Interference-aware Load Balancing for Multihop Wireless Networks

Yaling Yang, Jun Wang and Robin Kravets
University of Illinois at Urbana-Champaign
{yyang8, junwang3, rhk}@cs.uiuc.edu

***Abstract*— Load balancing is critical for improving performance in wireless mesh networks. The unique characteristics of mesh networks, such as static nodes and the shared nature of the wireless medium, invalidate existing solutions from both wired and wireless networks and introduce new challenges for providing load balancing. In this paper, we focus on addressing these challenges. We first formulate the objective of load balancing in mesh networks and provide a theoretical solution to optimally achieve this objective. Then, we investigate some existing practical approaches to load balancing in mesh networks and show that none of them sufficiently address these challenges and some may even cause non-optimal paths and forwarding loops. In response, we propose a new path weight function, called MIC, and a novel routing scheme, called LIBRA, to provide interference-aware and multi-channel/multi-radio aware load balancing for mesh networks, while still ensuring routing optimality and loop-freedom. We use extensive simulations to evaluate our scheme by comparing it with both the theoretical optimal solution and existing practical solutions. The results show close-to-optimum performance and indicate that LIBRA is a good candidate for load balancing and routing in wireless mesh networks.**

I. Introduction

The main goal of any routing protocol is to maintain effective communication. However, the specific characteristics of a network may emphasize specific aspects of routing protocol design. For example, the mobility of nodes in ad hoc networks demand the design of protