of users in the network. Each point is the average result of the scenarios which fall in that category. It therefore includes results from a number of heterogeneous scenarios (with different topology, number of users, users per domain and location of users in each domain).

We can see in the figure that GWLB outperforms all protocols across all metrics. On average, MAU will perform similar to NGW, and MLI will perform better than both when the number of users is not too high, but degrading afterward.

As the number of users increases, load and number of flows increases, which leads to higher aggregate throughput (as long as there is sufficient capacity), and to lower average flow throughput. By balancing load, GWLB is able to maintain higher flow throughput, which also translates to higher aggregate throughput. When the number of users in the network is very large (more than 75) the native domains have a similar number of users, and GWLB defaults to the NGW solution, or a close-to-NGW solution. This also explains why the different protocols tend to converge on high load. The duration of flows follows a similar pattern. Note that congestion has a considerable impact on flow duration. Finally, we can also see that GWLB achieves better fairness between flows, by balancing load between gateways.

The results also show that MLI and MAU have inconsistent behavior. At best, MAU performs slightly better than NGW, and with high number of users performs worse. There are two main reasons for this: as explained earlier, MAU uses congestion as a load indicator, which is not reliable because it is usually temporary when using TCP. In addition, the choice of sink to switch when congestion is detected is suboptimal. When a domain is congested, the sink which maximizes $ETX \times rate$ is chosen, where

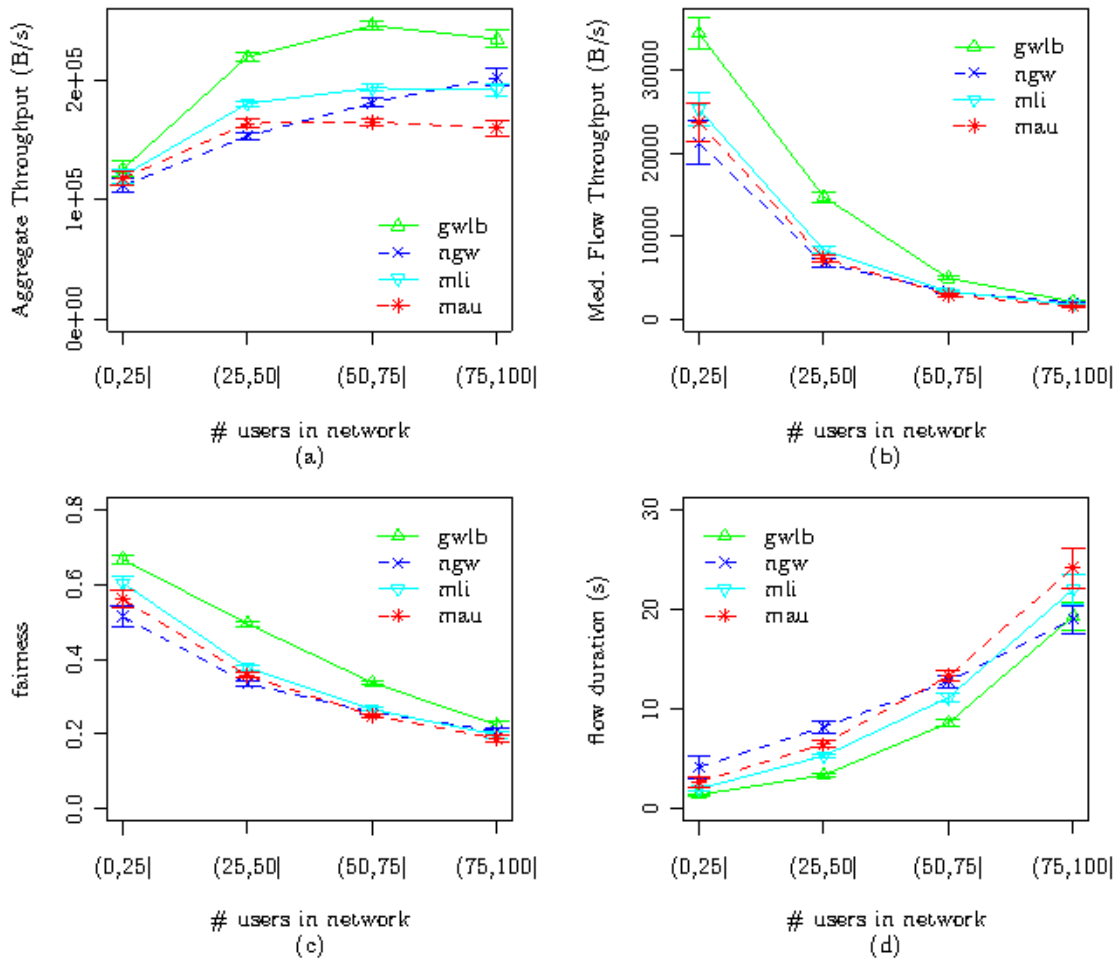


Figure 4.4: GWLB performance results with varying number of users - This figure shows the performance results of GWLB, NGW, MLI and MAU with varying number of users in the network.

ETX is the routing metric from the sink to its current gateway. With TCP flows, and due to the nature of multi-hop wireless transmission, flows of sinks closer to the gateway will have higher throughput. As a result, MAU may not select sinks which are farthest from the gateway but rather halfway. This means that the path to the new gateway will be longer, leading to increased contention in the network.

As for MLI, its performance is good when the number of users is below 50 but starts to degrade afterward, to the point of performing worse than NGW. This indicates that MLI makes erroneous decisions. The main reason behind this is that it attempts to even the load but doesn't take into account that TCP flows are elastic, and as such, differences in load don't necessarily correlate with differences in number of flows.

In addition to the results shown, it is important to note that comparing the throughput achieved by flows in the same scenario with GWLB over NGW, we have that the improvement in flow throughput per scenario when using GWLB is on average 128%. It is also important to note that the throughput of GWLB is noticeably less than NGW (more than 5%) only in 4% of all the scenarios tested, which indicates that GWLB seldom makes erroneous decisions, specially by avoiding arbitrary load-balancing (high path cost) which can increase contention and therefore prove detrimental.

Fig. 4.5 shows results under varying user imbalance between domains. This is measured as the standard deviation of the number of users in every native domain. Same as above, each value shown is the mean value taken from a large number of heterogeneous scenarios.

As in the above tests, GWLB performs best. We can observe how MLI once again makes erroneous decisions, which manifest when the imbalance is low. Only when the imbalance is high MLI distances itself from the other solutions. The performance of MAU is at best slightly better than NGW.

Increasing load imbalance progressively produces congestion in one or more domains, which leads to lower flow throughput and fairness and increased flow duration. Because GWLB balances load it is less affected by this. The performance of NGW is worse and decreases faster with increasing imbalance. Also, we can see that the difference in fairness between GWLB and NGW increases with load imbalance. By balancing load between gateways whenever possible and avoiding congestion, GWLB is more capable of maintaining inter-domain flow fairness.

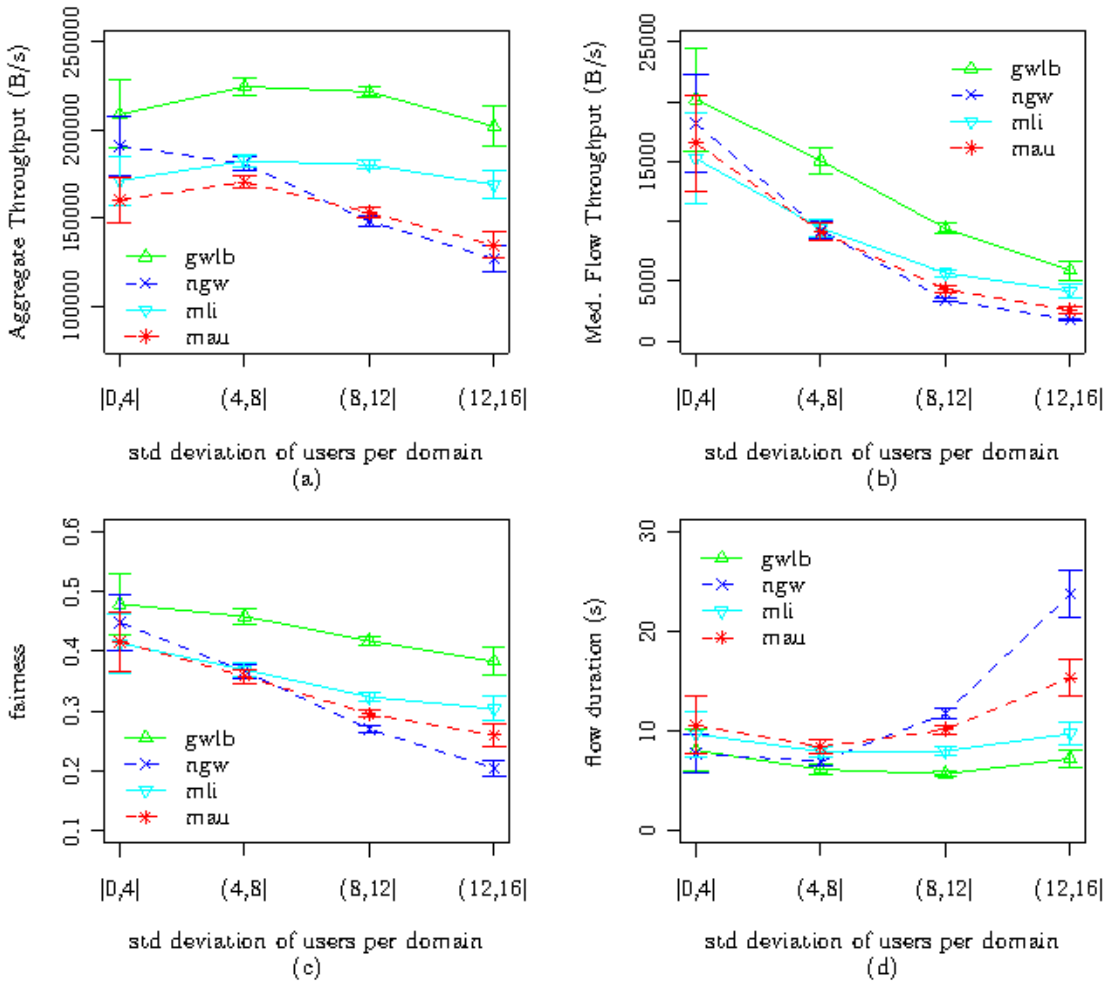


Figure 4.5: GWLB performance results with varying user imbalance - This figure shows the performance results of GWLB, NGW, MLI and MAU with varying user imbalance between native domains.

4.7.4 GWLB adaptation frequency

The responsiveness of the protocol depends on the frequency with which it adapts to changes in network conditions. Because at any instant the controller has knowledge of the current set of flows, responsiveness depends on the frequency with which it recalculates flow-gateway associations, as explained in section 4.3. This subsection evaluates the responsiveness of the protocol under highly dynamic traffic scenarios.

Tests are performed in one network topology with the same architecture and characteristics as described in section 4.7.1. Scenarios are generated with fixed number of active users in {5,10,15,20,25,30}. Given a quantity of users, 100 random scenarios are generated, totaling 600 unique scenarios. In each of these unique scenarios with x active users, the set of x users which are active at a time changes every 10 seconds and is chosen at random. A user is randomly assigned to a sink and generates connections in the same manner as described in 4.7.1, with the only difference that the time between start of connections follows an exponential distribution with $\lambda = 1/5$ s (this is to generate connections within the user's 10 second interval). Note that this produces scenarios which are highly dynamic. Connections start at second 25 and end at 125. A lower period G_T enables the protocol to adapt more quickly to changing conditions. Experiments vary G_T in $\{1, 10, 30, 60\}$ seconds. The other GWLB parameters remain as before.

The results are shown in Fig. 4.6. As we can see, the best performance is obtained with the highest frequency of adaptation, and the difference tends to increase with the number of users. When G_T is very low (one second), GWLB can respond quickly to changing conditions and is suitable for use in highly dynamic scenarios. Note that G_T can be as low as the time required to execute GSA.

4.8 Conclusions

Load-balancing between gateways in a WMN can be of crucial importance. Imbalance between domains can easily occur due to a number of reasons, including unplanned gateway placement or heterogeneous traffic demands. The limited wireless link capacity and interference aggravates the problem, turning gateways into bottlenecks, which can easily lead to congestion. Moreover, the roaming of end-systems across the network

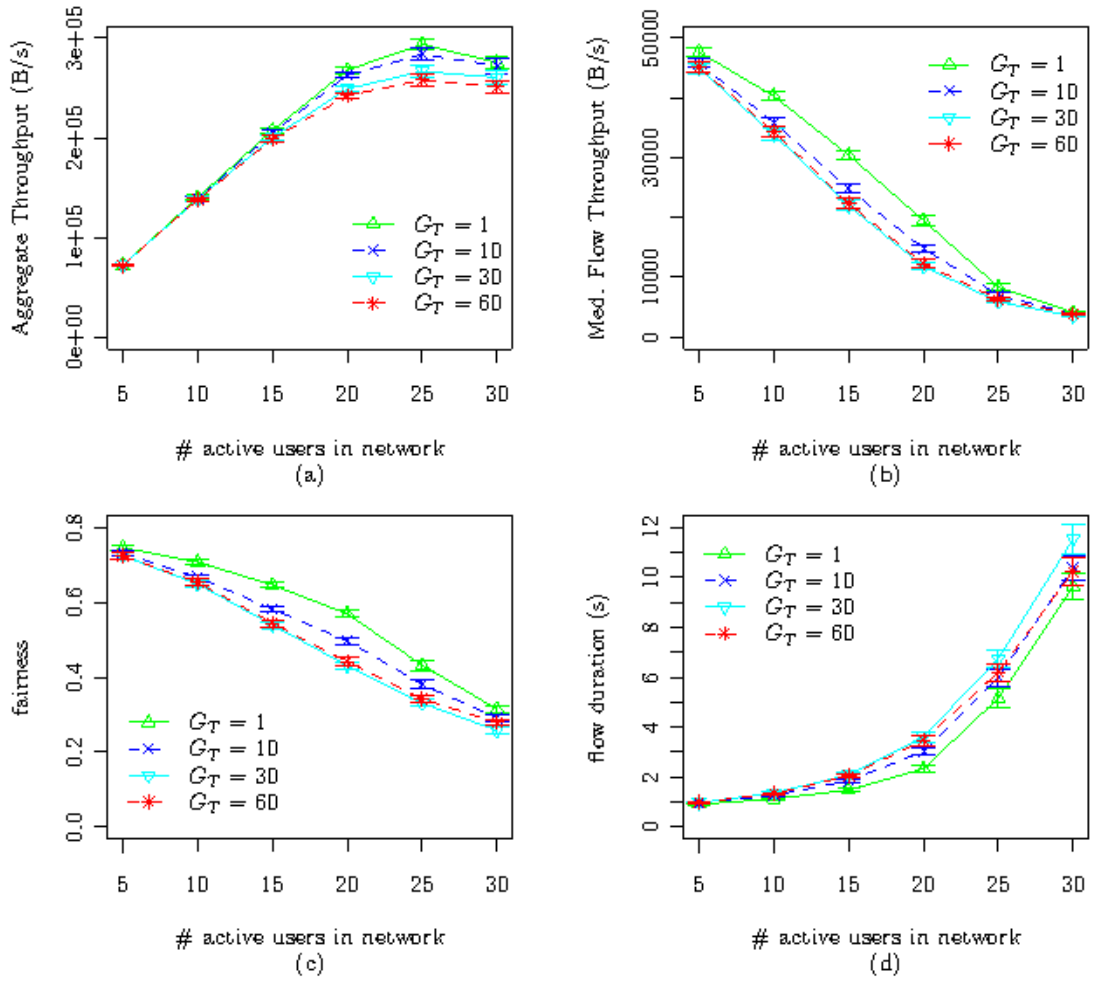


Figure 4.6: GWLB performance in highly dynamic scenarios - This figure shows the performance of GWLB in dynamic scenarios with varying adaptation frequency (determined by G_T).

together with variable radio conditions contribute to make demands more dynamic over time.

This chapter has proposed GWLB, a highly responsive online protocol which dynamically adapts to network conditions, balancing the load of gateways. It achieves improvements over shortest path routing in both throughput and fairness in scenarios with load imbalance. Importantly, because it is TCP flow-aware, balances traffic at the TCP flow level, and takes into account the effects of interference of flows when switching between domains, it is suitable for implementation in realistic scenarios.

Numerous tests have shown that GWLB can effectively balance load between gateways while avoiding the use of harmful paths in the network. This is achieved thanks to the use of the PSD metric defined in chapter 3. Using this strategy, GWLB effectively achieves a trade-off between balancing and traffic locality.

The simulations conducted in *ns-3* prove effectiveness of GWLB, and its advantage over previously proposed schemes. It outperforms all the alternatives tested. In particular, it achieves an average flow throughput gain of 128% over the nearest gateway strategy. GWLB specially distances itself from other solutions when congestion or load imbalance between domains increases, showing that the protocol can cope more effectively in these situations.

The next chapter addresses the problem of balancing the load served by a gateway inside the WMN, in multi-channel networks. The availability of multiple non-overlapping channels permits eliminating intra-flow and inter-flow interference, specially in the region surrounding the gateway, which cannot be done in single-channel networks. This permits finding routes in the network to balance the instantaneous load of a gateway. Routing and channel assignment will be considered jointly, due to their mutual dependency.

Chapter 5

LBCA: Load-balancing routing and channel assignment in multi-radio WMNs

This chapter addresses the problem of load-balancing in the gateway access scenario in multi-radio multi-channel Wireless Mesh Networks. The availability of multiple radios at each mesh node and multiple frequency channels can substantially increase the load balancing capability of the network. Joint routing and channel assignment is a highly effective mechanism to exploit this. This chapter presents a practical interference-aware flow-based load-balancing solution for WMNs based on the joint optimization of routing and channel assignment.

5.1 Introduction and motivation

Wireless Mesh Networks (WMNs) have recently attracted much attention. One of the main reasons is that they provide a cost-effective way of deploying a wide-area network and offer services such as Internet connectivity. Recent reduction in hardware costs permits equipping mesh routers with multiple radio interfaces. In these networks, the effective use of multiple non-overlapping channels (e.g. 3 in IEEE 802.11b/g and 12 in IEEE 802.11a) can significantly enhance the network capacity by allowing more concurrent transmissions. In this context, a key issue is the assignment of channels to radio interfaces to minimize interference. Balancing the load between collision domains in a

WMN is necessary to effectively use the capacity of the network, increasing throughput and flow fairness. Routing and channel assignment (CA) are two effective mechanisms to achieve this, more so when considered jointly.

In multi-radio multi-channel networks, a routing strategy can be used to find and use paths with low intra-flow and inter-flow interference. Interference is determined based on the channels assigned to transmission links and the spatial reuse factor. In other words, links transmitting in different channels will not interfere, as well as links separated in space. A load-balancing routing protocol must ultimately be able to balance the instantaneous traffic demands through these paths while avoiding network instability. Channel assignment, on the other hand, can be used to reduce both intra-flow and inter-flow interference, by determining the channel assigned to each link. Traffic-aware CA has the important benefit of optimizing radio resources taking into account real traffic patterns. It can reduce interference in links carrying more traffic, modifying and granting capacity to collision domains based on their load. When considering traffic-aware CA, there is a well-known interdependency between routing and channel assignment, as explained previously in section 2.5.5. Channel assignment affects routing, because a routing decision needs to choose paths that avoid interference. Routing, on the other hand, determines link load which also influences CA: a channel allocation procedure needs to grant more bandwidth to links with more traffic. The joint routing and channel assignment problem is NP-hard and is frequently solved by separately calculating routing and channel assignment (e.g. [8, 103]).

The previous chapter studied load-balancing in single-channel WMNs in the gateway access scenario. In this context, routing through gateway selection can be used to balance load between gateways while at the same time being able to avoid inter-flow interference. However, it was explained that in single-channel networks the traffic served by one gateway cannot be effectively balanced because every flow ultimately contends in the region surrounding the gateway. In multi-channel networks, however, CA can be used to reduce or even eliminate interference in this region, making it possible to balance the traffic inside a gateway domain. This chapter develops a joint routing and CA solution to balance the traffic inside a gateway domain.

Many works have studied the subject of joint routing and CA, and many have developed centralized algorithms to optimize routes and CA. One of the main limitations of previous centralized approaches is that they assume mostly static network conditions

where, for example, the traffic profile is known and expected to remain stable over long periods of time (hours or days). As such, the solution calculated by these systems is assumed to remain valid for a long period of time. In practice, network conditions (specially the instantaneous traffic demands) can vary frequently and unpredictably, and a centralized solution must be able to rapidly collect the necessary network state, calculate a solution and apply it in the network, while introducing low overhead. If CA is calculated centrally, a change in channel allocation must be communicated to all affected nodes in the network. This can generate overhead and incur in latency due to channel switching time of radio interfaces[1], temporarily disrupting network activity. When channel re-assignment occurs frequently (due to rapidly changing traffic conditions), it is important that the CA algorithm be able to limit the number of channel re-adjustments between channel assignments. In other words, the number of edges which have to change their channel must be kept to a minimum. This will reduce both the channel switching time, as well as the information that needs to be disseminated through the network.

Another limitation of previous centralized approaches is that they make unrealistic and simplifying assumptions regarding the network and interference model. The purpose of this work is to assume conditions which occur in practical WMN deployments and develop a solution suitable for commodity systems such as those based on 802.11. This work makes the following assumptions:

- Most traffic is TCP (individual flows have no specific demand and are elastic). It is also required that traffic belonging to one flow not be split among multiple paths (to avoid out of order delivery of packets and round-trip time variations which are harmful to TCP).

- The traffic matrix is unknown and, furthermore, this work contends that it cannot be known for two main reasons: (a) because the capacity of a WMN is limited, users will tend to use all available capacity. This means that routing and channel assignment affects resulting demands, and those demands don't necessarily reflect user requirements. As soon as routing or channel assignment changes, demands

[1] The switching delay for current 802.11 hardware ranges from a few milliseconds to a few hundred microseconds [103].

can change; (b) the small scale of a WMN leads to frequent and unpredictable traffic variations.

- The network model is based on a random access-based medium access model (e.g. DCF used by 802.11) and the SINR model of interference. This leads to the fact that modeling network throughput analytically is highly complex for various reasons, which include a dynamic wireless medium, complex interference interactions and MAC inefficiencies. Works which require a throughput model usually resort to some combination of the following: time-slotted synchronized medium access model [8, 62], unrealistic binary interference model or throughput estimation [81, 103, 114]. See section 2.2.3 for discussion of interference models.

As will be further explained in the next section, most previous research work on centralized routing and channel assignment in WMNs is incompatible with at least one of the above assumptions.

Another approach to routing and channel assignment is to use distributed protocols. The distributed and load-sensitive nature of these approaches imposes a challenge in the form of avoiding network instability. From the routing perspective, load-sensitivity can lead to route instability (frequent route oscillations), non-convergent behavior or slow convergence, and high overhead to communicate route updates through the network (this issue was discussed in section 2.4.6). In a similar way, a distributed CA protocol has to guarantee channel allocation stability. If traffic conditions do not change or change minimally, the protocol must maintain a stable CA. In addition, it must guarantee convergence and low convergence time, deal with overhead resulting from frequent node coordination, and avoid frequent and unnecessary channel re-adjustments that incur delay. As an example, some existing distributed approaches (e.g. [80, 102]) rely on static interface-to-neighbor bindings to prevent a local channel re-assignment decision from causing channel re-assignments across the whole network, also known as *ripple effect*. The drawback of this approach is that static bindings can restrict channel assignment, as explained in section 2.5.1.1. Another challenge is how to successfully couple routing and CA in a distributed approach and guarantee convergence. In other words, if routing and CA are two separate distributed processes, and the solution of one affects the other, the design of a distributed protocol must answer the following questions: will the routing and CA solutions converge? When will they converge?

This chapter develops a centralized traffic-aware routing and CA protocol for multi-radio multi-channel WMNs. The problem involves assigning channels to radio interfaces and calculating routes to optimize the performance of a set of TCP flows (flow throughput and fairness). To our knowledge, this is the first work that jointly balances traffic at the TCP flow level in a WMN and optimizes CA for a specific set of TCP flows. One of the main challenges of this approach is that, at the TCP flow level, rapid traffic fluctuations are expected and this imposes the need for frequent route calculation and channel re-assignment.

The proposed Load-Balancing and Channel Assignment (LBCA) system is intended for implementation in practical and dynamic WMN scenarios. It is an online solution which, given the current network and traffic conditions, efficiently calculates CA and assigns paths to every flow. It is assumed that most traffic is directed to/from the Internet. As such, there is no added overhead involved to determine current network load (this information is collected by the gateway as traffic passes through it). The solution can be recalculated periodically with a very high frequency to ensure responsiveness. Because it is calculated centrally, it will not suffer from convergence issues. That is, the solution converges, and it does so quickly, owing to the low time complexity of the proposed routing and CA algorithms. Frequent route oscillation does not occur because a flow can easily be prevented from switching routes until a specified time elapses. In a similar way, to prevent frequent channel re-adjustments in the network, the proposed CA algorithm [39] explicitly takes into account the previous CA when calculating a new channel allocation, and limits the number of edge channel re-adjustments. This work shows that channel re-assignment with a period of a few seconds is realizable in gateway access scenarios. To disseminate CAs, LBCA implements a novel protocol to quickly distribute channel allocation messages with low overhead. This CA distribution protocol does not require one interface in every node tuned to a common channel, as opposed to other existing protocols.

The rest of the chapter is organized as follows. Section 5.2 reviews related work. Section 5.3 describes the network model, defines and formulates the routing and CA problem. Section 5.4 describes the Load-Balancing Routing algorithm. Section 5.5 develops the Load-Aware Channel Assignment algorithm. In section 5.6, a network architecture is proposed to scale the capacity of the network. Section 5.7 explains the

LBCA protocol. Section 5.8 shows and analyzes protocol performance based on graph-theoretic performance metrics and simulations with *ns-3*. Finally, section 5.9 concludes the chapter.

5.2 Related work

In the last years there has appeared a large body of work on the subject of joint routing and channel assignment in WMNs. Some of these solutions were briefly discussed in section 2.5.5. This section focuses on the differences between previous approaches and the work described in this chapter.

Most of the centralized solutions described below calculate *flow*, which refers to traffic per unit of time traversing a link or node. We will write *flow* in italics to differentiate it from an IP flow, as defined in [19]. These solutions allocate specific *flow* to links, and thus require throughput models to guarantee that such allocations are feasible (are schedulable).

Most existing proposals can be classified broadly in two groups: centralized and offline solutions [8, 10, 41, 81, 103, 114] and distributive and online solutions [25, 64, 71, 102].

Raniwala et al. proposed centralized and distributed solutions. In [103], they propose a centralized solution that repeatedly applies separate routing and channel assignment algorithms until a solution converges. The routing algorithm calculates link-*flow* allocation while the channel assignment algorithm ensures that such routing is feasible. The solution is calculated offline assuming knowledge of a traffic matrix which specifies long-term average demands between all nodes, and thus intended to be valid for hours or days. Throughput is estimated based on a binary interference model. In [102], they propose a more practical distributed solution, which periodically modifies routes and channel assignment distributively based on current network load. Routing forms a tree structure rooted at the gateway. Of the routing metrics proposed, only one is suitable for balancing load in single-gateway networks, and consists in estimating residual bandwidth on a path. In the presence of TCP flows, the metric cannot discriminate between paths with a large number of flows and paths with few of them, due to flows utilizing all available bandwidth. Routing and load-balancing are decoupled and there is no guarantee of convergence (both are load-aware and one affects

the other). Further, required convergence time can make the protocol unsuitable for highly dynamic scenarios (simulation results for a sample scenario show a convergence time of two minutes). This solution also relies on static interface-to-neighbor bindings to avoid the ripple effect.

In [8], Alicherry et al. propose an approximation algorithm to solve a joint routing and channel assignment formulation. Their model assumes time-slotted synchronized medium access and a binary interference model in order to calculate interference-free scheduling. It requires knowledge of the traffic matrix and calculates splittable *flow* (the total demand between a source and destination is generally routed through multiple paths). Tang et al. [114] use mathematical optimization to calculate *flow* allocation, routing and channel assignment. The problem does not require a traffic matrix but rather assumes that all nodes are active simultaneously and allocates *flow* to each to achieve max-min fairness. Throughput is estimated based on a binary interference model. Mohsenian-Rad et al. [81] present a MILP formulation to calculate topology, channel assignment and routing. Similar to the above, problem assumes knowledge of traffic matrix, and calculates *flow*-link allocation. Avallone et al. develop a centralized offline heuristic for the joint routing and channel assignment problem [10]. It calculates link-*flow* allocation and channel assignment to maximize aggregate throughput, assuming all nodes are active simultaneously communicating with the external network. The method of precomputing *flow* calculates splittable *flow*, and additionally does not balance load in the network, because maximum *flow* computations are independent of each other (calculated as single-source, single-sink problems). Although their model uses SINR, the interference model is binary (cumulative interference is not taken into account and neither is link error probability (fractional interference)). The solution requires a special forwarding paradigm to ensure that *flow* allocation is schedulable in conventional MACs. In [41], Gardellin et al. propose a method to solve a joint routing and channel assignment problem by decomposing the problem into several ILP optimization subproblems which are solved optimally in sequence and later combining the solution in a post-processing phase. Calculates link-*flow* allocation and knowledge of the traffic matrix is required.

In [101], Ramachandran et al. propose a practical multi-radio WMN architecture. The online centralized channel selection algorithm (TIC) computes routes between the gateway and mesh routers and allocates channels to links on these routes. TIC

requires that all nodes operate one of its radios on a common default channel in order to coordinate topology measurement and disseminate channel switch commands. Routing and channel assignment are not load-aware.

Dhananjay et al. [25] propose a distributed channel assignment and routing protocol. The channel assigned to a node's interfaces depends on the hop-distance of the node to the selected gateway, with the purpose of assigning different channels to all links in the path to the gateway; channel assignment is not load-aware. Kyasanur et al. propose a distributed link-layer protocol for interface-channel assignment and a multi-channel aware routing metric [64]. Each router chooses a channel least used by nodes in its neighborhood without coordination with other routers. Routing metric and channel assignment are not load-aware. Lin et al. present a distributed solution for channel assignment, scheduling and routing in [71]. The system model is based on time-slotted synchronized access and binary interference model.

It is important to note that none of the above are intended for solution recalculation in small timescales (order of few seconds).

To our knowledge, the issue of channel re-assignment to minimize the number of channel re-adjustments was first studied in [9], with the objective of calculating a new CA without exceeding a specified number of re-adjustments.

Regarding the channel assignment algorithm, Subramanian et al. proposed centralized and distributed algorithms for the CA problem [112]. The mathematical form of their traffic-aware objective function is equivalent to the one optimized by the algorithm proposed in this work. This chapter compares the proposed algorithm with the centralized algorithm of the above paper, showing that the proposed one finds comparable solutions while being substantially faster. This work's CA algorithm is also shown to perform better than a heuristic which, not taking into account real loads, establishes the priority of edges based on their distance to the gateway (this strategy is used e.g. in [99]).

The system developed in this work is online, does not require a priori knowledge of traffic, adapts to real-time variations in traffic, and has high responsiveness due to high adaptation frequency (order of seconds). It routes and balances traffic at the TCP flow level. Channel assignment is load-aware and based on the physical model of interference. Channel re-assignment limits the number of channel re-adjustments, and the CA distribution protocol does not require one interface in every node tuned to a

common channel. Furthermore, the system does not rely on a throughput model, and rate allocation is performed by TCP.

5.3 Joint routing and channel assignment problem

This section explains and formalizes the joint routing and channel assignment problem. First it presents the network and interference model assumed in this chapter. It then proceeds to give an overview of the problem, followed by its mathematical formulation and subsequent decomposition into a separate routing and channel assignment problem.

5.3.1 Network and traffic model

This work considers WMNs which comprise of wireless static or quasi-static mesh *routers*, also called *nodes*. These nodes form a wireless multi-hop network. Mesh *clients*, also called *users*, connect to the mesh routers. A subset of nodes, referred to as *gateways*, are directly connected to a fixed infrastructure, which we will assume is the Internet. Each router has multiple radio interfaces with omni-directional antennas (e.g. 802.11a/b/g) for communication with other routers. Each interface can be tuned to one of several non-overlapping channels. In this system, there is no need to tune one radio of every node to a common channel. Communication between nodes and users is done via a separate interface and channel (wired or wireless). An example WMN was shown in Figure 1.1.

Although intra-WMN communication is possible, it is assumed that most of the traffic will be received from the Internet. Users are randomly located in the network. A user accesses the Internet through one or more links leading from its router to the gateway. To model Internet traffic realistically, it is assumed that all traffic entering the WMN is elastic, and that in any instant a user can initiate a connection to the Internet generating a download flow of any number of bytes. This is safe to assume, because the majority of Internet traffic today uses TCP.

As a consequence to the above, flows are elastic and have no specific demands. Traffic can be highly dynamic, and network conditions can constantly change.

The interface-channel usage policy (see section 2.5.1.2) used in this chapter can be considered a static assignment policy, in the sense that the time a channel remains assigned to a radio interface is long compared with the packet transmission time. In

practice, however, the proposed protocol supports channel re-assignment in the order of seconds.

5.3.2 Interference model

For the design of this system, the SINR model of interference is assumed (see section 2.2.3.2). The average signal strength at the receiver depends on the sender's transmission power, the path loss (due to any number of factors such as distance and obstacles) and fading. The SINR at the receiver and the modulation used determines the expected Bit Error Rate (BER) [45], and consequently the expected Packet Error Rate (PER).

A transmission link is specified by a sender and a receiver. The information the proposed protocol uses is the PER of links. Specifically, let $\text{per}(u)$ be the PER of transmission link u in the presence of ambient noise and no external interference. Let $\text{per}(u|v)$ be the PER of transmission link u in the presence of ambient noise and concurrent transmission of link v. In general $\text{per}(u|v) \neq \text{per}(v|u)$. If u and v are on different channels $\text{per}(u|v) = \text{per}(u)$. Note that this PER model takes into account *fractional* interference. In other words, it is not a binary interference model. However, it does not take directly into account cumulative interference, i.e. the PER of a link when multiple links transmit concurrently (e.g. $\text{per}(u|v, w, x)$), but as we will show, the channel assignment algorithm includes a simple mechanism to reduce its effect.

5.3.3 Problem preliminaries

In the same way as defined in chapter 4, an Internet flow is a set of TCP packets exchanged between two endpoints[1], one in the Internet and one in the WMN. Let F denote the set of Internet flows. We call a router in the WMN participating in a flow a *sink*. In general, flows are created after a user in the WMN connects to a server on the Internet.

The topology graph is modeled by an undirected graph $G = (V, E)$. We use the notation e_{mn} to refer to an edge e with pair of vertices (m, n). In the case of a directed edge, the order of the vertices in e_{mn} conveys direction. Let $dir(u_{mn}) = \{v_{mn}, v_{nm}\}$ be the pair of directed edges associated with an undirected edge u_{mn}. Let $D = \bigcup_{e \in E} dir(e)$ denote the set of all directed edges corresponding to edges in E. Directed edges model

[1] In this work an endpoint is a pair (address,port).

Table 5.1: Notation used in this chapter.

F	Set of Internet flows
G	$G = (V, E)$ is the undirected connectivity graph of the network
C	Set of orthogonal channels available in the system
$R(n)$	Set of radio interfaces available at node $n \in V$
R	The number of radios in every node. In this chapter, without loss of generality $R = \|R(m)\| = \|R(n)\| \; \forall m, n \in V$.
$sink(f)$	Endpoint of flow f in the WMN
e_{mn}	Edge whose endpoints are the vertices (m, n)
$dir(u_{mn})$	Pair of directed edges of undirected edge u_{mn}: $dir(u_{mn}) = \{v_{mn}, v_{nm}\}$
D	Set of all directed edges corresponding to edges in E: $D = \bigcup_{e \in E} dir(e)$
$inv(e_{mn})$	Inverted edge of directed edge e_{mn}, i.e. $inv(e_{mn}) = e_{nm}$
$E(n)$	Set of edges incident on node n, i.e. $E(n) = \{e_{ab} \in E \mid n \in \{a, b\}\}$
$E(p)$	Set of edges in path p
$\text{per}(u)$	PER of directed edge e when transmitting in isolation
$\text{per}(u\|v)$	PER of directed edge u when transmitting concurrently with edge v
ϕ	Flow-path assignment: $\phi(f) = p$ if flow $f \in F$ is assigned to path p
χ	Edge-channel assignment: $\chi(e) = c$ if $e \in E$ is assigned to channel $c \in C$
$t(e)$	Number of flows traversing edge e

unidirectional transmission links (given by a transmitter and receiver pair). They are used to model interference and direction of traffic. Undirected edges model a bidirectional link, and are used for channel assignment. All valid communication links are bidirectional, because every transmission requires acknowledgments. This work assumes uniform link capacity. Let $inv(e_{mn}) = e_{nm}$. Let $E(n) = \{e_{ab} \in E \mid n \in \{a, b\}\}$ be the set of edges incident on node n.

A directed edge e exists between two nodes if and only if $\text{per}(e) \leq X$ (in this work $X = 0.05$). An undirected edge represents a bidirectional communication link between the radio interface of one node and the interface of another. For practical reasons, this work only considers one undirected edge between two nodes. To effectively use multiple edges between nodes, the edges must be assigned to different channels, and therefore to different interfaces. However, having multiple routes to the same neighbor (i.e. through different interfaces) adds complexity to the routing layer, and can lead

to packets arriving in different order at the destination, as explained in section 2.5.1.1. Note that although multiple edges between two nodes can allow higher throughput, more interfaces per node and channels are needed to effectively exploit this.

The set of channels is denoted by C. Let $R(n)$ be the set of radio interfaces of node n. For ease of exposition it is assumed, without loss of generality, that $|R(m)| = |R(n)| = R \; \forall m, n \in V$. A channel assignment is a function $\chi : E \mapsto C$. We denote $\chi(e) = c$ if channel c is assigned to edge e. Channels are assigned to undirected edges or, in other words, the same channel is assigned to a directed edge and its inverse edge (a node expects to receive acknowledgments on the same interface). A channel is assigned to every edge, and so channel allocation does not alter network connectivity.

A path $p = (n_1, n_2, ..., n_l)$ is a sequence of unique nodes. Let $E(p)$ denote the set of edges in the path, i.e. $E(p) = \{e_{ab}\}$ where a and b are two consecutive nodes in p. Note that a path conveys two directions: from n_1 to n_l and from n_l to n_1. A routing decision assigns a path to each flow (connecting the gateway and the flow's sink). We denote $\phi(f) = p$ if flow f is assigned to path p. A routing decision therefore assigns all traffic of a flow to a single path (download and upload traffic use the same path, in different directions). This leads to a NP-hard routing problem as we will show, but is important to simplify actual routing of packets and to not harm the performance of TCP flows (avoiding out of order delivery of packets and round-trip time variations). Denote $\mathrm{t}(e) = |\{f \in F \mid e \in E(\phi(f))\}|$ as the number of flows traversing undirected edge e. The number of flows traversing a directed edge is the same as those traversing its corresponding undirected edge (flows have traffic in the download and upload direction).

Given a set of TCP Internet flows, the goal is to find a flow-path assignment ϕ and edge-channel assignment χ, so as to optimize flow performance (flow throughput and fairness). The objective is to balance load between edges and collision domains via routing and CA, improving utilization and fairness. Note that routing and CA are load-aware, to optimize the performance of the *current* set of flows. This is a joint routing and CA problem; in the next subsections this will be developed further.

Table 5.1 lists and summarizes notation used throughout this chapter.

5.3.4 Routing and channel assignment system overview

The Load-Balancing and Channel Assignment (LBCA) system proposed in this chapter consists in a centralized approach which, given the current set of Internet flows F and

network state, calculates a flow-path association ϕ and edge-channel allocation χ. The solution is calculated centrally at the gateway, also called the *controller*, and recalculated periodically with a high frequency to be able to adapt to frequently changing conditions. This work considers adaptation periods as small as one second. Conditions can be very dynamic, so it is important to be able to rapidly calculate a solution and apply it in the network.

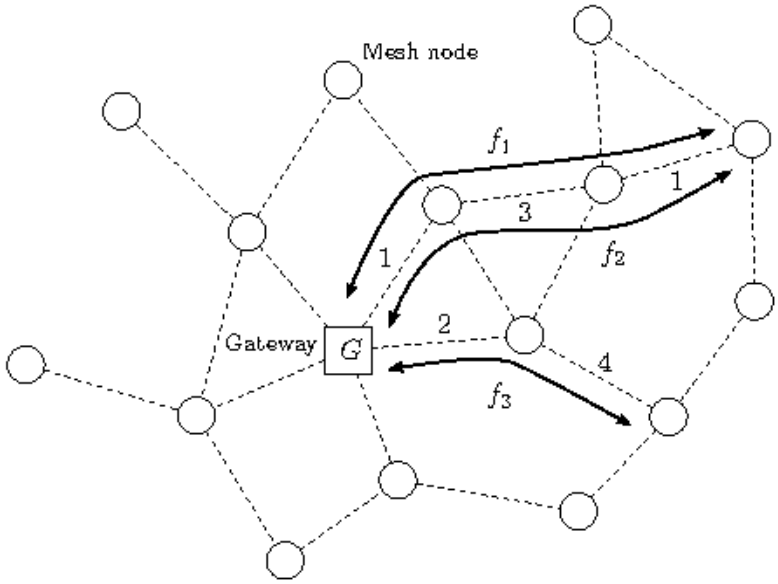
The solution only concerns Internet flows. In other words, load-balanced routing is performed only on these flows. For the rest of the traffic, it is assumed that a secondary routing protocol is executed inside the WMN, which establishes routes between the WMN nodes. No such protocol is imposed. Similarly, channel assignment only considers Internet flows. The problem is approached in this fashion for two main reasons: (a) Internet flows constitute the majority of traffic load; (b) the gateway can effectively perform load-balancing and channel allocation in a centralized manner, with the main benefit of fast adaptation while avoiding network instability, as we will show.

Figure 5.1 illustrates a simple example of LBCA operation. In this network, there are currently three active Internet flows (f_1,f_2,f_3), two of them belonging to the same sink.

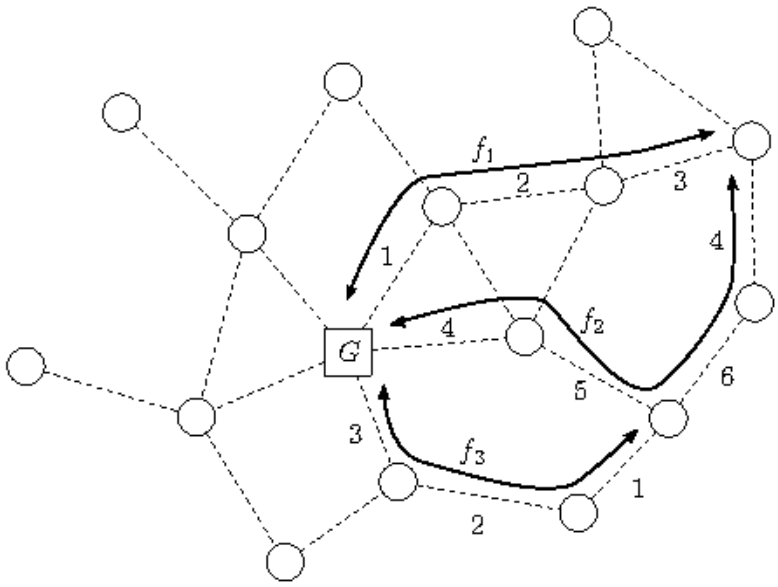
In Figure 5.1 (a), the system does not perform load balancing routing nor load-aware CA. It uses shortest path routing based on minimum hop count. The end result of this strategy is that flows f_1 and f_2 follow the same path and have to share its capacity. Furthermore, because the CA is not tailored to the current traffic, we see the same channel (channel 1) assigned to two edges of the path followed by f_1 and f_2, generating intra-flow interference.

Figure 5.1 (b) shows a simple example of load balancing routing and load-aware CA. In this case, the flows follow independent paths, at the cost of f_2 and f_3 using slightly longer paths. In addition, a CA is calculated tailored to the path followed by the flows, which eliminates all or almost all intra-flow and inter-flow interference. In summary, the strategy obtains interference-free paths and the capacity of these paths is close or equal to the maximum link capacity.

To calculate a routing and CA solution, the gateway needs to know the current set of Internet flows F, the network topology and link costs, and the Interference Matrix $\mathbf{IM}$, where $\mathbf{IM}_{u,v} = \text{per}(u|v)\ \forall u, v \in D$. Details on how this information is collected are in section 5.7. This information can be obtained with low overhead.



(a) Shortest path routing (no load balancing)



(b) Load balancing and load-aware CA

Figure 5.1: LBCA simple example - In this example there are six orthogonal channels {1,2,3,4,5,6}, and three active Internet flows (f_1,f_2,f_3). Arrows indicate paths followed by flows. Edge labels indicate the channel assigned to an edge.

More importantly, the information of the current flows F, which is expected to vary frequently, is obtained in real-time by the gateway without introducing any overhead in the WMN. This information is collected by the gateway as Internet flows pass through it.

To allow flows to follow the routes calculated by the gateway, the gateway uses source routing. More specifically, when download traffic belonging to a flow enters the WMN, the gateway inserts the path calculated for the flow into the header of download packets. In a similar way, when the sink receives download packets, it extracts the source route from their header and inserts the reverse route into upload packets.

The CA is traffic-aware and can therefore change every time the set of flows changes. A change in CA requires disseminating the new allocation to the affected nodes in the network via special channel assignment messages (CAM). Section 5.7.3 develops an efficient protocol to achieve this. A crucial mechanism to reduce the size and number of CAMs, the time required to send CAMs to affected nodes, and the time to switch channels in each radio interface, is to limit the number of channel re-adjustments between channel re-assignments. That is, it is important that the CA algorithm calculate a new CA varying the channel of the least number of edges possible.

One of the main benefits of the proposed centralized approach is that it avoids network instability problems that frequently arise when using load-sensitive metrics and protocols. See section 2.4.6 for a discussion on this issue. Because the solution is calculated centrally, it is guaranteed to converge, and it does so quickly, due to the low time complexity of the proposed routing and CA algorithms. In addition, path stability can be guaranteed by locking the path assigned to a flow for a specified time.

Further details on the LBCA architecture and protocol are given in section 5.7. The next sections focus on the routing and CA optimization problems, and the algorithms executed centrally at the gateway to solve them.

5.3.5 Joint routing and channel assignment problem

This work considers dynamic traffic based on TCP flows. Flows are elastic and don't have a specific demand. Rate control and bandwidth sharing depend on TCP. The goal is to optimize the performance of flows, increasing flow throughout and fairness. To that end, this work proposes solving the following joint routing and channel assignment problem:

$$\min_{\phi,\chi} \max_{e \in D} \{u(e)\} \tag{5.1}$$

where

$$u(e) = \mathrm{t}(e) + \sum_{v \in D \quad \{e\}} \mathrm{t}(v)\, \mathrm{per}(e|v) \tag{5.2}$$

The objective consists in finding the flow-path assignment ϕ and edge-channel assignment χ that minimizes the maximum edge utilization. Note that ϕ affects the number of flows in each edge $t(e)$ while χ affects $\mathrm{per}(u|v)\ \forall u, v \in D$. Minimizing the utilization of links (as defined in equation 5.2) implies balancing the load across the network (spreading flows across different edges and collision domains) and minimizing interference between these domains. By minimizing the utilization of a link, the capacity available to the link can be shared among fewer flows, increasing throughput. Because load is balanced across collision domains, fairness is improved (e.g. by avoiding cases where there are domains with many contending flows and domains with few). From the objective, it is clear that there is an interdependency between routing and channel assignment. Routing needs to choose paths with lower interference, while channel assignment needs to allocate channels to reduce interference on the paths carrying traffic. This suggests that to achieve an optimal solution both problems should be solved jointly.

Individually, the flow routing and channel assignment problems are NP-hard, as we will show in the next subsections. The joint problem is even more complex. At the same time, a fast algorithm is necessary to permit frequent recalculation, in order to promptly adapt to changing network conditions.

Although the problem is NP-hard, the particular network architecture considered in this work facilitates solving the problem, i.e. it is easier to obtain a solution close to the optimal solution. This permits using a simple efficient algorithm to solve the problem. In this architecture, there is a well-defined traffic pattern, characterized by flows going through the gateway. Links closer to the gateway will carry more load (more flows) while, as flows spread out, links farther from the gateway will carry less load. The maximum total throughput is limited by the aggregate capacity of the edges incident on the gateway. Therefore the optimal solution mainly involves balancing the

load across bottleneck links (which are clearly defined) and minimizing interference on these links.

Taking the above into account, this work proposes a two-step process to solve the problem: the first step performs routing and the second calculates CA. Routing and CA are performed only once, contributing to a low time complexity. This procedure is illustrated in Figure 5.2. The routing process assumes a interference-free network, and focuses on balancing the load across the bottleneck links, while the channel assignment process will give priority to reducing interference on these links. The next subsections formulate the individual problems.

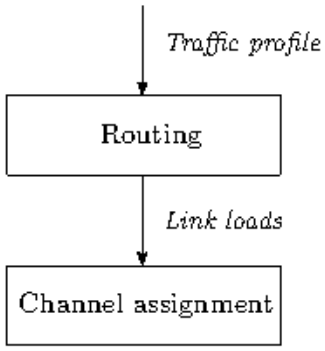


Figure 5.2: Routing and channel assignment strategy - The routing procedure takes as input the current traffic profile (set of Internet flows) in the network and assigns a path to each flow. This determines the load on each link. Based on this, the CA procedure calculates a CA.

5.3.6 Routing problem

For this step a interference-free network is assumed. The problem takes as input the set of flows F and the topology graph G. The load-balancing routing problem consists in finding a path for each flow such that load is balanced across the network. However, arbitrary load-balancing doesn't take into account the length of paths and can result in overly long paths. A trade-off is involved to optimize both load balance and path cost. For example, the shortest paths solution is the minimum path cost solution but does not balance load.

Based on the above, the following multi-objective problem is formulated:

$$\mathbf{P_{LBR}} : \min_{\phi}[\max_{e \in E}\{t(e)\}, \sum_{f \in F} cost(\phi(f))] \tag{5.3}$$

The goal is to find the flow-path assignment ϕ that optimizes both conflicting objectives. The first objective is to minimize the maximum utilization of edges, defined as the number of flows using the edge. The second objective seeks to minimize the cost of paths (one way to define cost is the number of edges (hops) in the path). Note that the link cost metric must be topology-dependent (*not* load-sensitive) because load is already known and controlled by solving this problem.

Minimizing the first objective alone is NP-hard. Consider the decision problem variant of the routing problem $\mathbf{P_{LBR}}$: Does a routing ϕ exist in G such that the maximum utilization $t(e)$ of an edge is less or equal to X? If $X = 1$, this problem is equivalent to asking if there exists a routing in G that produces mutually edge-disjoint paths for all flows (i.e. between every pair of endpoints). Determining if this is realizable is one of Karp's original NP-complete problems [58].

Both objectives can be evaluated from easily measurable network conditions: the current set of active flows F and the current topology and link costs (details on how this information is collected are given in section 5.7.1).

The goal of the proposed routing is to balance the load of edges. Per-flow routing is supported (i.e. flows of the same sink can follow different paths) and routing can take advantage of the mesh structure of the network and is not limited to a tree structure. This permits better load-balancing. Figure 5.3 shows examples of possible flow paths calculated by routing.

5.3.7 Channel re-assignment problem

The problem takes as input: (i) the paths $\phi(f)\ \forall f \in F$ assigned by the routing algorithm, which determine the link load $t(v)\ \forall v \in D$ and (ii) the previous CA χ_p. The problem is to assign a channel to each undirected edge, to minimize interference, while at the same time minimizing the number of edges which must change their assignment with respect to χ_p. Obviously, there is a trade-off involved to optimize both.

To achieve the best possible optimization, the goal is to impose the minimum number of restrictions on the channel assignment. As such, the only restriction is the

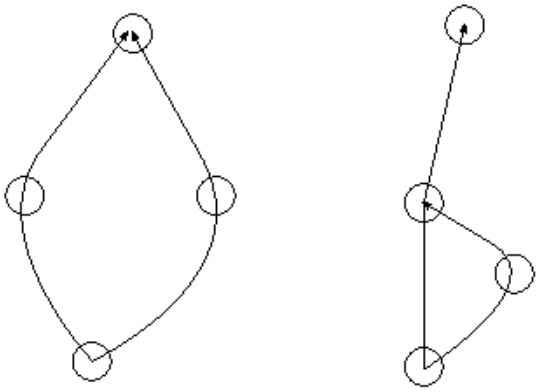


Figure 5.3: Routing is not tree-based - This figure shows examples of possible routes found by solving $\mathbf{P_{LBR}}$. Routing is not tree-based. As a result, flows of the same sink can follow different routes (a) and not all flows passing through a node have to arrive from the same node (b).

interface constraint, which determines that the number of distinct channels assigned to edges incident on a node is at most R[1]. The following multi-objective problem is proposed:

$$\mathbf{P_{CA}}: \min_{\chi}[obj_1, obj_2] \tag{5.4}$$

$$obj_1: \sum_{u \in D} \sum_{v \in D \atop \{u,v\} \in n(u)} \mathrm{t}(u)\,\mathrm{t}(v)\,\mathrm{per}(u|v) \tag{5.5}$$

$$obj_2: |\{e \in E \mid \chi_p(e) \neq \chi(e)\}| \tag{5.6}$$

subject to:

$$|\{\chi(e) \mid e \in E(n)\}| \leq R \quad \forall n \in V \tag{5.7}$$

The goal is to find the assignment $\chi : E \mapsto C$ that minimizes both objectives. Recall that the per function is defined for directed edges, that the channel assigned to a directed edge is the same as that assigned to its corresponding undirected edge and that $\mathrm{per}(u|v) = \mathrm{per}(u)$ if u and v are in different channels. The interface constraint is expressed in Equation 5.7.

[1] See section 2.5.2 for basics of channel assignment.

Accurately predicting throughput in a WMN requires complex models, due to a number of factors, including asynchronous random access, interference and dynamic wireless medium. Integrating a TCP model adds more complexity. It is difficult to use such models as objective functions for optimization problems and at the same time develop an efficient solution. This work's priority is to develop an efficient algorithm to support frequent channel re-assignment, and so we opt for an easily evaluable objective function that correlates with flow performance. Minimizing obj_1 implies assigning interfering edges with traffic to different channels. A flow is expected to share capacity with flows in the same edge and interfering edges. For this reason, it is important to avoid interference in edges with more flows, to decrease unfairness and flow starvation. Minimizing obj_1 is known to be NP-hard [112].

The second objective is to minimize the number of edges which have changed their assignment with respect to χ_p.

Note that the channel assigned to an edge without traffic does not affect the objective function, because such an edge does not generate interference. This holds true given the *current* set of flows. However, in a dynamic environment traffic conditions can change at any time and so some consideration must be given to edges which currently have no traffic. See discussion of this issue in section 5.5.

The proposed system does not require one radio on each node operating on a default common channel. All channels are available to the channel assignment procedure.

For maximum flexibility, no restriction is imposed on interface-to-neighbor binding. It was explained in section 2.5.1.1 that many decentralized channel allocation strategies rely on static bindings where a node always uses the same interface to communicate with a particular neighbor or set of neighbors. The drawback of this approach is that it can restrict channel assignment, as shown in Figure 2.5. In the proposed system there are no such restrictions, therefore bindings can change between channel assignments. A change in bindings requires updating the routing table. Section 5.7.3 will develop this further.

5.4 Load-balanced routing algorithm

This section develops a fast algorithm to solve the routing problem $\mathbf{P_{LBR}}$ described in section 5.3.6. Given the topology graph $G = (V, E)$ and set of flows F, route calculation

obtains a path for each flow.

The primary design goal for the algorithm is low time complexity, while at the same time finding good solutions. A low execution time is crucial to enable a high adaptation frequency of LBCA (see section 5.7.4). To solve the multi-objective problem $\mathbf{P_{LBR}}$, an upper bound P_θ is imposed on the cost[1] of valid paths, i.e. paths with cost higher than this bound are not considered valid. Given P_θ, for each mesh node the algorithm first obtains a set of candidate paths which can be used to reach it from the gateway. It then focuses on minimizing the first objective (balancing the load in the network).

The set of candidate paths to reach every node from the gateway is calculated only once until the topology changes. To compute these paths, the controller first solves all-pairs shortest path problem. An approach is to apply Dijkstra's algorithm (single-source shortest paths problem) on all nodes in the graph. Thus, given every pair of nodes, a list of shortest paths between them is stored. As a way of accessing the shortest paths stored, let shortestPaths(a, b) denote a function returning a list of shortest paths between nodes a and b.

The procedure to calculate candidate paths to each node is detailed in Algorithm 9. The set of candidate paths to reach a node n includes all shortest paths from the gateway to n (line 3), paths formed by concatenating all shortest paths from the gateway to nodes different from n with all shortest paths from those nodes to n (lines 4-7). Only valid paths are stored, i.e. paths with no loops, and whose cost does not exceed P_θ (lines 8-9).

We now explain the routing algorithm (LBR). It is a greedy algorithm, that iterates over the list of currently unlocked flows, in each iteration choosing the best path for the selected flow. The fitness of the solution depends on the order of flows. A heuristic is used to order them.

The algorithm is detailed in Algorithm 10. First, the list of unlocked flows is initialized and ordered (lines 1-2). For each unlocked flow, LBR chooses the candidate path with current minimum utilization (lines 3-5). The utilization of a path is defined as the maximum utilization of an edge in the path (Equation 5.8). Note that if there is more than one least used path, the one with least cost is chosen, because $cpaths(s)$

[1] The cost of a path from s to t is considered as the ratio between the length of the path (number of hops in the path) and the length of the shortest path to t. However, this work does not impose any particular metric, provided it is topology dependent.

Algorithm 9 Calculate candidate paths from gateway to every node.

```
1: function calculateCandidatePaths() do
2:   for n ∈ V − {gateway} do
3:     paths ← shortestPaths(gateway, n)
4:     for t ∈ V − {gateway, n} do
5:       for p ∈ shortestPaths(gateway, t) do
6:         for q ∈ shortestPaths(t, n) do
7:           p ← concatenate(p, q)
8:           if ¬hasLoops(p) and cost(p) ≤ P_θ then
9:             Insert p into paths
10:          end if
11:        end for
12:      end for
13:    end for
14:    Sort paths // ascending order of path cost
15:    cpaths(n) ← paths
16:  end for
17: end function
```

is ordered in ascending order of path cost. This contributes to generating paths with lower cost. After allocating a path to a flow, the utilization of edges along the path is updated (line 5).

$$\text{pathutil}(p) = \max_{e \in \mathcal{E}(p)} \{t(e)\} \tag{5.8}$$

The flow comparison function (lines 8-10) is a less-than operator that determines the order of flows. For this particular problem, we find that it is important that flows with more candidate paths be picked last, because they offer more opportunities for balancing load, and these opportunities should not be used up too early.

Let $S = \sum_{f \in F} |cpaths(sink(f))|$. Let $\bar{d}$ be the average distance from the gateway to sinks.

Theorem 3. *The time complexity of LBR in the worst case is* $O(F \log F + S\bar{d})$.

Proof. In line 2, the list of flows is sorted by a comparison sort, requiring $O(F \log F)$.

Algorithm 10 LBR: Load-balanced flow routing algorithm.

1: $flows \leftarrow list(\{f \in F \mid \neg locked(f)\})$
2: Sort $flows$
3: **for** $f \in flows$ **do**
4: $\quad \phi(f) \leftarrow \arg\min_{p \in cpaths(sink(f))} \text{pathutil}(p)$
5: $\quad \forall e \in E(\phi(f)) : t(e) \leftarrow t(e) + 1$
6: **end for**
7:
8: **function** less_than(f_1, f_2) **do**
9: $\quad$ **return** $|cpaths(sink(f_1))| < |cpaths(sink(f_2))|$
10: **end function**

The loop of lines 3-5: Line 4 requires traversing the candidate paths to the flow's sink. Because the average number of candidate paths per flow is $\frac{S}{F}$, the loop runs in

$$O(F\frac{S}{F}\bar{d}) = O(S\bar{d})$$

□

5.5 Load-aware channel assignment algorithm

This section develops an algorithm to solve the channel assignment problem $\mathbf{P_{CA}}$ described in section 5.3.7. Same as the routing algorithm, a primary design goal is low time complexity, while obtaining good solutions. A low execution time is crucial to enable a high adaptation frequency of LBCA (see section 5.7.4). In addition, the proposed algorithm also has the advantage that the execution time only depends on the topology (number of edges and neighbor degree in conflict graph). Given a network topology, the execution time can be expected to remain the same independent of traffic patterns, number of radios per node and number of channels. This can ease implementation on specific network deployments, guaranteeing that the required adaptation frequency requirements are always met.

This work proposes a fast greedy algorithm. The algorithm visits each edge and assigns it to a channel. In general, edges are visited only once, except when a merge operation is needed (see merge operation in later subsection). The quality of the solution depends on the order on which edges are visited. More critical edges are visited

Table 5.2: Structures used by channel assignment algorithm.

$c.edges = \{e \in E \mid \chi(e) = c\}$	Set of edges assigned to channel c
$v.channels = \{\chi(e)\}\ \forall e \in E(v)$	Set of channels assigned to edges of v
$v.edgesUsingChannel(c) = \{e \in E(v) \mid \chi(e) = c\}$	Set of edges of v assigned to channel c

first, i.e. edges are visited in descending order of $t(e)$. In each iteration, the algorithm assigns the current edge e to the channel where the network interference (measured by Equation 5.5) is minimized. The priority is thus minimizing obj_1, and an edge e will only be assigned to $\chi_p(e)$ if it minimizes the current value of obj_1. In the studied scenarios, edge criticality is well-defined (the edges that carry more traffic are always those closest to the gateway). This leads to the fact that a greedy algorithm is sufficient to obtain good solutions, as we will show in section 5.8.1.

Table 5.2 lists data structures used by CA. Note that it is not necessary to build these structures every time they are needed, but rather they are built and modified as the algorithm progresses. For ease of exposition we obviate showing this.

The main algorithm (LACA) is shown in Algorithm 11. In each iteration, first it builds the list of valid channels vc which can be assigned to e without violating interface constraints (lines 5-16): when assigning edge e_{mn} to a channel, if both m and n have R channels assigned, e must be assigned to a channel shared by m and n. If they don't share channels, a merge procedure needs to be executed. If one of m or n has R channels assigned, e must be assigned to a channel in that endpoint. If no endpoint has R channels assigned, all channels can be used. In this case a random order is established in vc.

The selection of a random channel when an edge can be assigned to multiple equally good channels is a simple measure that carries two important benefits. One manifests in dynamic scenarios where sudden variations of traffic can occur in any instant, which can lead to situations of groups of edges on the same channel suddenly interfering. This can occur because edges with no traffic do not produce interference and can be assigned to any channel. In addition, this measure also helps to reduce the effect of cumulative interference (see section 5.3.2). Because the interference matrix does not account for this phenomenon, by uniformly distributing channels in the network when possible, its

Algorithm 11 LACA: Load-aware channel assignment.

```
1: edges ← list(E)
2: Sort edges in descending order of t(e)
3: for e_mn ∈ edges do
4:    merge ← false
5:    if (|m.channels| = R) ∧ (|n.channels| = R) then
6:       vc ← m.channels ∩ n.channels
7:       if |vc| = 0 then
8:          merge ← true
9:       end if
10:   else if |m.channels| = R then
11:      vc ← m.channels
12:   else if |n.channels| = R then
13:      vc ← n.channels
14:   else
15:      vc ← randomOrder(C)
16:   end if
17:   if ¬merge then
18:      vc ← avoidMerge(e, vc)
19:      bestC ← arg min_{c∈vc} edgeInterference(e, c.edges)
20:      if χ_p(e) ∈ bestC then
21:         χ(e) ← χ_p(e)
22:      else
23:         χ(e) ← First element of bestC
24:      end if
25:   else
26:      mergeop(e)
27:   end if
28: end for
29:
30: function edgeInterference(u, V) do
31:   Δ ← ⋃_{v∈V} dir(v)
32:   return Σ_{u⃗∈dir(u)} Σ_{v⃗∈Δ} (t(u⃗)t(v⃗) per(u⃗|v⃗) + t(u⃗)t(v⃗) per(v⃗|u⃗))
33: end function
```

effect can be reduced.

The next step involves refining the list of valid channels to attempt to avoid future merge operations (explained later), and assigning the edge to the channel in vc where less interference is produced (lines 17-24). LACA attempts to maintain the same channel previously assigned to e, if possible (lines 20-21). The edgeInterference(u, V) function determines the increase in interference when the undirected edge u is in the same channel as the set of edges V. More specifically, the set Δ contains the directed edges corresponding to the undirected edges in V (line 31). The interference in directed edges of u by directed edges of Δ and vice versa is calculated in line 32. Note that, although not shown for ease of presentation, interference of u is only checked against its adjacent nodes in the conflict graph (i.e. against edges which can interfere with u). Also note there is no need to check interference between a directed edge and its inverse edge, because they will always be assigned to the same channel regardless.

The basic structure of the greedy algorithm is similar to the one proposed by Raniwala et al. in [103]. However, LACA optimizes a different objective function and network interference is measured differently. In addition, the merge operation explained in the following subsection has not been proposed before, and this work introduces a novel procedure to avoid merge operations. Lastly, their algorithm is intended for static scenarios, where the traffic profile is expected to remain stable over long periods of time, and does not consider the issue of channel re-assignment.

5.5.1 Merge operation

The channel assignment algorithm needs to perform a merge operation when assigning a channel to edge e_{mn} and both m and n have already been assigned R channels and don't have a channel in common, i.e. $|m.channels| = R \ \wedge \ |n.channels| = R \ \wedge \ m.channels \cap n.channels = \emptyset$. In this situation, a channel cannot be assigned to e without breaking interface constraints.

This condition can only occur in cases where both m and n have less radios than their node degree, the number of radios is less than the number of channels, and the algorithm happens to have previously assigned different channels to m and n. In other words, the likelihood of this occurring depends on the particular scenario: topology (node degree), number of radios in each node and number of channels.

The goal of the merge operation is for m and n to have a channel in common, and to assign e_{mn} to the common channel. To do this, a channel c_1 from one of the endpoints is converted to a channel c_2 from the other endpoint, and c_2 is assigned to e. However, changing the channel c_1 of edges of one endpoint may involve propagating the changes through the graph. To illustrate this, consider the example in Figure 5.4. The algorithm wants to assign a channel to edge e. Nodes m and n have already been assigned 3 channels and don't have a channel in common. Suppose we want to reassign edges of m in channel C to a channel in n, say F. Note that this change affects node z, which must also reassign all its edges in C to F. And, for the same reason, this change in turn might involve successive propagation through the graph.

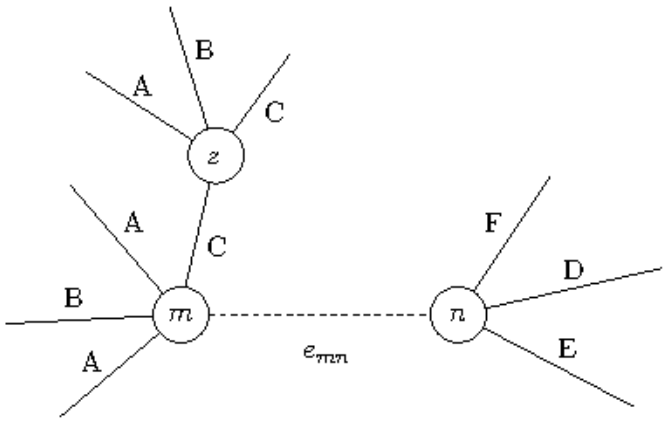


Figure 5.4: Merge operation example - This figure shows an example of when a merge operation is needed. Nodes have 3 radio interfaces, and there are 6 channels {A,B,C,D,E,F}. A channel needs to be assigned to edge e_{mn}.

The end result is that a set of edges in c_1 need to be reassigned to a different channel c_2, and the algorithm must choose the best pair (c_1, c_2) such that the overall interference (objective function) is minimized. The merge operation transforms the channel of one endpoint to a channel of the other endpoint. There are a total of $2(R \times R)$ possible transformations, given by the Cartesian products of the set of channels in each endpoint, i.e. $m.channels \times n.channels$ and $n.channels \times m.channels$, where each ordered pair (c_1, c_2) denotes that channel c_1 is transformed into c_2. Each one of these transformations can require changing the channel of several edges in the graph. The merge operation tests every one of these pairs and chooses the one that produces least interference.

The algorithm is detailed in Algorithm 12. For each one of these pairs (lines 2-3), the algorithm first calculates the set of edges E_x that have to be reassigned from c_1 to c_2 (lines 4-22). To this end, it first inserts into a queue the endpoint of e which has c_1 assigned to it (lines 7-10). Nodes in the queue will be processed to determine if their edges in c_1 have to change to c_2. In the case of node m or n, all its edges in c_1 have to be reassigned. Next, the algorithm checks if these changes have to be propagated to other nodes. To this end, the endpoints of the edges marked for modification are inserted into the queue. When extracting a node from the queue, the following rule determines if the rest of its edges in c_1 have to be reassigned to c_2: the node does not have free interfaces, c_2 is not assigned to the node, and the node has more than one edge in c_1 (line 14). Endpoints of edges marked for modification are inserted into the queue to continue propagating the changes and the process repeats until no more edges are added.

Once the set of edges E_x to change from c_1 to c_2 is obtained, the algorithm calculates the modification in network interference (lines 24-29). Edge e will interfere with all its neighbor edges in c_2 (line 25). This includes the edges in E_x. And edges which were previously in channel c_1 will no longer interfere with edges in this channel but will interfere with their neighbor edges in c_2 (lines 26-28).

Finally, the algorithm chooses the transformation that produces least interference, and switches affected edges in c_1 to c_2 (lines 30-37).

As we will show, the simple rule used to determine if a channel switch has to be propagated through the graph (line 14), when integrated in the merge procedure of the Tabu-based algorithm in [112] (which is a merge procedure different to the one proposed in this work), improves the solution of their algorithm.

5.5.2 Avoiding merge operations

The merge operation can prove costly due to having to test, for each possible transformation (c_1, c_2), the interference of every propagation of changes through the graph. The step explained in this subsection is optional and is performed in order to reduce the likelihood of having to execute merge operations. As such, it can reduce the execution time of the algorithm, while maintaining a similar solution. This is evaluated empirically in section 5.8.1.

Algorithm 12 Merge operation initiated in edge e_{mn}.

```
1: bestI ← ∞
2: pairs ← (m.channels × n.channels) ∪ (n.channels × m.channels)
3: for (c1, c2) ∈ pairs do // test converting c1 to c2
4:    // propagation effect
5:    Ex ← ∅ // edges to change from c1 to c2
6:    queue ← FifoQueue()
7:    if c1 ∈ m.channels then
8:       queue.append(m)
9:    else
10:      queue.append(n)
11:   end if
12:   while queue¬empty do
13:      x ← queue.pop()
14:      if (x ∈ {m, n}) ∨ ((R |x.channels| = 0) ∧ (c2 ∉ x.channels) ∧
         (|x.edgesUsingChannel(c1)| > 1)) then
15:         for uxy ∈ x.edgesUsingChannel(c1) do
16:            if u ∉ Ex then
17:               Ex ← Ex ∪ {u}
18:               Insert y into queue
19:            end if
20:         end for
21:      end if
22:   end while
23:
24:   // measure effect of merge operation
25:   I ← edgeInterference(e, Ex ∪ c2.edges)
26:   for u ∈ Ex do
27:      I ← I  edgeInterference(u, c1.edges  Ex)
28:      I ← I + edgeInterference(u, c2.edges)
29:   end for
30:   if I < bestI then
31:      bestI ← I
32:      bestEx ← Ex ∪ {e}
33:      bestC ← c2
34:   end if
35: end for
36: // perform merge operation
37: χ(u) ← bestC  ∀u ∈ bestEx
```

First, let's call *critical* neighbors of a node the set of neighbors such that the edge that connects the node to those neighbors has not yet been assigned to a channel, the neighbors have already been assigned R channels and currently the node does not have a channel in common with them. The basic idea is that when first assigning a channel to an edge e_{mn}, the heuristic gives priority to assigning one that will permit m and n to have a channel in common with their *critical* neighbors.

The heuristic is detailed in Algorithm 13. It receives as input the current edge e, and the set of valid channels vc which can be assigned to e, determined by the interface constraint and calculated previously (see lines 5-16 of Alg. 11). The purpose of this algorithm is to further reduce vc to favor choosing channels which permit m and n to communicate with their critical neighbors, when necessary.

A channel is said to cover a node n if the channel is in $n.channels$. The following function returns the number of nodes in a set N that are covered by a channel c:

$$coveredNodes(c, N) = |\{n \in N \mid c \in n.channels\}| \tag{5.9}$$

The first step is to estimate the minimum number of channels the endpoints of e need to communicate with their critical neighbors. For each endpoint x, the algorithm initializes the number of free interfaces fi and the list of critical neighbors. It then proceeds by selecting channels greedily (in each step selecting the channel which covers most critical neighbors), until all critical neighbors are covered. The size of the resulting set determines the minimum number of channels needed. Note that this heuristic does not guarantee optimality, and therefore only represents an estimate of the minimum number of channels needed.

After the minimum number of channels mc needed by each endpoint is calculated, the algorithm does one of the following:

- If mc of both endpoints equals their number of free interfaces fi, it forces the choice of a channel belonging to their set of critical neighbors. Channels which cover more neighbors are given priority.

- If mc of one endpoint equals its number of free interfaces fi, it forces the choice of a channel belonging to its set of critical neighbors. Channels which cover more neighbors are given priority.

Algorithm 13 Avoid merge heuristic (initiated in edge e_{mn}).

```
1: for x ∈ {m, n} do
2:   x.fi ← (R  |x.channels|)
3:   x.critNbs ← {nb ∈ x.neighbors : |nb.channels| = R ∧ (nb.channels ∩
     x.channels = ∅) ∧ nb ∉ {m, n}}
4:   if |x.critNbs| > 0 then
5:     x.mc ← 0
6:     x.critChannels ← (⋃_{nb∈x.critNbs} nb.channels) ∩ vc
7:     critNbs ← copy(x.critNbs)
8:     critChannels ← copy(x.critChannels)
9:     while |critNbs| > 0 ∧ |critChannels| > 0 do
10:      bestChannel ← arg max_{c∈critChannels} coveredNodes(c, critNbs)
11:      x.mc ← x.mc + 1
12:      critNbs ← critNbs  {nb ∈ critNbs : bestChannel ∈ nb.channels}
13:      critChannels ← critChannels  {bestChannel}
14:    end while
15:    if (|critNbs| > 0) ∨ (x.mc > x.fi) then
16:      return vc // Cannot cover all critical neighbors
17:    end if
18:  end if
19: end for
20:
21: if (|m.critNbs| > 0) ∧ (|n.critNbs| > 0) ∧ (m.mc = m.fi) ∧ (n.mc = n.fi) then
22:   X ← m.critChannels ∪ n.critChannels
23:   Sort X in descending order of coveredNodes(c, m.critNbs ∪ n.critNbs)
24:   return X
25: else if (|m.critNbs| > 0) ∧ (m.mc = m.fi) then
26:   Sort m.critChannels in descending order of coveredNodes(c, m.critNbs)
27:   return m.critChannels
28: else if (|n.critNbs| > 0) ∧ (n.mc = n.fi) then
29:   Sort n.critChannels in descending order of coveredNodes(c, n.critNbs)
30:   return n.critChannels
31: else
32:   return vc
33: end if
```

- If the endpoints have more free interfaces than mc, no restriction is enforced on the candidate channels.

We recommend implementing this procedure in scenarios where merge operations are likely (as explained earlier, this depends on node degree, number of radios in each node and number of channels).

5.6 Network architecture and capacity scaling

The basic architecture described in this chapter is that of a WMN defined by a connected undirected graph, with a gateway node providing access to the Internet. Transmissions to the mesh nodes occur under the same physical layer technology and frequency ranges. Beyond this, the proposed routing and channel assignment protocol doesn't make any further assumptions on the underlying network architecture, nor does it require a specific architecture to function.

However, the specific WMN architecture used can have a substantial effect on network capacity, because it determines the achievable peak network throughput as well as the load-balancing possibilities. To both provide higher capacity and permit the load-balancing protocol to utilize this capacity, this work proposes the following architecture. Protocol evaluations will be conducted in the architecture described in this section.

Consider the basic architecture described until now. A major limitation is the capacity of the region surrounding the gateway. All connections to the Internet pass through the gateway, turning the edges incident on the gateway into a bottleneck. Let $link\_rate$ denote the maximum rate of WMN links. The maximum aggregate throughput achievable by Internet flows is:

$$T_1 = link\_rate \times |E(gateway)| \tag{5.10}$$

That is, the maximum throughput is the aggregate capacity of the edges incident on the gateway, assuming no other edge interferes with a gateway edge (including other gateway edges). This holds only if:

$$UL(gateway) \geq T_1$$

where $UL(x)$ denotes the capacity of the uplink of node x (link used by x to access the Internet). It is reasonable to assume that the gateway's uplink uses a high speed wired connection capable of supporting this bandwidth.

In this architecture, it is possible to increase the maximum achievable throughput and balancing options by increasing the number of neighbors of the gateway (and consequently the number of gateway edges). However, because each gateway edge must be free of interference to achieve peak capacity, this implies that the channel used by a gateway edge cannot be used by another edge in its interference range. A large number of channels are needed to obtain such a CA.

To address this issue, a new type of node is introduced, called *ring-node*. These nodes are positioned around the gateway. The gateway is connected to each using a high speed connection which does not interfere with regular WMN transmissions. There are many technologies (wired or wireless) to achieve this (e.g. WiMAX). In the case of wireless transmission, the use of a different frequency range or directional antennas can help to achieve this. This architecture is illustrated in Figure 5.5. This effectively creates multiple spread Internet access points, providing increased balancing options. The rest of the mesh nodes are located outside the gateway and ring-node zone.

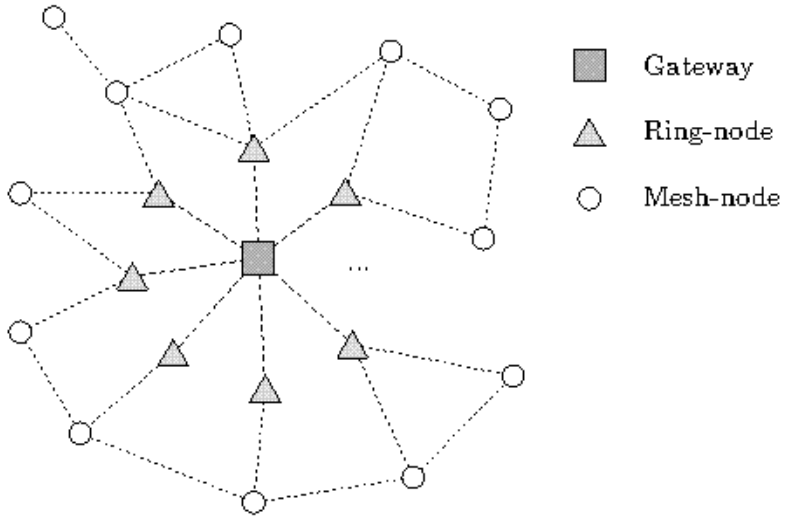


Figure 5.5: WMN ring-node architecture - Links between the gateway and ring-nodes do not interfere with regular WMN transmissions (i.e. with mesh node links).

Let RN be the set of ring-nodes. The maximum throughput achievable by flows is

now:

$$T_2 = link\_rate \times \sum_{r \in RN} (|E(r)| \quad 1) \tag{5.11}$$

where $|E(r)| \quad 1$ is the number of downlinks of a ring-node. Note that $E(r)$ includes the uplink of the ring-node (i.e. gateway-ringnode edge). The above holds only if:

$$UL(gateway) \geq T_2$$
$$UL(r) \geq link\_rate \times (|E(r)| \quad 1) \quad \forall r \in RN$$

A ring-node to gateway connection $UL(r)$ must have more bandwidth than the combined bandwidth of the ring-node's downlinks. Likewise, the gateway uplink bandwidth must be greater than the combined bandwidth of every ring-node downlink. We assume this is true. Same as before, the necessary gateway uplink bandwidth can be supported by a high speed wired connection. The necessary bandwidth between the gateway and ring-nodes can be supported by a wired connection or high speed wireless connection such as the future WiMAX standard.

To achieve T_2, each edge incident on a ring-node must be free of interference. This is easier to achieve than the previous case of edges incident on the gateway. Separation between ring-nodes facilitates reusing channels assigned to ring-node edges in other parts of the network, including other ring-node edges.

5.7 Load balancing and channel assignment (LBCA) protocol

This section explains the proposed Load-Balancing and Channel Assignment (LBCA) protocol. The gateway node, called the *controller*, is responsible for route and CA calculation. Routes and CA are calculated periodically every α_T seconds (*adaptation period*) based on the current network state. The evaluations in this chapter have used an adaptation frequency as high as once per second ($\alpha_T = 1$). Every time routes are calculated and channel allocation changes, these changes must be applied.

The rest of the section explains how the controller obtains the necessary information to calculate a solution and how to apply the solution in the network.

5.7.1 Network state monitoring

To calculate and apply a routing and CA solution, the controller needs the following information:

1. WMN topology graph $G = (V, E)$ and link costs.
2. Interference matrix $\mathbf{IM}$, where $\mathbf{IM}_{u,v} = \text{per}(u|v)\ \forall u, v \in D$.
3. Spanning tree T of the topology graph.
4. Knowledge of active flows F.

Items 1 and 4 are needed by the routing algorithm (LBR). 1, 2 and 4 are used by the channel assignment algorithm (LACA). Item 3 is used for distributing the CA throughout the network.

It is assumed that the controller obtains the topology graph and link costs from information received periodically from a standard WMN routing protocol (link-state protocols like OLSR -described in section 2.4.2.3- provide this information). Note that the frequency used by the routing protocol to send this information is independent of α_T. In addition, as explained in section 5.3.6, the cost metric must be topology-dependent. Due to the static nature of the WMN topology, the update frequency of the routing protocol is expected to be low.

In section 2.2.3.3 it was explained that the only reliable method of estimating packet delivery and interference in MWNs is to measure it. Taking this into account, we assume the use of a technique which measures $\text{per}(u)$ and $\text{per}(u|v)\ \forall u, v \in D$. Measurement techniques have been proposed in [90, 91, 104].

A spanning tree of G is built in the network, rooted at the gateway. As long as the topology does not change the tree remains stable, and so is expected to remain static over long periods of time (compared to α_T). The structure of the tree is not particularly important as long as it is not extremely unbalanced. A simple method is to build the tree where each node chooses as parent the neighbor closest to the gateway (in terms of hop-count). The tree is used only for the purposes of channel assignment distribution (explained later), not for routing. Because the tree is expected to remain stable for long periods of time and because the exact tree structure is not important,

this work does not focus on developing a procedure and protocol for tree formation (see 802.1D [4] for a well-known spanning tree formation protocol).

Internet TCP traffic is classified into flows. Traffic classification is performed by the controller when traffic passes through it. A system such as IPFIX [19] can be used for this. Note that these systems employ two basic mechanisms to determine if a flow has finished: flow aging (no packets have been observed for a specified time) and detection of TCP session termination messages. The controller will have knowledge of all flows but, obviously, flows with a duration shorter than α_T are not guaranteed to be taken into account by the protocol. This does not pose a problem when α_T is very small, because the flows which are not taken into account will be very small-sized flows, which have negligible repercussions on the load and imbalance of network traffic.

We note that the overhead of maintaining up-to-date network state in this system is not high. The set of current flows can be maintained at all times without introducing any overhead in the network, because traffic classification and book-keeping is done at the controller in real-time. The overhead of maintaining an up-to-date topology graph and interference matrix at the controller mainly depends on the frequency with which this information changes. In the case of the topology, it is expected to remain stable over long periods of time (hours or days) and thus the frequency with which the WMN routing protocol needs to update it will be low. Similarly, interference measurements are not expected to be conducted frequently, due to the static nature of the network (set of mesh nodes, their location and radio interface parameters) and because these measurements are lengthy in time and likely require disrupting all network activity [91].

5.7.2 Routing

Every α_T seconds the controller calculates routes for every current flow $f \in F$ using the LBR algorithm (section 5.4).

It is important to note that routing from the gateway to the mesh nodes and vice versa takes advantage of the mesh structure of the network and does not form a tree structure. For example, flows of the same sink can follow different paths. This permits better load-balancing.

To enable each flow to follow the path calculated by the routing algorithm, the controller uses source routing. The controller inserts in every packet of download traffic of a flow the route to reach the sink (as a sequence of node addresses), and intermediate

nodes in the WMN route the packet based on this route. Routing changes are therefore applied instantaneously at the gateway. Upload traffic of a flow follows the reverse route. To do this, for each ongoing flow, a sink merely extracts the source route used by download traffic, reverses it and inserts the reverse route in packets of upload traffic.

When a route is chosen for a flow, the flow is locked for $Lock_l$ seconds. The routing algorithm will not re-route locked flows. This is done to prevent frequent oscillation of a flow's route, which can prove detrimental to the performance of TCP due to issues like packet reordering and variations in round-trip time.

5.7.3 CA distribution protocol

Immediately following route calculation, the controller calculates a CA using the LACA algorithm (section 5.5). The new CA must be communicated to the affected nodes in the WMN so that they can update their bindings, i.e. switch the channel of their interfaces and update interface-to-neighbor binding (interface bindings were discussed in section 2.5.1). The proposed distribution protocol is simple and efficient. It minimizes overhead and enables the CA to reach every node in a short time.

The output of LACA is the mapping $\chi : E \mapsto C$. This output is sent verbatim to WMN nodes via Channel Assignment Messages (CAM). A CAM contains a list of edges and their assigned channel. More specifically, the CAM is a sequence of (e, c) pairs, where $e \in E$ and $c \in C$. Note that a CAM will only contain edges whose channel has changed since the previous channel assignment χ_p and that affect nodes of the subtree to which the CAM is directed. Because LACA limits the number of channel re-adjustments, the size and number of CAMs transmissions is reduced.

CAM messages should have a higher transmission priority than other packets. If the WMN uses a 802.11 stack, this work proposes using 802.11e to send CAM messages with the highest priority class.

Every child of the gateway in the spanning tree T is the root of a subtree of T. Let $subtree(n)$ be the set of nodes in the subtree rooted at node n. The gateway sends one CAM to each child subtree. For the CAM directed to a subtree, the gateway includes an edge in the CAM only if the channel assigned to the edge has changed since χ_p and if an endpoint of the edge is in the subtree. This is detailed in Algorithm 14.

Every node has a local version of the CA map χ, storing the channel assigned to each of the node's edges. When a node receives a CAM, it updates the channel allocation of

Algorithm 14 Controller sends CAMs.

1: **for** $ch \in children(gateway)$ **do**
2: $CAM \leftarrow \emptyset$
3: $CAM.seqnum \leftarrow currentSeqNum$
4: **for** $e_{mn} \in E$ **do**
5: **if** $\{m, n\} \cap subtree(ch) \neq \emptyset$ **and** $\chi(e) \neq \chi_p(e)$ **then**
6: Insert $(e, \chi(e))$ into CAM
7: **end if**
8: **end for**
9: Send CAM to ch
10: **end for**
11: $currentSeqNum \leftarrow currentSeqNum + 1$

Table 5.3: CAM distribution notation.

$CAM.seqnum$	Sequence number of CAM
$i.oldNbs$	Previous set of neighbors reachable via interface i
$i.newNbs$	New set of neighbors reachable via interface i
$i.oldChannel$	Previous channel assigned to interface i
$i.newChannel$	New channel assigned to interface i

each of its interfaces, updates interface-to-neighbor binding, and forwards a trimmed copy of the CAM to each child. This procedure is detailed in Algorithms 15 and 16. Table 5.3 lists notation used in these algorithms.

First, the node checks the CAM and discards it if it is duplicate (lines 1-4 of Alg. 15). After data initialization, the node updates its local CA map with the information contained in the CAM (lines 15-20). Note that the node's edges which are not present in the CAM don't change channel.

For each interface, the procedure updates the channel to which the interface is tuned, its interface-to-neighbor binding, and sends the CAM to the children currently reachable via the interface. This is detailed in Alg. 16. When assigning a channel to an interface, if possible, the algorithm retains the current channel of the interface to avoid channel switching (lines 2-9). Channel switching incurs a delay; however small this delay, it is best to avoid it. As explained previously, a channel reassignment can alter the set of neighbors of an interface. The algorithm updates the set of neighbors

Algorithm 15 Node n receives a CAM.

```
1: // check if duplicate CAM
2: if CAM.seqnum ≤ lastCAMSeqnum then
3:    return // is duplicate CAM
4: end if
5: lastCAMSeqnum ← CAM.seqnum
6:
7: // interface initialization
8: for i ∈ R(n) do
9:    i.oldNbs ← i.newNbs
10:   i.newNbs ← ∅
11:   i.oldChannel ← i.newChannel
12:   i.newChannel ← ∅
13: end for
14:
15: // update my CA mapping
16: for (e, c) ∈ CAM do
17:    if n ∈ e then
18:       χ(e) ← c
19:    end if
20: end for
21:
22: oldChannels ← {i.oldChannel}   ∀i ∈ R(n)
23: newChannels ← {χ(e)}   ∀e ∈ E(n)
24: for i ∈ R(n) do
25:    updateInterfaceBindings(i, oldChannels, newChannels)
26: end for
```

Algorithm 16 Update interface bindings of interface i on node n and forward CAM.

```
1: function updateInterfaceBindings(i, oldChannels, newChannels) do
2:   // assign channel and neighbors to interface i
3:   if i.oldChannel ∈ newChannels then
4:     c ← i.oldChannel
5:   else
6:     c ← element from (newChannels   oldChannels)
7:   end if
8:   i.newChannel ← c
9:   newChannels ← newChannels   {c}
10:  for {e_mn ∈ E(n) | χ(e_mn) = c} do
11:    i.newNbs ← i.newNbs ∪ {m}
12:  end for
13:
14:  // send CAM and switch channel
15:  ichildren ← {nb ∈ i.oldNbs | nb ∈ children(n)}
16:  if |ichildren| = 0 then
17:    if i.oldChannel ≠ i.newChannel then
18:      Switch channel of i to i.newChannel
19:    end if
20:  else
21:    pendingCAM ← 0
22:    for ch ∈ ichildren do
23:      newCAM ← ∅
24:      for (e_ab, c) ∈ CAM do
25:        if {a, b} ∩ subtree(ch) ≠ ∅ then
26:          Include (e, c) in newCAM
27:        end if
28:      end for
29:      if newCAM ≠ ∅ then
30:        pendingCAM ← pendingCAM + 1
31:        Send newCAM to ch
32:      end if
33:    end for
34:  end if
35:
36:  // update routing (interface-to-neighbor binding)
37:  for nb ∈ i.newNbs do
38:    Update route to nb via i
39:  end for
40: end function
```

in lines 10-12.

Before performing interface channel switching, the node must forward the CAM to its children currently reachable via the interface (or else network connectivity would break). If no children are reachable and the channel has changed, channel switching occurs immediately (lines 16-18). If children are reachable, a trimmed copy of the CAM is sent to each (lines 20-34). The CAM only contains edges which have an endpoint in the child subtree. If the interface channel has changed, the interface will switch only after every child receives the CAM ($pendingCAM$ reaches zero). Finally, the node updates the interface-to-neighbor binding of the interface (lines 37-39).

The maximum number of transmissions in a CA distribution is $|V|$ 1 (the number of edges of the tree). In practice, however, LACA is effective at reducing the number of channel re-adjustments, as we will show in section 5.8.2. This reduces both the size of CAM messages as well as the number of CAM transmissions, and avoids channel switching delays.

5.7.4 LBCA adaptation frequency

In practice, scenarios are dynamic and the performance obtained by the protocol will thus largely depend on its responsiveness. The solution calculated at a particular instance is tailored for a specific traffic profile (set of flows) and network conditions, and if these conditions or the traffic changes the solution will no longer remain valid or optimal. The responsiveness of the protocol depends on α_T. The lower the adaptation period the more quickly it can react to changes in network conditions.

The adaptation period α_T cannot be lower than the time required to calculate and apply a solution. In practice, this time is very small (e.g. in simulations is less than one second so the parameter has been set to $\alpha_T = 1$). The reasons for this are: (a) the route and channel assignment algorithms have low complexity; (b) new routes are applied instantaneously at the controller simply by adding the route to new packets; (c) the CA distribution is efficient, requires few transmissions, CAMs are small, and by assigning high priority to CAMs it is possible to distribute them quickly.

Obviously, for the solution to be valid, the network state must be up-to-date at the controller. We assume this state is updated with the necessary frequency to track any relevant changes.

Table 5.4: Physical wireless parameters used in evaluations.

Wifi standard	802.11b @ 11mbps constant rate
Path loss model	Log-distance, exponent = 2.7
RTS/CTS (*in ns-3*)	Disabled
Transmit power	$\min\{p \mid \text{per}(e) \leq 5\%\} \quad \forall e \in D$
Energy detection threshold (EDth) (W)	RxPower(max neighbor distance)
Carrier sense threshold (CSth) (W)	EDth $\times$ 10 $^{9/10}$

5.8 Performance evaluations

This section evaluates the performance of LBCA in gateway access scenarios from both a graph-theoretic perspective and through simulation in *ns-3* [3].

In all cases, the WMN topologies used are the same. They consist of static networks in a square 1000 m $\times$ 1000 m area. The network architecture is explained in section 5.6. The gateway is located in the center, surrounded by 8 ring-nodes. The distance from the gateway to ring-nodes is 120 m. There are 70 mesh nodes placed randomly outside the gateway-ringnode zone. All topologies are connected, with a minimum distance of 100 m between mesh nodes. Maximum transmit range between mesh nodes is 150 m, i.e. an undirected edge exists between mesh nodes when the distance is less than or equal to 150 m (transmit power is adjusted to achieve this).

All tests use the physical model of *ns-3* (Yans[1]) [3, 65]. The parameters used to configure wireless interfaces and propagation are shown in Table 5.4. The transmit power chosen is the minimum power that guarantees a Packet Error Rate (PER) of at most 5% for all links in the network (with no concurrent transmissions). The energy detection threshold is the received power at the maximum link distance. Carrier sense threshold is derived based on a ratio of $CSth/EDth$ of 9 dB. This ratio is chosen because it produced maximum aggregate throughput in numerous simulation tests.

Given a network topology (node positions), the Yans physical model, and the parameters listed in Table 5.4, the Interference Matrix **IM** is calculated for packet sizes of

[1] In the Yans physical model of *ns 3* successful reception depends on the BER. The BER depends on the SINR and modulation used. The SINR is calculated based on signal strength, noise and cumulative interference.

2048 bytes (this is a conservative measure because the larger the packet size the larger the PER).

Two types of routing strategies are used in the evaluations: shortest path routing (based on hop count) and load-balanced routing using the LBR algorithm. With shortest path routing, the path assigned to a flow is the shortest path that connects the gateway to the flow's sink.

5.8.1 Evaluation of channel assignment algorithm

This section analyzes the performance of LACA based on the value of the objective function obj_1 (Equation 5.5). Given a network graph G, its interference matrix $\mathbf{IM}_{u,v} = \text{per}(u|v)\ \forall u, v \in D$, and a set of flows, the tests calculate routes using the LBR algorithm, and subsequently apply and compare different CA algorithms.

Traffic is considered to be based on TCP flows. Random Internet flows are generated (one endpoint is the gateway, another is a mesh node). The number of concurrent flows in a scenario can be one of {10,20,30,40,50}. For each number of flows, 100 random scenarios are generated, totaling 500 unique scenarios per topology. In each scenario, a flow's sink is chosen randomly (a sink can be selected multiple times).

Tests have been performed with $R = 3$ and $R = 6$. The maximum number of usable interfaces per node in these scenarios is six because the maximum node degree is six. Note that when $R = 6$ no merge operations are needed (no node has less radios than its node degree).

The tests compare LACA with the Tabu-based algorithm of Subramanian et al. [112] (we refer to it as TABU)[1]. Two variants of LACA are used: one using the avoid merge procedure (LACA_AM) and one without it (LACA_NOAM). Two variants of TABU are tested: the original one (TABU_ORIG) and a modification which adapts the propagation rule of LACA's merge procedure (line 14 of Alg. 12) to the merge procedure of TABU (we refer to this variant as TABU_MODIF). Parameters chosen for TABU are $i_{max} = |E|$ and $r = 20$. All implementations are in the Python language.

Performance results are shown in Figure 5.6. Each point is the average result of the scenarios that fall in that class. Figs. 5.6 (a) and (c) show the value of the objective of the CA problem (Equation 5.5) while Figs. 5.6 (b) and (d) show the time to calculate a

[1] The mathematical form of the traffic aware objective function optimized by TABU is equivalent to the one optimized by LACA.

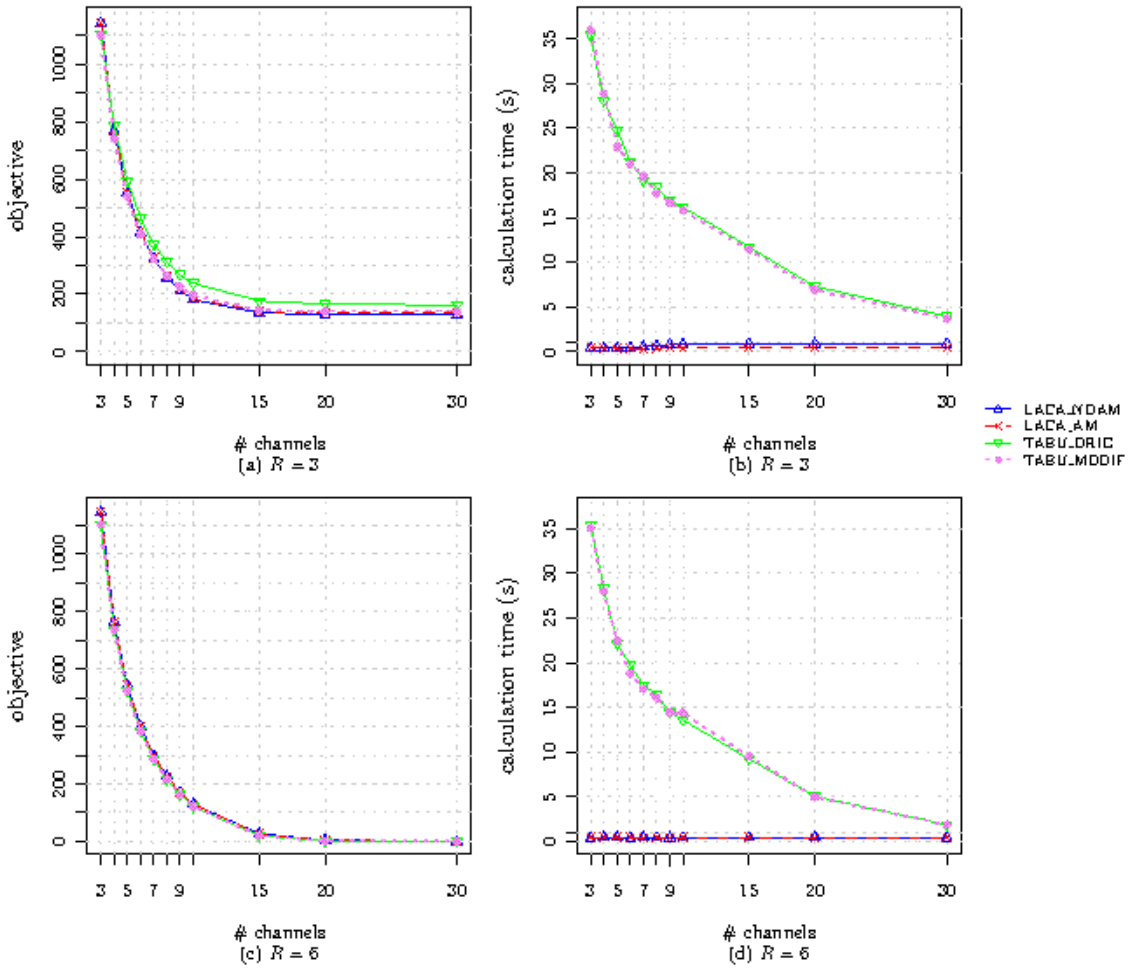


Figure 5.6: LACA performance results - This figure compares the performance of LACA and TABU from a graph-theoretic perspective.

Table 5.5: LACA and TABU execution time.

	Average time (ms)	
	$R = 3$	$R = 6$
LACA_NOAM	652	436
LACA_AM	404	449
TABU_ORIG	18486	16710

solution. A number of conclusions can be extracted. First, the solution found by LACA is very similar to the one found by TABU, and in most cases better than the original TABU when merge operations are needed ($R = 3$). This is due to inefficiencies in the merge operation of TABU. Note that the proposed modification of TABU improves this. The main advantage of LACA over TABU is that it is substantially faster. As stated earlier, this is crucial to enable a high adaptation frequency of LBCA. The execution time of TABU is several order of magnitudes higher to that of LACA (see table 5.5 for details). Also, the execution time of TABU suffers considerably with lower number of channels. We have found that in these cases, due to the nature of the TABU algorithm, the interference of edges is repeatedly tested against large groups of edges, because the probability of edges residing in the same channel is much higher. In contrast, the execution time of LACA is fairly constant independently of traffic patterns, number of radios and channels.

As explained previously, minimizing the objective function will obviously lead to increasing network performance (due to assigning interfering edges with traffic to different channels). This can also be seen graphically in the results of this section and the next. An important observation is the curve depicting logarithmic decrease in the objective value as the number of channels increases. A similar curve showing logarithmic increase in throughput with the number of channels will be seen in simulations.

Also note that, without sufficient interfaces, the objective can never reach zero regardless of how many channels are available. With sufficient number of interfaces and channels, however, the objective can reach zero. In this case the performance will match that of a interference-free network (this is shown in the simulation results of the next section). It is important to note that this can be achieved with much less channels than number of undirected edges, and therefore it is not necessary to assign each edge

to a unique channel to achieve interference-free performance (in the scenarios tested the average number of undirected edges is 146).

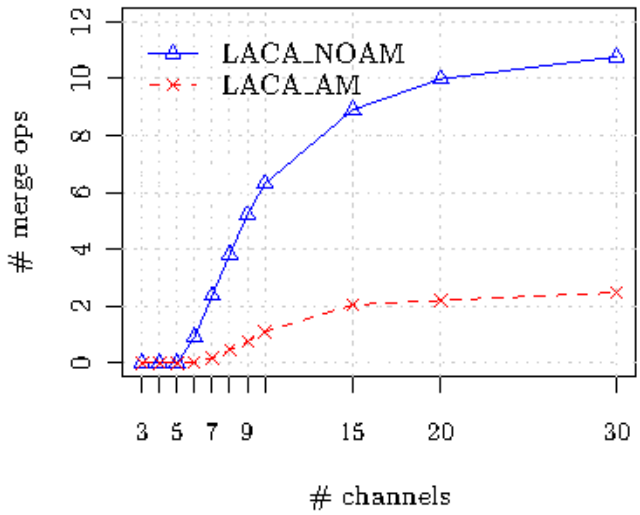


Figure 5.7: Merge operations executed by LACA - This figure shows the number of merge operations performed by LACA_NOAM and LACA_AM when $R = 3$.

Finally, Figure 5.7 shows the number of merge operations performed by LACA when $R = 3$. Each point shows the average number of times a merge operation (Algorithm 12) is performed during a execution of LACA. As we can see, the avoid merge process effectively reduces the number of merge operations, and thus the execution time of the algorithm (see table 5.5). And this does not affect the quality of the solution.

5.8.2 Channel re-adjustments

This section studies the number of affected edges by channel re-assignment of LACA, when the traffic pattern of a network randomly varies in time. Given a network graph G, we generate a list $L = \{t_1, t_2, ..., t_{100}\}$ of 100 random traffic patterns. Each pattern consists of 50 random download flows. For each pattern t_i a CA is calculated, using as previous CA the one calculated for t_{i-1}. The tests evaluate the objective function obj_2 (Equation 5.6).

Tables 5.6 and 5.7 show the average percentage of edges that need to re-adjust their channel in each re-assignment, when using shortest path routing and LBR, respectively.

Table 5.6: Number of channel re-adjustments with shortest path routing.

		Channels											
		2	3	4	5	6	7	8	9	10	15	20	30
% edges	$R = 3$	13	17	19	18	16	14	14	12	11	6	5	3
	$R = 6$	12	16	18	15	14	12	10	8	5	0	0	0

Table 5.7: Number of channel re-adjustments using LBR.

		Channels											
		2	3	4	5	6	7	8	9	10	15	20	30
% edges	$R = 3$	27	36	42	47	48	48	49	47	45	40	38	35
	$R = 6$	26	36	39	40	40	39	38	36	34	21	11	2

This percentage is calculated as $100 \times \frac{\overline{obj_2}}{\mathcal{E}}$ where $\overline{obj_2}$ is the average value of obj_2 over all runs.

When using shortest path routing, only a small percentage of edges are affected (less than 20%), while the number of affected edges in these scenarios with LBR varies roughly from 25% to 50% of the edges, depending on the number of radios and channels. In all cases, with less radios per node, more channel re-adjustments are necessary. The reason why shortest path routing requires less channel re-adjustments is that this routing strategy utilizes substantially less links than load-balanced routing; this strategy does not balance load, and so traffic generally traverses the same links. This also leads to the fact that shortest path routing can reach interference-free performance with less channels than LBR, as we will show and explain in the next section.

5.8.3 Simulation results

In this section, the performance of LBCA is evaluated by simulation in *ns-3*. Tests are implemented as described below.

5.8.3.1 Simulation environment

The WMN architecture is explained in section 5.6 and has one gateway. There is a server on the Internet connected to the gateway by a point-to-point 1000 Mbps link. Similarly,

the connection from the gateway to each ring node is implemented as a point-to-point 100 Mbps link. Users connect to the Internet server, which sends data to them. The gateway runs LBCA. It knows the complete topology, active flows, Interference Matrix **IM** and spanning tree (see section 5.7.1). The **IM** is calculated once at the start of simulation using the underlying *ns-3* physical model. Topologies are static and the spanning tree is calculated prior to simulation start.

Two different traffic models have been simulated: TM1 and TM2. All traffic is TCP.

TM1 generates "long-lived" TCP Internet flows, which start at a random instant in 30 ± 2.5 seconds, and have a download size of 512 KB. A flow implies that a user associated with a given mesh node connects to the server to request a download of 512 KB (i.e the request originates at a mesh node, and reaches the server which sends data to the destination). The number of flows in a scenario varies in {10,20,30,40,50}. For each number of flows, 100 random scenarios are generated, totaling 500 unique scenarios. In each scenario, a flow's sink is chosen randomly (one sink can participate in multiple flows). These scenarios are mostly static (i.e. traffic conditions vary minimally), but note that although flows have the same size, it is possible that not all flows will be active simultaneously due to slightly different start times, and different achieved rate.

TM2 generates highly dynamic traffic patterns. It generates scenarios with fixed number of active users in {10,20,30,40,50}. Given a quantity of users, 10 random scenarios are generated, totaling 50 unique scenarios. In each of these unique scenarios, the set of users which are active at a time changes every 10 seconds. At the start of each 10 second period, an active user is deactivated with probability 0.6. Subsequently, inactive users are randomly activated to complete the number of active users for that scenario. Each user is randomly assigned to a mesh node. When a user is active, the time between start of flows follows an exponential distribution with $\lambda = 1/5$ s (generating flows within the user's 10 second interval), and the size of download traffic follows an exponential distribution with $\lambda = 1/200000$ bytes. Note that this produces scenarios which are highly dynamic. Traffic generation commences at second 30 and finishes at 130.

Each scenario is given an initial CA. This CA is not traffic-aware but however attempts to give more bandwidth to edges which are more likely to carry traffic. To this end, more priority is given to edges closer to the gateway. A similar heuristic is

proposed in [99]. The CA is calculated by applying LACA in the following manner. If $d_h(g, n)$ denotes the minimum distance in hops from node n to the gateway g, the priority of an edge is inversely proportional to the average distance of its endpoints to the gateway, i.e. $t(e_{mn}) = (\frac{d_h(g,m)+d_h(g,n)}{2})^{-1}$. Given the weight $t(e)$ of edges and the Interference Matrix **IM** a CA is calculated using LACA.

In all cases, simulations last until all flows finish. Points shown in graphs represent the average result of scenarios that fall in that class. Confidence intervals are shown at the 95% level.

5.8.3.2 Performance metrics

The following metrics are used to study protocol performance. Flows refer to TCP download flows: (i) *Average network throughput* - total data bytes delivered by the network divided by the time elapsed since the first packet was sent and the last packet was received; (ii) *Peak network throughput* - the maximum throughput of the network observed in a one second interval; (iii) *Flow duration* - time elapsed since the first packet of a flow was sent and the last packet was received; (iv) *Flow throughput* - the number of bytes delivered divided by the duration of the flow; (v) *Flow throughput fairness* - rates the fairness of the throughput achieved by flows in one scenario using Jain's fairness index.

Regarding flow throughput, in each scenario we measure the minimum and median throughput of flows. For flow duration, we measure the median. The median is chosen due to the frequent presence of outliers and skewed distribution of the flow metrics. For example, there can be scenarios where one flow benefits from a much higher throughput than other flows.

5.8.3.3 TM1 results and analysis

The parameters used for LBCA are $\alpha_T = 1$ s, $Lock_t = 2$ s. A value of $\alpha_T = 1$ provides high responsiveness to changing conditions. In this subsection, when we refer to load-balanced routing we refer to the use of routes calculated by LBR (Algorithm 10), whereas CA refers to applying the solution calculated by LACA (Algorithm 11).

This study analyzes and compares the following four strategies, based on whether LBR and/or LACA is enabled or not. The recalculation period is $\alpha_T = 1$ in all cases. The strategies are lbr_cra, nolbr_nocra, nolbr_cra, lbr_nocra. The first one

represents joint load-balanced routing and channel re-assignment. The second one does not perform dynamic routing nor channel re-assignment: routing is static shortest path routing (using hop-count) and channel assignment is based on the initial one. The third does not perform dynamic routing but periodically applies LACA given the current traffic. Finally, the last one performs dynamic routing but channel assignment is static.

The results are shown in Figures 5.8 and 5.9. Fig. 5.8 shows results with varying number of channels and $R = 3$ interfaces in each node. Fig. 5.9 shows results with $R = 6$. The horizontal solid line represents the average performance of LBR in an interference-free case (i.e. every edge assigned to a unique channel). This represents the best performance that can be achieved with LBR. Similarly, the horizontal dashed line is the average performance of shortest path routing in a interference free case. The benefits of load balanced routing are clearly visible by comparing both lines. As we will show, the capability of the protocols of approximating interference-free performance will depend on the number of available radios and channels, and if load-aware channel re-assignment is used.

As can be seen in the figures, there is a notable improvement across all performance metrics when using joint routing and channel re-assignment. Load-balanced routing (calculated by LBR) always balances the traffic across the network links in the same manner regardless of the number of available channels. However, the performance gained by this balancing depends on the number of available radios and channels (see the logarithmic increase in throughput of lbr_cra with the number of channels). The increase in throughput is very similar to the logarithmic decrease of the objective calculated by LACA (see section 5.8.1). When the number of channels is low ($|C| = 5$), there is no performance gain when using load-balanced routing. However, it is also notable that it does not prove particularly harmful even when the number of channels is low (only a few metrics such as median rate and flow duration are slightly worse than nolbr_cra in this case). Note that the fact that traffic is spread throughout the network but there are few channels available may lead to thinking that it is better to use shortest paths to concentrate traffic in the same links and thus avoid interference and MAC inefficiencies.

Load-balanced routing should not be performed without load-aware channel re-assignment. LBR dynamically distributes the load across multiple links and thus uti-

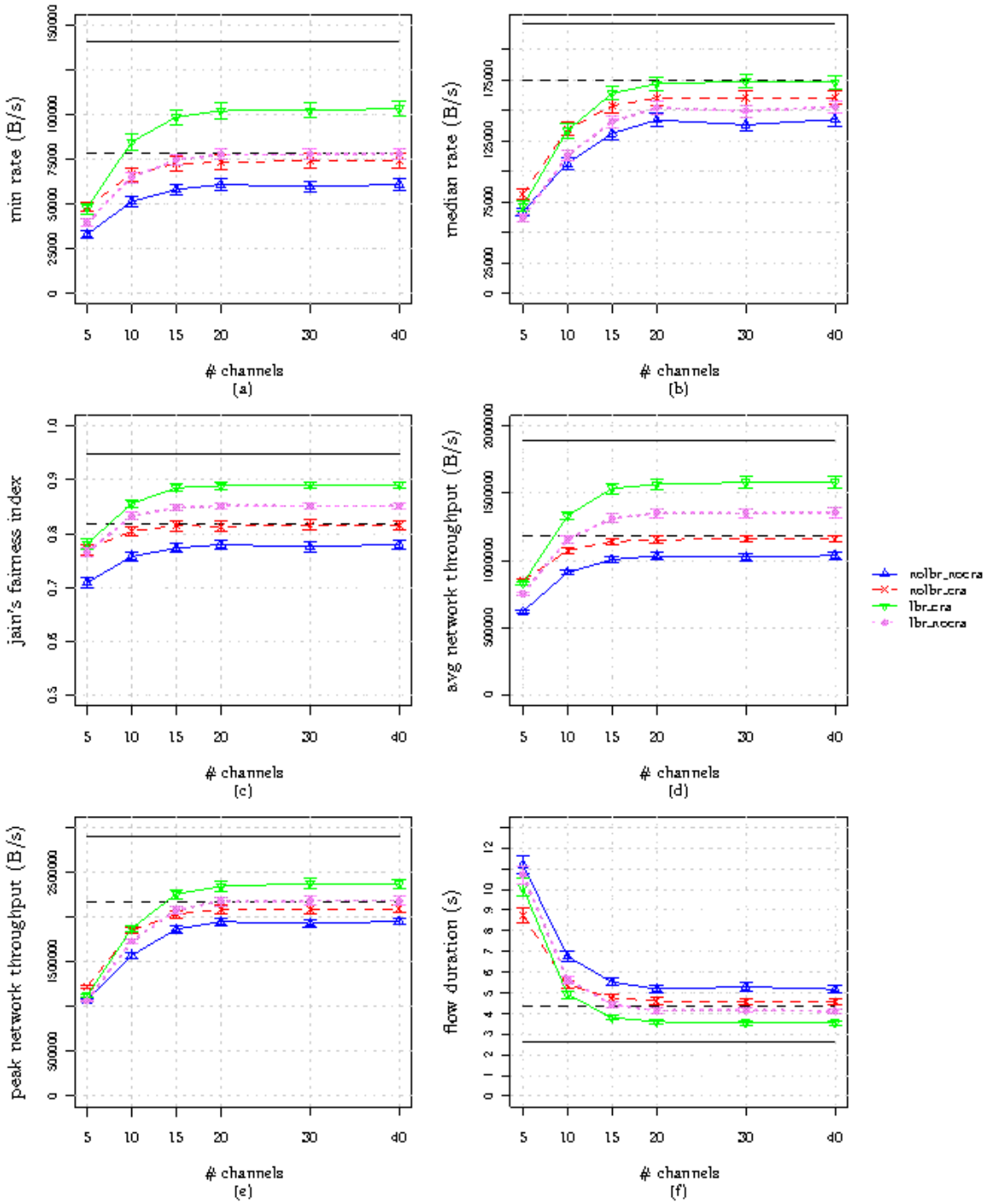


Figure 5.8: LBCA performance results in TM1 with $R = 3$ - The figure shows performance results of LBCA using Traffic Model 1 when $R = 3$.

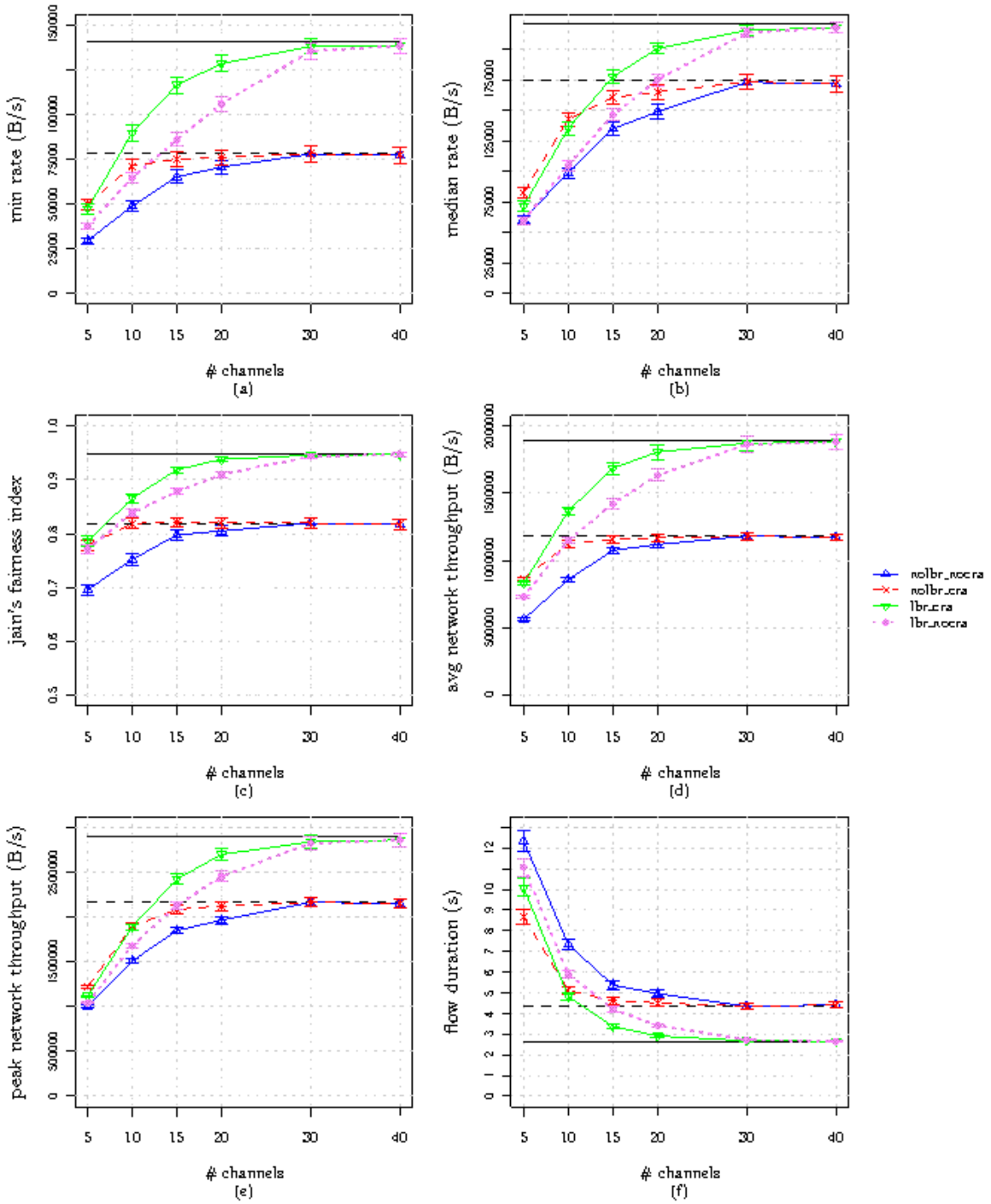


Figure 5.9: LBCA performance results in TM1 with $R = 6$ - The figure shows performance results of LBCA using Traffic Model 1 when $R = 6$.

lizes more network resources than shortest path routing. As such, it relies heavily on load-aware CA to effectively utilize these resources. That is why if LBR is used without LACA, results in some cases can be similar or worse than nolbr_cra (see minimum rate and median rate).

When coupled with LACA, there is a substantial benefit to using load-balanced routing. The best utilization and performance is obtained with the lbr_cra strategy. In other words, the coupling of load-balanced routing and load-aware channel re-assignment is essential to achieve high performance. Probably the greatest advantage of lbr_cra is how it achieves fairness and prevents flow starvation: when $R = 3$, there is an increase in minimum flow rate of up to 70% and median flow rate of up to 25% (with $|C| = 15$), with respect to nolbr_nocra. When $R = 6$, minimum flow rate increases up to 84% and median flow rate up to 37% (when $|C| = 10$). A consequence of improving minimum rate is that fairness improves.

Load-aware CA improves utilization and throughput with both routing strategies (LBR and shortest path), showing that there is an important benefit to optimizing CA for the given set of TCP flows. In the case of shortest path routing, if we compare nolbr_nocra with nolbr_cra, when $R = 3$, there is an increase in minimum flow rate of up to 46% (with $|C| = 5$) and median flow rate of up to 26% (with $|C| = 10$), when using LACA. When $R = 6$, minimum flow rate increases up to 68% (with $|C| = 5$) and median flow rate up to 44% (when $|C| = 10$).

Similarly, if we compare lbr_nocra with lbr_cra, when $R = 3$, there is an increase in minimum flow rate of up to 32% (with $|C| = 15$) and median flow rate of up to 18% (with $|C| = 10$), when using LACA. When $R = 6$, minimum flow rate increases up to 38% and median flow rate up to 29% (when $|C| = 10$).

As we explained previously, without sufficient interfaces, the channel assignment objective (Equation 5.5) may never reach zero regardless of how many channels are available. This is seen in the simulation results, where the use of more than 20 channels when $R = 3$ no longer provides any benefit. With sufficient number of interfaces and channels, however, the performance can match that of a interference-free network. This can be seen when $R = 6$ and the number of channels is above 20. With 20 channels the performance is already very close to that of a interference-free network, and this is achieved with much less channels than number of undirected edges (in the scenarios tested the average number of undirected edges is 146). Note that, in practice, it is not

necessary to equip all routers with the same number of interfaces. First, the maximum number of usable interfaces per node is equal to its node degree, and second, only the routers which carry more traffic (those closest to the gateway) will effectively take advantage of more interfaces.

Note that the shortest path routing strategy reaches interference-free performance with less number of channels than the LBR strategy. In particular, the performance of nolbr_cra can reach the dashed horizontal line with approximately 10 channels, while lbr_cra needs 20 or more channels to reach the solid horizontal line. This is because LBR uses more resources to distribute load. In other words, more channels are needed to realize the maximum potential of load-balanced routing with respect to shortest path routing. In exchange, the performance gain achievable by using load-balanced routing is notably higher.

This section has shown performance results of LBCA in mostly static scenarios. The performance advantage of LBCA is even higher in dynamic scenarios, as we will show in the next section.

5.8.3.4 TM2 results and analysis

Responsiveness of LBCA can prove critical in dynamic scenarios, and depends on the frequency with which it recalculates routing and CA (see section 5.7.4 for a discussion on adaptation frequency and its importance). This subsection evaluates the responsiveness of LBCA under highly dynamic traffic scenarios using traffic model TM2. In these tests, the adaptation period α_T varies in $\{1, 10, 30\}$ seconds.

The results are shown in Figures 5.10 and 5.11. As we can see, in these scenarios there is a substantial impact of the adaptation frequency on the performance of LBCA.

When $R = 3$, there is an observed minimum flow rate when $\alpha_T = 1$ that is 188% higher than $\alpha_T = 30$, and median flow rate 54% higher (with $|C| = 15$). These results also lead to an important increase in fairness and decrease in flow duration (the duration of flows when $\alpha_T = 30$ is up to 158% higher than $\alpha_T = 1$ when $|C| = 15$). A lower period α_T enables LBCA to adapt more quickly to changing conditions, achieving higher performance.

When $R = 6$, the observed minimum flow rate when $\alpha_T = 1$ is up to 175% higher than $\alpha_T = 30$, and median flow rate up to 63% (with $|C| = 10$). The duration of flows when $\alpha_T = 30$ is up to 164% higher than $\alpha_T = 1$ when $|C| = 15$.

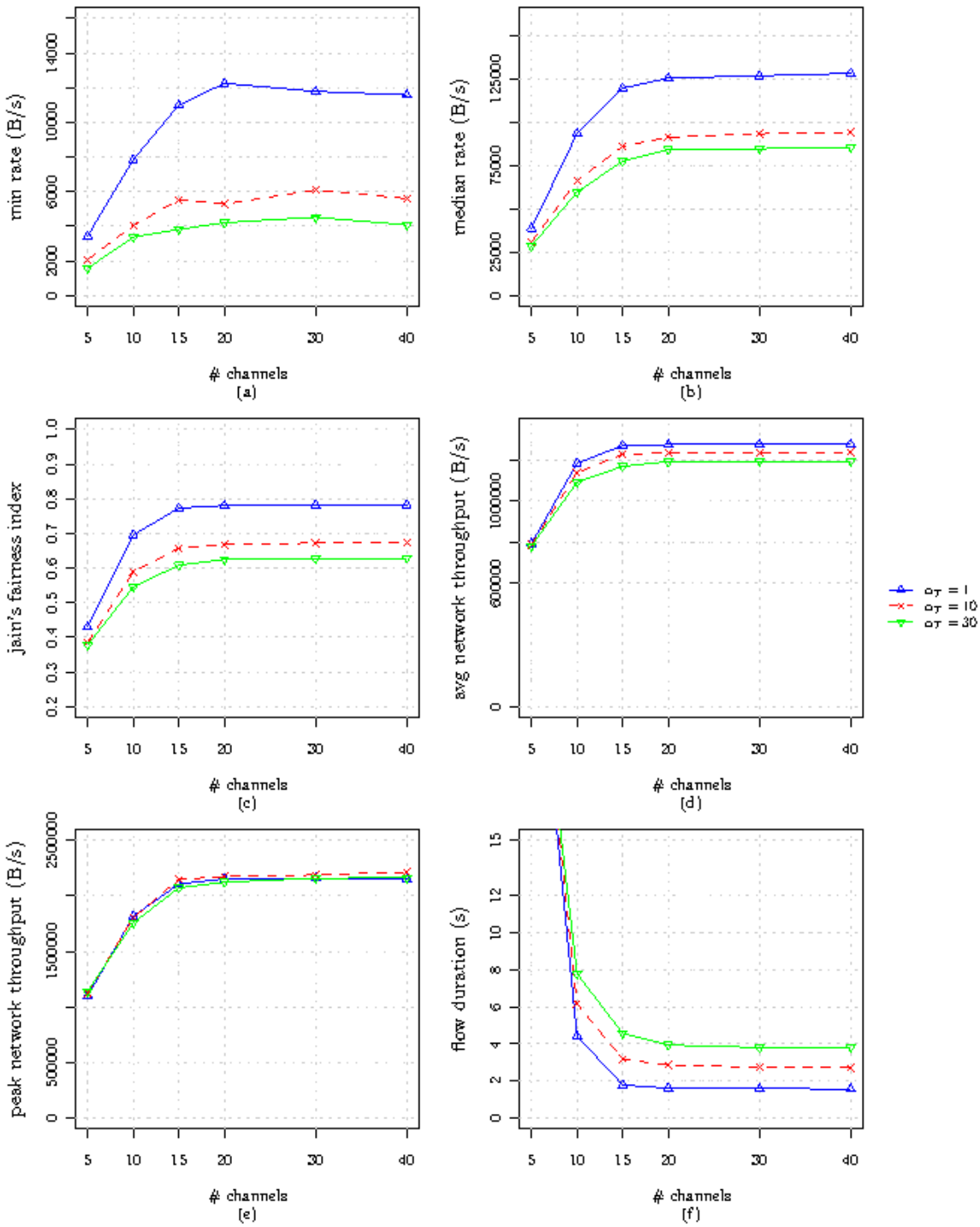


Figure 5.10: LBCA performance results in TM2 with $R = 3$ - The figure compares performance of LBCA when $R = 3$ in the highly dynamic scenarios of Traffic Model 2, with varying adaptation period α_T.

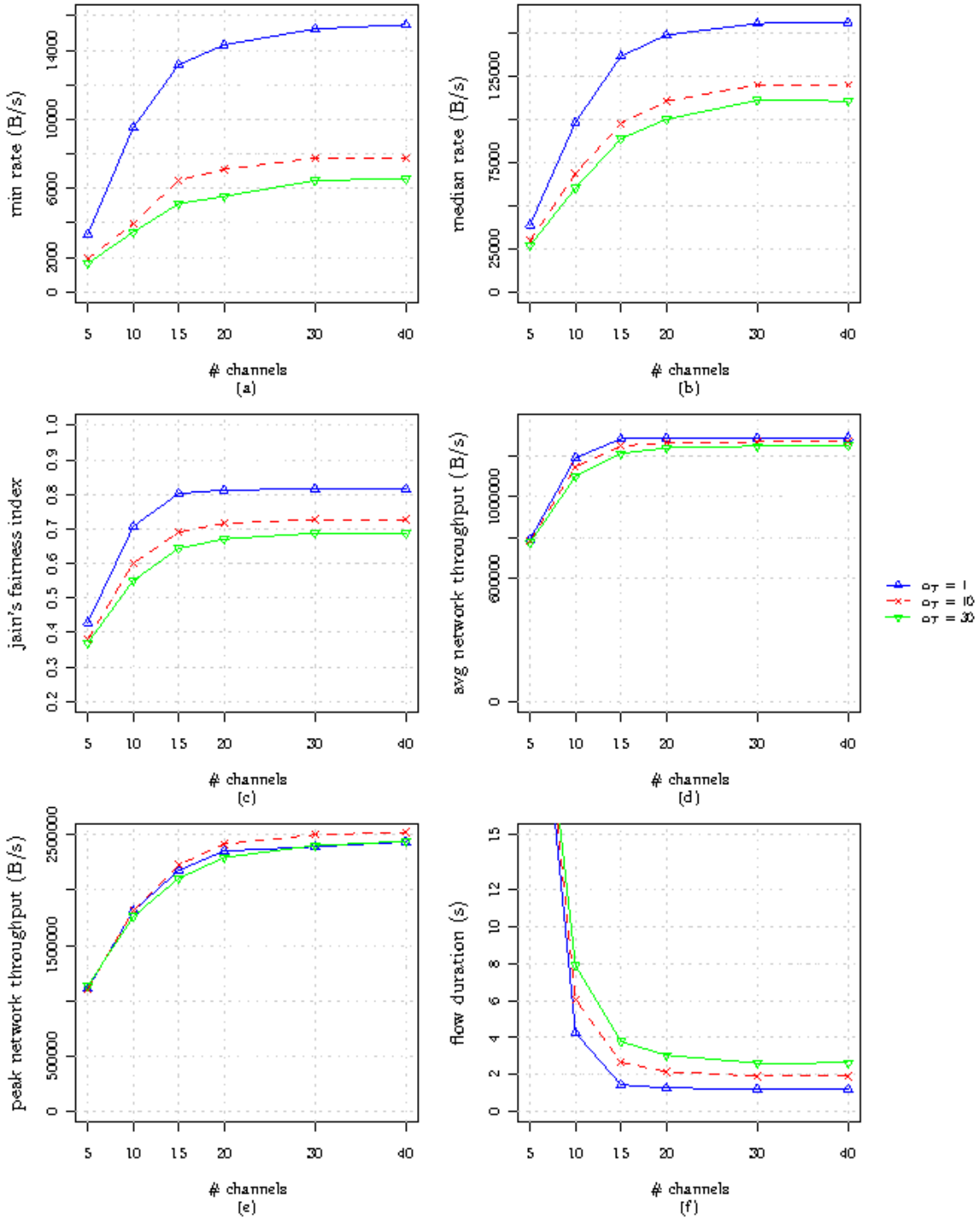


Figure 5.11: LBCA performance results in TM2 with $R = 6$ - The figure compares performance of LBCA when $R = 6$ in the highly dynamic scenarios of Traffic Model 2, with varying adaptation period α_T.

We can conclude that the adaptation frequency of a routing and CA protocol is decisive in dynamic scenarios, specially in WMNs where the small scale of the network can lead to frequently changing and unpredictable traffic. A particular routing solution and channel assignment is only valid as long as the traffic profile and network conditions do not change. Afterward, the solution is invalid or suboptimal until it is updated.

A major advantage of LBCA is precisely the capability of using a high adaptation frequency while introducing minimal overhead in the WMN. To our knowledge, there have not been any proposals of routing and channel assignment protocols for WMNs with an intended adaptation period of less than 60 s.

5.9 Conclusions

Load-balancing in WMNs is necessary to fairly use and exploit all the capacity offered by the network. To achieve this, interference-aware approaches capable of balancing the traffic between collision domains and avoiding intra-flow and inter-flow interference are crucial. Routing and channel assignment are two powerful mechanisms to achieve this. Routing can balance load using paths with low intra-flow and inter-flow interference. Channel assignment, on the other hand, reduces both types of interference by assigning non-overlapping channels to links. By taking into account the current load in the network, CA can grant more capacity to links with more traffic. When considered jointly, interference-aware routing and CA can substantially increase flow throughput and fairness.

This chapter has proposed LBCA, a highly responsive online protocol which dynamically adapts to network and traffic conditions, balancing the load served by a gateway, using dynamic routing and channel re-assignment. The protocol is TCP flow-aware, balances traffic at the TCP flow level, and optimizes CA for the current set of flows in the network, improving flow performance. To the best of our knowledge, this is the first joint routing and CA algorithm to optimize the performance of a specific set of TCP flows. At the TCP flow level, rapid traffic fluctuations are expected, imposing the need for frequent channel re-assignment to adapt to current traffic conditions. The proposed CA algorithm can support frequent channel re-assignment by two properties: (i) low computational complexity, permitting it to quickly calculate a solution, and (ii)

by taking into account the previous CA to generate a new one with minimum variation. A novel CA distribution protocol has also been proposed to quickly disseminate channel allocations through the network with low overhead.

Extensive evaluations show the effectiveness of LBCA. Simulations have been conducted in *ns-3* with an adaptation period of 1 s, showing that the protocol can adapt to the current network conditions with a very high frequency. Tests show that the CA algorithm can efficiently calculate good solutions in gateway access scenarios. Furthermore, tests have shown that channel re-assignment only affects between 25% and 50% of edges when using the proposed load-balancing routing strategy. This means that adapting to varying traffic conditions only requires re-adjusting the channel of a limited set of edges. When performing joint routing and CA, results in (mostly) static scenarios have shown an increase in minimum flow rate of up to 84% and median flow rate of up to 37% compared to a shortest path routing strategy and traffic-unaware CA. LBCA is thus effective at preventing flow starvation and increasing flow fairness. In dynamic scenarios, the advantage of LBCA is even greater. It is shown that adaptation frequency is a determining factor in highly dynamic networks scenarios. Result have shown an increase in minimum flow rate when $\alpha_T = 1$ of up to 175% compared with LBCA when $\alpha_T = 30$, and median flow rate up to 63%. The duration of flows when $\alpha_T = 30$ is up to 164% higher than $\alpha_T = 1$.

Chapter 6

Conclusion

This chapter presents the main conclusions of this thesis, lists the research publications made during its elaboration and discusses future work and research lines.

6.1 Summary and main contributions

The capacity of MWNs is greatly affected by wireless interference between links, namely intra-flow and inter-flow interference. Load balancing is an important mechanism to optimize network utilization. To balance load in MWNs, interference-aware strategies are necessary to distribute load in the network while avoiding both types of interference. This thesis has proposed load-balancing solutions for both single-channel and multi-channel networks. GWLB is designed for single-channel WMNs and balances traffic between gateways, avoiding inter-flow interference. To do this, it estimates interference based on the distance of paths to other nodes, using the proposed PSD metric. LBCA is designed for multi-radio multi-channel WMNs and jointly optimizes routing and channel assignment to balance the load of collision domains inside a gateway domain. Channel assignment is used to reduce both intra-flow and inter-flow interference. Results have shown joint routing and channel assignment to notably improve network utilization and throughput, and greatly improve minimum flow rate (preventing flow starvation) and fairness. The main advantages of the proposed solutions is that they are suitable for implementation in practical scenarios, adapt quickly to changes in network and traffic conditions, and increase the performance and fairness of TCP flows.

This work has shown that it is necessary to balance the instantaneous load of the network. In other words, it is not enough to optimize performance based on long-term average traffic demands. The small scale of MWNs leads to the fact that traffic can vary frequently and unpredictably. A routing or channel assignment solution optimized for a given network state and traffic profile will be invalid or suboptimal if those conditions change. The experiments conducted in dynamic scenarios with GWLB and LBCA clearly show this. In dynamic scenarios, the frequency of adaptation and the convergence time are crucial properties of a load-balancing protocol.

Protocols sensitive to small-scale variations in traffic can lead to network instability. Such protocols must be able to quickly adapt to changes in load, converge quickly to a solution, avoid frequent and unnecessary changes in the solution, all the while limiting the control overhead in the network. The particular architecture of the gateway access scenario in WMNs permits designing efficient centralized solutions that can adapt to the instantaneous load and avoid network instability. It is important to note that the gateway access scenario is one of the most attractive scenarios for the application of WMNs, in which the mesh routers and gateways provide a cost-effective high-speed wireless backbone capable of offering broadband Internet access.

Both GWLB and LBCA solve NP-hard load-balancing optimization problems. The algorithms proposed to solve them are of low-time complexity and calculated centrally. The protocols therefore do not suffer from convergence issues nor high convergence time. GWLB avoids solution instability by locking the gateway assigned to a flow for a specified time. LBCA avoids routing instability in the same manner, and avoids channel assignment instability by limiting the number of channel re-adjustments between channel re-assignments. Through simulation, both protocols have been shown to be able to adapt with an adaptation period as low as one second, showing improved performance with a lower period.

An important advantage of both GWLB and LBCA is that they balance traffic at the TCP flow level. That is, the solution is optimized for the current set of TCP flows in the network. It is important to note that most Internet traffic today is based on TCP. Balancing traffic at the TCP flow level allows to improve both the throughput and fairness of flows. This is specially necessary in MWNs, where TCP flows are known to suffer from starvation and unfairness.

6.2 Publications

This subsection lists the most relevant research publications made during the development of this thesis.

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "Spatially Disjoint Multipath Routing protocol without location information". 33rd IEEE Conference on Local Computer Networks (LCN), 2008.

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "A Distributed Algorithm for Gateway Load-Balancing in Wireless Mesh Networks", in 1st IFIP Wireless Days Conference 2008. Nov. 2008.

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "Achieving spatial disjointness in multipath routing without location information". In Proceedings of the 2009 IEEE conference on Wireless Communications & Networking Conference (WCNC'09).

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "A feedback-based adaptive online algorithm for Multi-Gateway load-balancing in wireless mesh networks". In Proceedings of the 2010 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM 2010).

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "Multipath routing with spatial separation in wireless multi-hop networks without location information". Computer Networks, The International Journal of Computer and Telecommunications Networking, Elsevier. 55, 3 (February 2011), 583-599.

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "Responsive On-line Gateway Load-Balancing for Wireless Mesh Networks". Ad Hoc Networks journal (accepted for publication), Elsevier.

- Galvez, J.J.; Ruiz, P.M.; Skarmeta, A.; "TCP flow-aware Channel Re-Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks". Accepted for publication at the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2011), October 2011.

6.3 Future work

The gateway access scenario in WMNs considered in this work offers a promising field for further research. The fact that mesh routers are stationary, are not limited in energy or processing power and can be equipped with advanced antenna and radio technology offers many opportunities to increase the network capacity and optimize its utilization. The LBCA system proposed in this work is an online load-balancing system for multi-radio multi-channel WMNs through joint routing and channel assignment. There are many ways to improve and extend the LBCA architecture and system.

One line of future work involves taking into account heterogeneous link capacity in the LBCA system. Several factors can lead to heterogeneous link capacity, such as multi-rate WMNs where links can transmit using different rate, and varying link quality (e.g. packet loss rate). The Expected Transmission Time (ETT) is a suitable metric to reflect link bandwidth, taking into account both the link rate and link quality. However, depending on when and how it is measured, and the interference conditions of the network, its value can be influenced by the current network load. For the particular case of LBCA, it is important that the link quality metric reflect link capacity independent of the current load. This is because the controller already has accurate traffic information (set of active TCP flows), and so the link metric must not be not load-dependent. Obtaining up-to-date, load-insensitive, link capacity measurements with minimal overhead is a problem that has to be addressed. Another problem is optimizing routing and channel assignment based on heterogeneous link capacities.

Results for LBCA are promising and show that interference-free performance can be approximated in the gateway access scenario when sufficient channels are available. The number of channels necessary for this is not too high using the proposed "ring-node" architecture. Cognitive radio techniques can prove particularly useful when the number of channels available to the physical layer technology over the unlicensed band is not enough to achieve interference-free performance. These techniques can be used to increase the number of channels available in the system. In this paradigm, it is possible to allow unlicensed users to utilize licensed bands whenever it does not cause any interference with primary (licensed) users. Spectrum sensing techniques permit radios to detect the unused spectrum (spectrum holes) and share it without harmful interference with other users. Detecting primary users is the most efficient way to

detect spectrum holes. Because the presence of primary users can vary across the network area, this introduces modifications to the channel assignment problem. The set of channels that can be assigned to links is not the same in general. In the case of LBCA, this requires collecting information on the channel availability for each network link and may require modifications to the channel assignment algorithm.

In this thesis, load balancing has been shown to substantially improve throughput and fairness by effectively utilizing available network resources. The use of multiple links (using different radios and channels) to connect two neighbors may provide additional balancing options. First it is necessary to study if notable gains can be achieved using this strategy in the gateway access scenario. And if there are, solutions are needed to achieve this without negatively impacting the routing process. In this paradigm, a node can communicate with a neighbor using multiple interfaces, which means that there are multiple routes to reach a neighbor. This can lead to out-of-order delivery of packets and round trip time variations, which can harm the performance of TCP flows. To avoid this, it is necessary to send the packets of a flow through the same links, i.e. same sequence of next hops and interfaces. In the case of LBCA, this can be achieved by extending the source routing technique to take into account the complete route (including interfaces), at the cost of increased overhead. To avoid having to include the source route in every packet, it is possible to use a stateful routing mechanism similar to DSRFLOW, such that each node stores the link to be used to route each flow currently traversing the node.

To optimize the capacity and utilization of WMNs, an approach which introduces transmission power control should also be considered. Transmission power of nodes influences interference, link rate and quality, and network connectivity (set of neighbors reachable by a node). A joint optimization of routing, channel assignment and transmission power based on the instantaneous traffic conditions can further increase network performance. Much in the same way as channel re-assignment, adapting power control to current traffic will require dynamically varying the transmit power of radio interfaces. In LBCA, this may require disseminating power control commands through the network, and therefore it is important to devise a scheme to limit overhead (e.g. by limiting the number of power variations).

Another important aspect for the improvement of service in WMNs is Quality of Service (QoS) and fairness. It is important to guarantee not only flow fairness but

user fairness as well. QoS mechanisms are important because they offer performance guarantees in the network and classes of service. To effectively achieve this, support at every network layer is needed.

Finally, it is important to validate the algorithms and protocols proposed in this thesis using a real-world multi-hop wireless testbed or deployment. To effectively validate the load-balancing capabilities of the proposed systems, an experimental multi-radio multi-channel WMN testbed is necessary with a scale resembling the expected deployment of future WMNs.

References

[1] **BonnMotion: A mobility scenario generation and analysis tool**. Available from: http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/. 99

[2] **Mobile Ad hoc Networks (MANET) charter**. Available from: http://datatracker.ietf.org/wg/manet/charter. 18

[3] **The ns-3 network simulator**. Available from: http://www.nsnam.org/. 127, 178

[4] **IEEE 802.1D - MAC Bridges**, 2004. Available from: http://standards.ieee.org/getieee802/download/802.1D-2004.pdf. 172

[5] **Couenne, an exact solver for nonconvex MINLPs**, 2010. Available from: https://projects.coin-or.org/Couenne. 125

[6] Norman Abramson. **THE ALOHA SYSTEM: another alternative for computer communications**. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, AFIPS '70 (Fall), pages 281–285, New York, NY, USA, 1970. ACM. Available from: http://doi.acm.org/10.1145/1478462.1478502. 18

[7] Arup Acharya, Archan Misra, and Sorav Bansal. **High-performance architectures for IP-based multihop 802.11 networks**. *IEEE Wireless Communications*, **10**:22–28, 2003. 20

[8] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li. **Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks**. In *MobiCom '05: Proceedings of the 11th annual*

international conference on Mobile computing and networking, pages 58–72, New York, NY, USA, 2005. ACM. Available from: http://doi.acm.org/10.1145/1080829.1080836. 10, 54, 106, 138, 140, 142, 143

[9] S. Avallone, F.P. D'Elia, and G. Ventre. **A traffic-aware channel re-assignment algorithm for wireless mesh networks**. In *Proceedings of European Wireless*, pages 683–688. IEEE, April 2010. Available from: http://dx.doi.org/10.1109/EW.2010.5483450. 144

[10] Stefano Avallone, Ian F. Akyildiz, and Giorgio Ventre. **A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks**. *IEEE/ACM Trans. Netw.*, **17**:267–280, February 2009. Available from: http://dx.doi.org/10.1109/TNET.2008.918091. 54, 142, 143

[11] Yossi Azar, Joseph (Seffi) Naor, and Raphael Rom. **The competitiveness of on-line assignments**. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 203–210, 1992. 220, 223

[12] Paramvir Bahl, Ranveer Chandra, and John Dunagan. **SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks**. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom '04, pages 216–230, 2004. Available from: http://doi.acm.org/10.1145/1023720.1023742. 4, 27

[13] Yigal Bejerano, Seung-Jae Han, and Amit Kumar. **Efficient load-balancing routing for wireless mesh networks**. *Comput. Netw.*, **51**(10):2450–2466, 2007. Available from: http://dx.doi.org/10.1016/j.comnet.2006.09.018. 106

[14] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. **Branching and bounds tightening techniques for non-convex MINLP**. *Optimization Methods and Software*, **24**(4-5):597–634, 2009. 125

[15] Dimitri P. Bertsekas and Robert G. Gallager. **Distributed Asynchronous Bellman-Ford Algorithm**. In *Data Networks*, chapter 5.2.4, pages 325–333. Prentice Hall, Englewood Cliffs, 1987. 32, 33

[16] I. Chakeres and C. Perkins. **Dynamic MANET On-demand (DYMO) Routing**, July 2010. INTERNET-DRAFT draft-ietf-manet-dymo-21.txt. Available from: http://tools.ietf.org/id/draft-ietf-manet-dymo-21.txt. 37

[17] Wei Cheng, Kai Xing, Xiuzhen Cheng, Xicheng Lu, Zexin Lu, Jinshu Su, Baosheng Wang, and Yujun Liu. **Route recovery in vertex-disjoint multipath routing for many-to-one sensor networks**. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 209–220, New York, NY, USA, 2008. ACM. Available from: http://doi.acm.org/10.1145/1374618.1374648. 61

[18] B. Claise. **Cisco Systems NetFlow Services Export Version 9**, October 2004. RFC 3954. Available from: http://www.ietf.org/rfc/rfc3954.txt. 112

[19] B. Claise. **Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information**, January 2008. RFC 5101. Available from: http://www.ietf.org/rfc/rfc5101.txt. 112, 121, 142, 172

[20] T. Clausen and P. Jacquet. **Optimized Link State Routing Protocol (OLSR)**, October 2003. RFC 3626. Available from: http://www.ietf.org/rfc/rfc3626.txt. 34, 59, 80

[21] M. Scott Corson and Anthony Ephremides. **A Distributed Routing Algorithm for Mobile Wireless Networks**. *Wireless Networks*, **1**(1):61–81, 1995. Available from: http://dx.doi.org/10.1007/BF01196259. 38

[22] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. **A high-throughput path metric for multi-hop wireless routing**. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146. ACM, 2003. Available from: http://doi.acm.org/10.1145/938985.939000. 27, 42, 104

[23] Arindam K. Das, Hamed M. K. Alazemi, Rajiv Vijayakumar, and Sumit Roy. **Optimization models for fixed channel assignment in wireless mesh networks with multiple radios**. In *SECON*, pages 463–474, 2005. 52

[24] Ernest Davis and Jeffrey M. Jaffe. **Algorithms for Scheduling Tasks on Unrelated Processors**. *J. ACM*, **28**(4):721–736, 1981. Available from: http://doi.acm.org/10.1145/322276.322284. 222, 224, 226

[25] Aditya Dhananjay, Hui Zhang, Jinyang Li, and Lakshminarayanan Subramanian. **Practical, distributed channel assignment and routing in dual-radio mesh networks**. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 99–110, 2009. Available from: http://doi.acm.org/10.1145/1592568.1592581. 10, 55, 142, 144

[26] Richard Draves, Jitendra Padhye, and Brian Zill. **Comparison of routing metrics for static multi-hop wireless networks**. *SIGCOMM Comput. Commun. Rev.*, **34**:133–144, August 2004. Available from: http://doi.acm.org/10.1145/1030194.1015483. 42

[27] Richard Draves, Jitendra Padhye, and Brian Zill. **Routing in multi-radio, multi-hop wireless mesh networks**. In *MobiCom '04: Proc. of conference on Mobile computing and networking*, pages 114–128. ACM, 2004. Available from: http://doi.acm.org/10.1145/1023720.1023732. 27, 43, 104

[28] K. Fall and K. Varadham. **The ns Manual**. Available from: http://www.isi.edu/nsnam/ns/ns-documentation.html. 92

[29] Eli M. Gafni and Dimitri P. Bertsekas. **Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology**. In *IEEE Transactions on Communications*, **29**, pages 11–18. IEEE, January 1981. Available from: http://www.mit.edu/people/dimitrib/Gafni_Loopfree.pdf. 38

[30] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. **Computing Nash equilibria for scheduling on restricted parallel links**. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 613–622. ACM, 2004. Available from: http://doi.acm.org/10.1145/1007352.1007446. 222, 224, 226

[31] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. **Computing Nash Equilibria for Scheduling on Restricted Parallel Links**. *Theor. Comp. Sys.*, **47**(2):405–432, 2010. Available from: http://dx.doi.org/10.1007/s00224-009-9191-9. 116, 117

[32] Martin Gairing, Burkhard Monien, and Andreas Woclaw. **A Faster Combinatorial Approximation Algorithm for Scheduling Unrelated Parallel Machines**. In *ICALP '05: Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, pages 828–839, 2005. 222

[33] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **A Distributed Algorithm for Gateway Load-balancing in Wireless Mesh Networks**. In *Wireless Days, 2008. WD '08. 1st IFIP*, pages 1–5, 2008. 105, 108, 127

[34] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **Spatially Disjoint Multipath Routing protocol without location information**. In *33rd IEEE Conference on Local Computer Networks (LCN)*. IEEE Computer Society, 2008. 60

[35] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **Achieving Spatial Disjointness in Multipath Routing without Location Information**. In *IEEE Wireless Communications and Networking Conference (WCNC 2009)*, pages 1–6. IEEE Computer Society, 2009. Available from: http://dx.doi.org/10.1109/WCNC.2009.4917556. 60

[36] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **A Feedback-based Adaptive Online Algorithm for Multi-Gateway Load-Balancing in Wireless Mesh Networks**. In *Proceedings of the 2010 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, WOWMOM '10, pages 1–9, Washington, DC, USA, 2010. IEEE Computer Society. Available from: http://dx.doi.org/10.1109/WOWMOM.2010.5534914. 105

[37] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **Multipath routing with spatial separation in wireless multi-hop networks without location information**. *Computer Networks*, **55**(3):583–599, 2011. Available from: http://dx.doi.org/10.1016/j.comnet.2010.09.016. 60

[38] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **Responsive on-line gateway load-balancing for wireless mesh networks**. *Ad Hoc Networks*, 2011. Available from: http://dx.doi.org/10.1016/j.adhoc.2011.06.002. 105

[39] Juan J. Galvez, Pedro M. Ruiz, and Antonio F. Gomez-Skarmeta. **TCP flow-aware Channel Re-Assignment in Multi-Radio Multi-Channel Wireless Mesh Networks**. In *8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS)*. IEEE Computer Society, 2011. 141

[40] Yashar Ganjali and Abtin Keshavarzian. **Load Balancing in Ad Hoc Networks: Single-path Routing vs. Multi-path Routing**. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, **2**, pages 1120–1125, 2004. 57

[41] Vanessa Gardellin, Sajal K. Das, Luciano Lenzini, Claudio Cicconetti, and Enzo Mingozzi. **G-PaMeLA: A divide-and-conquer approach for joint channel assignment and routing in multi-radio multi-channel wireless mesh networks**. *J. Parallel Distrib. Comput.*, **71**:381–396, March 2011. Available from: http://dx.doi.org/10.1016/j.jpdc.2010.10.008. 54, 142, 143

[42] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan. **Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey**. *Annals of Discrete Mathematics*, **5**:287–326, 1979. 222

[43] Piyush Gupta and P. R. Kumar. **The Capacity of Wireless Networks**. *IEEE Transactions on Information Theory*, pages 388–404, 2000. Available from: http://dx.doi.org/10.1109/18.825799. 21, 25, 52, 111, 123

[44] Hossam Hassanein and Audrey Zhou. **Routing with load balancing in wireless Ad hoc networks**. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '01, pages 89–96. ACM, 2001. Available from: http://doi.acm.org/10.1145/381591.381614. 46

[45] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. **A rate-adaptive MAC protocol for multi-Hop wireless networks**. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 236–251, New York, NY, USA, 2001. ACM. Available from: http://doi.acm.org/10.1145/381677.381700. 20, 146

[46] Yih-Chun Hu and David B. Johnson. **Implicit source routes for on-demand ad hoc network routing**. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '01, pages 1–10, New York, NY, USA, 2001. ACM. Available from: http://doi.acm.org/10.1145/501417.501418. 37

[47] Chi-Fu Huang, Hung-Wei Lee, and Yu-Chee Tseng. **A Two-Tier Heterogeneous Mobile Ad Hoc Network Architecture and its Load-Balance Routing Problem**. *Mobile Networks and Applications*, **9**(4):379–391, 2004. Available from: http://doi.acm.org/10.1145/1012215.1012228. 10, 107, 108, 127, 129

[48] Lifei Huang and Ten-Hwang Lai. **On the scalability of IEEE 802.11 ad hoc networks**. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 173–182, New York, NY, USA, 2002. ACM. Available from: http://doi.acm.org/10.1145/513800.513822. 20

[49] Oscar H. Ibarra and Chul E. Kim. **Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors**. *J. ACM*, **24**(2):280–289,

1977. Available from: http://doi.acm.org/10.1145/322003.322011. 116, 222, 224

[50] Masashi Ito, Toshihiro Shikama, and Akira Watanabe. **Proposal and evaluation of multiple gateways distribution method for wireless mesh network**. In *ICUIMC '09: Proc. of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 18–25. ACM, 2009. Available from: http://doi.acm.org/10.1145/1516241.1516246. 108

[51] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. **Optimized Link State Routing Protocol for Ad Hoc Networks**. In *Proc. IEEE International Multi Topic Conference Technology for the 21st Century (INMIC '01)*, pages 62–68, 2001. 34, 80

[52] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. **Impact of interference on multi-hop wireless network performance**. *Wirel. Netw.*, **11**:471–487, July 2005. Available from: http://dx.doi.org/10.1007/s11276-005-1769-9. 7, 24, 26, 27, 52

[53] David B. Johnson, David A. Maltz, and Josh Broch. **DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks**. In *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley, 2001. 36

[54] David B. Johnson, David A. Maltz, and Yih-Chun Hu. **The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4**, February 2007. RFC 4728. Available from: http://www.ietf.org/rfc/rfc4728.txt. 36, 37

[55] E.P.C. Jones, M. Karsten, and P.A.S. Ward. **Multipath Load Balancing in Multi-hop Wireless Networks**. In *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005)*, **2**, pages 158–166, August 2005. 57

[56] Sung ju Lee and Mario Gerla. **Dynamic Load-Aware Routing in Ad hoc Networks**. In *Proceedings of the IEEE Conference on Communications*,

pages 3206–3210, 2001. Available from: http://dx.doi.org/10.1109/ICC.2001.937263. 46

[57] R.E. Kahn, S.A. Gronemeyer, J. Burchfiel, and R.C. Kunzelman. **Advances in packet radio technology**. *Proceedings of the IEEE*, **66**(11):1468–1496, 1978. 17, 18

[58] R. M. Karp. **Reducibility Among Combinatorial Problems**. *Complexity of Computer Computations*, pages 85–103, 1972. 154

[59] Frank Kelly. **Charging and Rate Control for Elastic Traffic**. *European Transactions on Telecommunications*, **8**:33–37, 1997. 123

[60] J. M. Kleinberg. **Single-source unsplittable flow**. In *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 68, Washington, DC, USA, 1996. IEEE Computer Society. 107, 110, 116

[61] Jon Kleinberg, Eva Tardos, and Yuval Rabani. **Fairness in Routing and Load Balancing**. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 568, Washington, DC, USA, 1999. IEEE Computer Society. 110

[62] Murali Kodialam and Thyaga Nandagopal. **Characterizing the capacity region in multi-radio multi-channel wireless mesh networks**. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, MobiCom '05, pages 73–87, 2005. Available from: http://doi.acm.org/10.1145/1080829.1080837. 10, 140

[63] P. R. Kumar, Piyush Gupta, and Robert Gray. **An Experimental Scaling Law for Ad Hoc Networks**. In *Mobile Ad Hoc Networking and Computing*, 2001. 25

[64] Pradeep Kyasanur and Nitin H. Vaidya. **Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks**. *SIGMOBILE Mob. Comput. Commun. Rev.*, **10**:31–43, January 2006. Available from: http://doi.acm.org/10.1145/1119759.1119762. 10, 50, 55, 142, 144

[65] Mathieu Lacage and Thomas R. Henderson. **Yet another network simulator**. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, New York, NY, USA, 2006. ACM. Available from: http://doi.acm.org/10.1145/1190455.1190467. 178

[66] Sriram Lakshmanan, Raghupathy Sivakumar, and Karthikeyan Sundaresan. **Multi-gateway Association in wireless mesh networks**. *Ad Hoc Netw.*, 7(3):622–637, 2009. Available from: http://dx.doi.org/10.1016/j.adhoc.2008.07.005. 108

[67] S. Lee and M. Gerla. **Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks**. *In Proceedings of the IEEE ICC*, pages 3201–3205, 2001. Available from: http://dx.doi.org/10.1109/ICC.2001.937262. 39, 57, 61

[68] B.M. Leiner, R.J. Ruther, and A.R. Sastry. **Goals and challenges of the DARPA GloMo program**. *IEEE Personal Communications[see also IEEE Wireless Communications]*, **3**(6):34–43, 1996. 18

[69] J. K. Lenstra, D. B. Shmoys, and É. Tardos. **Approximation algorithms for scheduling unrelated parallel machines**. *Math. Program.*, **46**(3):259–271, 1990. Available from: http://dx.doi.org/10.1007/BF01585745. 222

[70] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. **Capacity of ad hoc wireless networks**. In *MobiCom '01: Proc. 7th ACM Int. Conf. on Mobile Computing and Networking*, 2001. 6, 25, 111

[71] Xiaojun Lin and Shahzada B. Rasool. **Distributed and provably efficient algorithms for joint channel-assignment, scheduling, and routing in multichannel ad hoc wireless networks**. *IEEE/ACM Trans. Netw.*, **17**:1874–1887, December 2009. Available from: http://dx.doi.org/10.1109/TNET.2009.2021841. 142, 144

[72] V. Loscri and S. Marano. **A new Geographic Multipath Protocol for Ad hoc Networks to reduce the Route Coupling phenomenon**. In *Ve-*

hicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, pages 1102–1106, 2006. 8, 61

[73] Liang Ma and Mieso K. Denko. **A Routing Metric for Load-Balancing in Wireless Mesh Networks**. In *AINAW '07, Advanced Information Networking and Applications Workshops*, pages 409–414, 2007. 10, 108

[74] Mohammad Mahdian. **On the computational complexity of strong edge coloring**. *Discrete Appl. Math.*, **118**:239–248, May 2002. Available from: http://dx.doi.org/10.1016/S0166-218X(01)00237-2. 52

[75] Mahesh K. Marina and Samir R. Das. **Ad hoc On-demand Multipath Distance Vector Routing**. *Wireless Communications and Mobile Computing*, **6**(7):969–988, 2006. Available from: http://dx.doi.org/10.1002/wcm.v6:7. 40, 58, 61, 93

[76] Mahesh K. Marina, Samir R. Das, and Anand Prabhu Subramanian. **A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks**. *Comput. Netw.*, **54**:241–256, February 2010. Available from: http://dx.doi.org/10.1016/j.comnet.2009.05.015. 52

[77] Silvano Martello, François Soumis, and Paolo Toth. **Exact and approximation algorithms for makespan minimization on unrelated parallel machines**. *Discrete Appl. Math.*, **75**(2):169–188, 1997. Available from: http://dx.doi.org/10.1016/S0166-218X(96)00087-X. 222

[78] Stefano Maurina, Roberto Riggio, Tinku Rasheed, and Fabrizio Granelli. **On Tree-Based Routing in Multi-Gateway Association based Wireless Mesh Networks**. In *The 20th Personal, Indoor and Mobile Radio Communications Symposium (PIMRC'09)*, pages 807–812, 2009. 108, 127

[79] Vivek Mhatre, Henrik Lundgren, Francois Baccelli, and Christophe Diot. **Joint MAC-aware routing and load balancing in mesh networks**. In *Proc. of the 2007 ACM CoNEXT conference*, pages 1–12, 2007. Available from: http://doi.acm.org/10.1145/1364654.1364679. 107

[80] A.H. Mohsenian-Rad and V.W.S. Wong. **Logical topology design and interface assignment for multi-channel wireless mesh networks**. In *Proceedings of the Global Telecommunications Conference*, Globecom '06, 2006. 49, 140

[81] A.H. Mohsenian-Rad and V.W.S. Wong. **Joint logical topology design, interface assignment, channel allocation, and routing for multi-channel wireless mesh networks**. *IEEE Transactions on Wireless Communications*, **6**:4432–4440, December 2007. Available from: http://dx.doi.org/10.1109/TWC.2007.060312. 10, 54, 140, 142, 143

[82] E. Mokotoff and P. Chretienne. **A cutting plane algorithm for the unrelated parallel machine scheduling problem**. *European Journal of Operational Research*, **141**(3):515–525, 2002. 222

[83] S. Motegi and H. Horiuchi. **AODV-based Multipath Routing Protocol for Mobile Ad hoc Networks**. *IEICE Trans. Commun.*, 2004. 61

[84] Maimour Moufida. **Maximally Radio-Disjoint Multipath Routing for Wireless Multimedia Sensor Networks**. In *WMuNep '08: Proceedings of the 4th ACM workshop on Wireless multimedia networking and performance modeling*, pages 26–31, New York, NY, USA, 2008. ACM. Available from: http://doi.acm.org/10.1145/1454573.1454579. 61

[85] Stephen Mueller and Dipak Ghosal. **Analysis of a Distributed Algorithm to Determine Multiple Routes with Path Diversity in Ad Hoc Networks**. In *Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05)*, pages 277–285, 2005. 61

[86] Shree Murthy and J. Garcia-Luna-Aceves. **An efficient routing protocol for wireless networks**. *Mobile Networks and Applications*, **1**:183–197, 1996. 10.1007/BF01193336. Available from: http://dx.doi.org/10.1007/BF01193336. 33

[87] Deepti Nandiraju, Lakshmi Santhanam, Nagesh Nandiraju, and Dharma P. Agrawal. **Achieving Load Balancing in Wireless Mesh Net-**

works Through Multiple Gateways. In *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 807–812, 2006. 10, 108

[88] Asis Nasipuri, Robert Castaeda, and Samir R. Das. **Performance of Multipath Routing for On-demand Protocols in Mobile Ad Hoc Networks**. *ACM/Kluwer Mobile Networks and Applications (MONET)*, **6**(4):339–349, 2001. Available from: http://dx.doi.org/10.1023/A:1011426611520. 38, 61

[89] Nam T. Nguyen, An-I Andy Wang, Peter Reiher, and Geoff Kuenning. **Electric-Field-Based Routing: A Reliable Framework for Routing in MANETs**. *SIGMOBILE Mob. Comput. Commun. Rev.*, **8**(2):35–49, 2004. Available from: http://doi.acm.org/10.1145/997122.997129. 8, 61, 68

[90] Dragoş Niculescu. **Interference map for 802.11 networks**. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 339–350. ACM, 2007. Available from: http://doi.acm.org/10.1145/1298306.1298355. 7, 23, 171

[91] Jitendra Padhye, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. **Estimation of link interference in static multi-hop wireless networks**. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 28–28, Berkeley, CA, USA, 2005. USENIX Association. Available from: http://portal.acm.org/citation.cfm?id=1251086.1251114. 7, 23, 171, 172

[92] Vincent D. Park and M. Scott Corson. **A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks**. In *Proc. IEEE Infocom*, pages 1405–1413, 1997. 38

[93] Marc R. Pearlman, Zygmunt J. Haas, Peter Sholander, and Siamak S. Tabrizi. **On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks**. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 3–10, Piscataway, NJ, USA, 2000. IEEE Press. 8, 57, 60, 62

[94] C. Perkins, E. Belding-Royer, and S. Das. **Ad hoc On-Demand Distance Vector (AODV) Routing**, July 2003. RFC 3561. Available from: http://www.ietf.org/rfc/rfc3561.txt. 33, 35, 61, 93

[95] C. E. Perkins and E. M. Royer. **Ad Hoc On-Demand Distance Vector Routing**. *In Proc. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999. 35

[96] Charles E. Perkins and Pravin Bhagwat. **Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers**. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, NY, USA, 1994. ACM Press. Available from: http://doi.acm.org/10.1145/190314.190336. 32

[97] P.P. Pham and S. Perreau. **Performance analysis of reactive shortest path and multipath routing mechanism with load balance**. In *INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, **1**, pages 251–259, 2003. 41

[98] K. Ramachandran, M. Buddhikot, G. Chandranmenon, S. Miller, E. Belding-Royer, and K. Almeroth. **On the Design and Implementation of Infrastructure Mesh Networks**. In *Proceedings of the IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005. 10, 108

[99] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot. **Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks**. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, INFOCOM '06, 2006. Available from: http://dx.doi.org/10.1109/INFOCOM.2006.177. 144, 185

[100] Krishna Ramachandran, Irfan Sheriff, Elizabeth Belding, and Kevin Almeroth. **Routing stability in static wireless mesh networks**. In *Proceedings of the 8th international conference on Passive and active network measurement*, PAM'07, pages 73–83, Berlin, Heidelberg, 2007. Springer-Verlag. Available from: http://portal.acm.org/citation.cfm?id=1762888.1762899. 47

[101] Krishna Ramachandran, Irfan Sheriff, Elizabeth M. Belding, and Kevin C. Almeroth. **A multi-radio 802.11 mesh network architecture**. *Mob. Netw. Appl.*, **13**:132–146, April 2008. Available from: http://dx.doi.org/10.1007/s11036-008-0026-8. 10, 55, 143

[102] A Raniwala and Tzi-Cker Chiueh. **Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network**. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, **3**, pages 2223–2234, 2005. Available from: http://dx.doi.org/10.1109/INFCOM.2005.1498497. 10, 49, 55, 104, 107, 127, 140, 142

[103] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh. **Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks**. *SIGMOBILE Mob. Comput. Commun. Rev.*, **8**(2):50–65, 2004. Available from: http://doi.acm.org/10.1145/997122.997130. 10, 27, 53, 55, 106, 138, 139, 140, 142, 162

[104] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. **Measurement-based models of delivery and interference in static wireless networks**. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 51–62. ACM, 2006. Available from: http://doi.acm.org/10.1145/1159913.1159921. 7, 23, 171

[105] Siuli Roy, Somprakash Bandyopadhyay, Tetsuro Ueda, and Kazuo Hasuike. **Multipath Routing in Ad Hoc Wireless Networks with Omni Directional and Directional Antenna: A Comparative Study**. In *Distributed Computing*, Lecture Notes in Computer Science, pages 184–191. Springer Berlin / Heidelberg, 2002. 61

[106] R. Ruth. **Global Mobile Information Systems Program Overview**, July 1998. 18

[107] Dola Saha, Siuli Roy, Somprakash Bandyopadhyay, Somprakash B, Tetsuro Ueda, and Shinsuke Tanaka. **An Adaptive Framework for Multipath Routing via Maximally Zone-Disjoint Shortest Paths in Ad**

hoc Wireless Networks with Directional Antenna. In *In IEEE Global Telecommunications Conference*, pages 226–230, 2003. 61

[108] Anees Shaikh, Jennifer Rexford, and Kang G. Shin. **Load-sensitive routing of long-lived IP flows**. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 215–226, New York, NY, USA, 1999. ACM. Available from: http://doi.acm.org/10.1145/316188.316225. 47

[109] Yi Shi, Y. Thomas Hou, Jia Liu, and Sastry Kompella. **How to correctly use the protocol interference model for multi-hop wireless networks**. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, pages 239–248, New York, NY, USA, 2009. ACM. Available from: http://doi.acm.org/10.1145/1530748.1530782. 23

[110] Dongseung Shin and Dongkyun Kim. **3DMRP: 3-Directional Zone-Disjoint Multipath Routing Protocol**. *IEICE Transactions on Information and Systems*, **E92-D**(4):620–629, 2009. 8, 61

[111] Jungmin So and Nitin H. Vaidya. **Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver**. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, pages 222–233. ACM, 2004. Available from: http://doi.acm.org/10.1145/989459.989487. 4, 27

[112] Anand Prabhu Subramanian, Himanshu Gupta, Samir R. Das, and Jing Cao. **Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks**. *IEEE Transactions on Mobile Computing*, 7:1459–1473, December 2008. Available from: http://dx.doi.org/10.1109/TMC.2008.70. 144, 156, 164, 179

[113] Prabhu Subramanian, Milind M. Buddhikot, and Scott Miller. **Interference aware routing in multi-radio wireless mesh networks**. In *In Proc. of IEEE Workshop on Wireless Mesh Networks (WiMesh*, pages 55–63, 2006. 45

[114] J. Tang, G. Xue, and W. Zhang. **Maximum Throughput and Fair Bandwidth Allocation in Multi-Channel Wireless Mesh Networks**. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications*. IEEE Computer Society, 2006. Available from: http://dx.doi.org/10.1109/INFCOM.2006.249. 10, 54, 106, 140, 142, 143

[115] Preetha Thulasiraman, Srinivasan Ramasubramanian, and Marwan Krunz. **Disjoint multipath routing to two distinct drains in a multi-drain sensor network**. In *INFOCOM '07: 26th IEEE International Conference on Computer Communications*, pages 643–651, 2007. 61

[116] Hiroshi Tokito, Masahiro Sasabe, Go Hasegawa, and Hirotaka Nakano. **Routing Method for Gateway Load Balancing in Wireless Mesh Networks**. In *ICN '09: Proceedings of the 2009 Eighth International Conference on Networks*, pages 127–132. IEEE Computer Society, 2009. Available from: http://dx.doi.org/10.1109/ICN.2009.21. 10, 107

[117] Thiemo Voigt, Adam Dunkels, and Torsten Braun. **On-demand construction of non-interfering multiple paths in wireless sensor networks**. In *In Proceedings of the 2nd Workshop on Sensor Networks at Informatik 2005*, pages 277–285, 2005. 8, 61

[118] S. Waharte and R. Boutaba. **Totally Disjoint Multipath Routing in Multihop Wireless Networks**. In *IEEE International Conference on Communications, ICC '06*, pages 5576–5581, 2006. 57

[119] Kui Wu and Janelle Harms. **Performance Study of a Multipath Routing Method for Wireless Mobile Ad Hoc Networks**. In *MASCOTS '01: Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 99, Washington, DC, USA, 2001. IEEE Computer Society. 8, 57, 60, 62

[120] Bin Xie, Yingbing Yu, Anup Kumar, and Dharma Parkash Agrawal. **Load-balancing and Inter-domain Mobility for Wireless Mesh Networks**. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 409–414, 2006. 10, 107

[121] Yuan Xue, Baochun Li, and Klara Nahrstedt. **Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-Based Approach**. *IEEE Transactions on Mobile Computing*, **5**(4):347–364, 2006. Available from: http://doi.ieeecomputersociety.org/10.1109/TMC.2006.52. 123

[122] Yaling Yang, Jun Wang, and Robin Kravets. **Interference-aware Load Balancing for Multihop Wireless Networks**. Technical report, University of Illinois at Urbana-Champaign, 2005. Technical Report UIUCDCS-R-2005-2526. Available from: http://www.cs.uiuc.edu/research/techreports.php?report=UIUCDCS-R-2005-2526. 104

[123] Yaling Yang, Jun Wang, and Robin H. Kravets. **Designing routing metrics for mesh networks**. In *IEEE WiMesh*, 2005. Available from: http://mobius.cs.uiuc.edu/system/files/wimesh05.pdf. 44

[124] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. **A Framework for Reliable Routing in Mobile Ad Hoc Networks**. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, **1**, pages 270–280, 2003. Available from: http://dx.doi.org/10.1109/INFCOM.2003.1208679. 61

Appendix A

Heuristic for fast load-balancing algorithm

This appendix is a study of the general problem of restricted job scheduling on identical machines. Many load-balancing problems in networking are instances of this problem, such as the gateway load-balancing problem addressed in chapter 4. A fast heuristic algorithm is proposed to solve it and its empirical performance is evaluated and compared against traditional and state of the art algorithms.

A.1 Introduction

Consider the general problem of assigning a set of users to machines with the objective of balancing the load on the machines. Each user has a given amount of required service which can only be provided by a *subset* of the machines. Given n users and m machines, the problem is to assign each user to a machine such that the maximum load on a machine is minimized. The load of a machine is the sum of loads of all users assigned to the machine. Machines have identical processing capabilities, i.e. the time to process a given load is the same on all machines.

This problem appears in multiple forms in network scenarios. One example is the assignment of users to servers, where a user can only be assigned to a subset of the servers. Reasons may include cost issues (e.g. distance to server) or because the user's job can only be processed by a particular type of server. Another example is the assignment of a node's traffic to a network "path". The node's load is related to the

amount of traffic it needs to send (e.g. total demand or number of flows). A path may refer to one of several independent network entities, such as individual links or sequences of links, base stations or gateways. In the studied model, a node's traffic may only be routed through a subset of all available paths. Reasons include that only some paths lead to the destination, only some paths may be reachable by the node (e.g. proximity to base station), or based on the cost of using the paths (distance or time to reach the destination via the path).

By balancing the load in the above mentioned scenarios, key performance metrics are expected to improve, such as lower service time, higher throughput and fairness.

This study considers the case where n and m are known in advance. That is, in a given instant the set of users, their load, and the set of machines is known. However, in network scenarios we can expect a high level of variability. At any time, the users and their load can change, which prompts the need for a new assignment (which may include the reassignment of previously allocated users). For this reason, a fast load-balancing algorithm is of special interest, to enable high responsiveness and adaptivity in a number of load-balancing scenarios.

This study considers the "off-line" version of the problem, in the sense that the current set of users and machines are known in advance. In the on-line version [11], there is no knowledge of the set of users. Users arrive one by one and are assigned to a machine in the order they arrive. This is an important limitation which leads to worse solutions, as we will explain later. Because the network conditions can be expected to remain stable for a period of time, it is best to apply an off-line algorithm given the current conditions, and recalculate the solution in the next time period. The period of time can be arbitrarily small as long as the responsiveness of the system allows it.

This appendix studies the load-balancing problem and finds it to be a special case of a job scheduling problem with unrelated machines, which is NP-hard. It proposes a fast heuristic algorithm to solve the problem, and evaluates its empirical performance, comparing it with a number of proposed algorithms. Results show that it can compete with the best algorithms in terms of the fitness of the solutions, while being more efficient.

The rest of the appendix is organized as follows. Section A.2 formalizes the load-balancing problem. Section A.3 briefly reviews related work. In section A.4 the proposed algorithm is explained. Section A.5 shows experimental results.

Table A.1: Load-balancing problem symbols.

$\mathbb{U}$	Set of users
$\mathbb{M}$	Set of machines
n	Number of users
m	Number of machines
l_u	Load of user u
α	User assignment function
$\alpha(u)$	Machine assigned to user u
$\delta_j(\alpha)$	Total load of machine j on assignment α
W	Total load in system: $\sum_{u \in \mathbb{U}} l_u$
S_u	Valid machines of user u
S	$\sum_{u \in \mathbb{U}} \lvert S_u \rvert$

A.2 Load-balancing problem

We consider the problem of assigning a set $\mathbb{U}$ of n users to a set $\mathbb{M}$ of m machines. The load of a user u is l_u. Machines are identical in the sense that a user's job incurs the same load on every machine. A user u can only be assigned to a subset of machines $S_u \subseteq \mathbb{M}$. We call this the set of *valid machines* of the user.

An assignment of users to machines is denoted by a function $\alpha : \mathbb{U} \mapsto \mathbb{M}$. We denote $\alpha(i) = j$ if user i is assigned to machine j. For any assignment α, the load δ_j on machine j is the sum of the load of all the users that are assigned to machine j, thus

$$\delta_j(\alpha) = \sum_{\alpha(u)=j} l_u \tag{A.1}$$

The objective is to obtain the assignment with least maximum load on a machine, i.e.

$$\min_{\alpha} \max_{j \in \mathbb{M}} \delta_j(\alpha) \tag{A.2}$$

Table A.1 lists all the symbols used in this appendix to explain the problem and algorithm.

This problem is equivalent to scheduling n independent jobs on m identical machines with no preemption. The users represent the jobs, and the weight of a job is the user's load. The objective is to find an assignment of jobs to machines that minimizes the

makespan of the schedule (i.e. the maximum load on a machine). The *restricted* model that this study considers, where jobs can only be assigned to a subset of machines, represents a special case of job scheduling on unrelated machines, or $R||C_{max}$ following the notation of Graham et al. [42]. This special case has the additional restriction that the processing time of a job on a machine is either the weight of the job or ∞ if the job cannot be processed on the machine. The job scheduling problem with unrelated machines was shown to be NP-hard. Lenstra et al. [69] proved that unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial-time approximation algorithm for the optimum schedule with approximation factor less than $\frac{3}{2}$.

A.3 Related work

The problem studied is a special case of job scheduling on unrelated machines, which is NP-hard.

The first published approximation algorithms for scheduling on unrelated machines were based on list scheduling [24, 49]. Many more methods have been proposed since then, with techniques ranging from combinatorial approaches with partial enumeration to integer programming with branch-and-bound and cutting planes [77, 82].

One of the best algorithms recently proposed for this problem was presented by Gairing et al. in [32]. A similar approach was previously considered in [30] for the special case of *restricted* machines considered in this work. The advantages of their approach is that it is a fairly simple combinatorial algorithm, which achieves an approximation factor of 2, and runs faster than previous methods with the same approximation guarantee.

This work develops a simple efficient algorithm for the problem of scheduling to minimize the makespan or maximum load on a machine. It develops heuristics for a List Scheduling (LS) based algorithm, adapted for the particular problem of restricted scheduling. LS is a well-known simple combinatorial algorithm often used for scheduling problems. Owing to its simplicity, the proposed approach is faster than previous methods.

A.4 Load-balancing algorithm

This work proposes a LS-based algorithm called RSA. LS is frequently used in scheduling problems due to its simplicity and the fact that any optimal schedule can be constructed with an appropriately chosen list. The key therefore relies on generating the list of jobs. To this end, a special order for jobs is defined.

Algorithm 17 LS-based algorithm.

```
1: Sort 𝕌 // Comparison sort based on Algorithm 18
2: for u in 𝕌 do
3:     α(u) ← arg min_{j∈S_u} δ_j(α)
4: end for
```

Algorithm 18 Job comparison function used for sorting.

```
1: function less_than(j, k) do
2:     if (|S_j| = 1) and (|S_k| > 1) then
3:         return true
4:     end if
5:     if (|S_j| > 1) and (|S_k| = 1) then
6:         return false
7:     end if
8:     if l_j/S_j > l_k/S_k then
9:         return true
10:    else
11:        return false
12:    end if
13: end function
```

The LS-based algorithm (Algorithm 17) works as follows: after sorting the list of users, it iteratively assigns each user to the current least loaded machine.

In the on-line version of the problem, the set of users is unknown and there is no control on the order on which they are assigned. Instead, users are assigned to a machine in the order they arrive. For example, Azar et al. [11] propose a deterministic algorithm for the on-line case where each new user is assigned to the current least loaded machine. Note that this behavior is the same as Alg. 17 with no specific order

of the users. The results of the next section clearly show that the order on which a LS algorithm assigns users is crucial for obtaining better solutions.

The order of users is defined as follows. Users with only *one* valid machine are selected first to be assigned before any other users (lines 2-7 of Algorithm 18). For all other cases (lines 8-12), a heuristic similar to Longest Processing Time (LPT) first is used. The proposed heuristic takes into account the number of valid machines of each job. The idea is that jobs with more valid machines should be chosen last, because they offer more opportunities to balance load, and these opportunities should not be used up too early. In addition, choosing the jobs with more load first also remains important, which leads to considering both factors in the decision.

Theorem 4. *The time complexity of the algorithm in the worst case is* $O(n \log n + S)$.

Proof. In line 1, the list of users is sorted by a comparison sort, requiring $O(n \log n)$.

The loop of lines 2-3: Line 3 requires traversing the valid machines of the current user. Because the average number of valid machines per user is $\frac{S}{n}$, the loop runs in

$$O(n\frac{S}{n}) = O(S)$$

□

A.5 Experimental results

This section presents computational results to study the performance of the algorithm. For comparison, a subset of previously proposed algorithms has been chosen. The goal is to compare against LS algorithms which are similar in simplicity and running time to the proposed algorithm, and with Gairing's algorithm for the reasons mentioned in section A.3. The following algorithms have been selected: LS algorithm using LPT heuristic; Algorithm 2 of Davis and Jaffe [24]; and algorithm of Gairing [30]. We call them LPT, DAVIS and GAIRING, respectively. The algorithm of Davis et al. was designed for the more general unrelated machine problem. We don't choose algorithms proposed by Ibarra [49] because most are equivalent or worse than LPT when applied to the restricted scheduling problem, and because the algorithms later proposed by Davis offer better approximation guarantees. The algorithm by Gairing was proposed for the restricted scheduling problem, studied in this appendix. We also test an obvious

modification of the LPT heuristic which first assigns users with $|S_j| = 1$. We call it LPTA.

Because the optimum is unknown, tests measure the approximation ratio to a lower bound of the optimum. The optimal lower bound is defined as:

$$C^{\lambda} = \max\{C^1_{max}, \lceil \frac{W}{m} \rceil\} \tag{A.3}$$

C^1_{max} is the makespan when only the users with *one* valid machine are assigned.

A problem instance is given by multiple parameters, listed in Table A.2. To generate problem instances, the first two parameters m and n take on fixed values (shown in Table A.2). We generate all possible pairs of (m, n) values. For each pair, 100 random problem instances are generated. In each instance, l_u is generated from a uniform distribution in interval $[1, 20]$. $|S_u|$ can be generated from two distributions $\mathcal{U}_1$ and $\mathcal{G}_1$. $\mathcal{U}_1$ is a uniform distribution in $[1, m]$ and $\mathcal{G}_1$ is a geometric distribution with $p = \frac{1}{3}$. To select S_u, we use two probability samples: $\mathcal{U}_2$ and $\mathcal{G}_2$. $\mathcal{U}_2$ generates indexes to the list of machines from a uniform distribution, and $\mathcal{G}_2$ from a geometric distribution with $p = \frac{4}{m}$. When using geometric distributions, if the generated value is above the domain upper bound, it is truncated to the upper bound.

Using $\mathcal{G}_1$, in roughly half the cases, $|S_u| = 1$ and $|S_u| = 2$, i.e. most jobs will have very few valid machines (one or two). This illustrates a particular case that can occur, where conventional algorithms perform poorly compared to algorithms which take into account the number of valid machines of each job.

With $\mathcal{G}_2$, the objective is to make a few machines more likely of being selected as valid machines for a job. The distribution depends on m, with the end result that the first 20-30% of the machines in the list are the ones most likely of being selected as valid machines. This generates test cases which have an inherent imbalance (most load is focused on a few machines), and are harder to solve by conventional algorithms.

Given the above, we can distinguish between four sets of problem instances:

- Model A: using $\mathcal{U}_1$ and $\mathcal{U}_2$
- Model B: using $\mathcal{G}_1$ and $\mathcal{U}_2$
- Model C: using $\mathcal{U}_1$ and $\mathcal{G}_2$
- Model D: using $\mathcal{G}_1$ and $\mathcal{G}_2$

Table A.2: Benchmark set parameters.

Parameter	Value
m	$[3, 20]$
n	$[1, 25m]$
l_u	uniform distribution in $[1, 20]$
$\|S_u\|$	distributions $\mathcal{U}_1$ and $\mathcal{G}_1$
S_u	probability samples $\mathcal{U}_2$ and $\mathcal{G}_2$

Each set contains a total of 517500 instances.

The results for each set are shown in Figure A.1. The x axis shows the number of machines and the y axis shows C/C^{λ} where C is the makespan calculated by the algorithms. Each point is the average of all scenarios which fall in that category. Confidence intervals are shown at 95% level.

In these tests, RSA is the best performing in some cases and in others performs similar to GAIRING. We can see that the obvious modification of the LPT heuristic of first assigning the jobs with $|S_j| = 1$ can substantially improve the solution. The heuristic of RSA of taking the number of valid machines of each user into account also contributes to a better solution, specially when machine sampling is not based on a uniform sample, i.e. some machines have more probability of being selected than others (models C and D). Note than in models B and D most jobs have a low number of valid machines (1 or 2). This contributes to a worse solution of the traditional LPT heuristic, but also means that the distance between RSA and LPTA is less in these cases.

A.5.1 Comparison of running times

These are the running times of the tested algorithms, according to Theorem 4 and [24, 30]:

- RSA, LPT: $O(n \log n + S)$
- DAVIS: $O(mn \log n)$
- GAIRING: $O(mS \log W)$

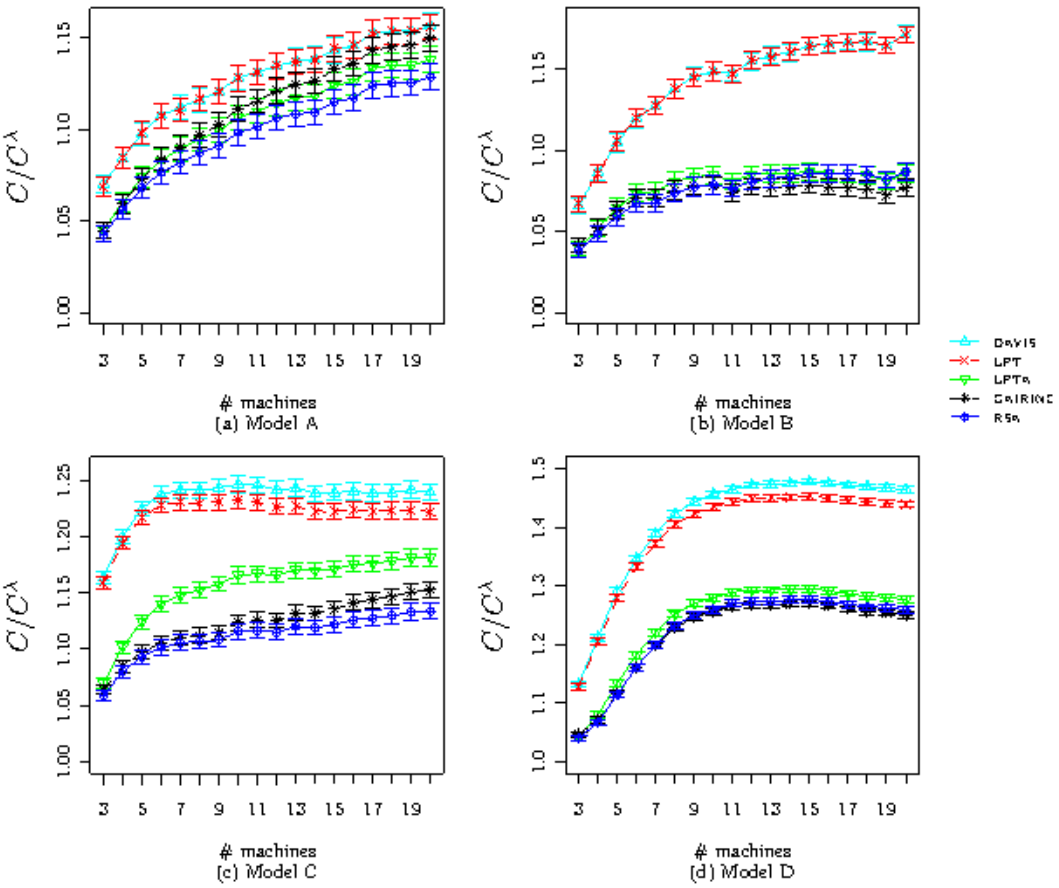


Figure A.1: Benchmark set computational results - The figures compare the approximation of the algorithms to the lower bound of the optimal makespan in models A, B, C and D.

A. HEURISTIC FOR FAST LOAD-BALANCING ALGORITHM

We compute an upper bound on the running time of GAIRING relative to RSA. Assuming $S = nm$:

$$\text{if } n \log n > S: \qquad \frac{mS \log W}{n \log n} = m^2 \frac{\log W}{\log n}$$
$$\text{if } S > n \log n: \qquad \frac{mS \log W}{S} = m \log W$$

GAIRING can take up to $O(m^2 \frac{\log W}{\log n} + m \log W)$ times longer than RSA.

Similarly, we compute an upper bound on the running time of DAVIS relative to RSA. Assuming $S = n$:

$$\text{if } n \log n > S: \qquad \frac{mn \log n}{n \log n} = m$$
$$\text{if } S > n \log n: \qquad \frac{mn \log n}{S} = m \log n$$

DAVIS can take up to $O(m \log n)$ times longer.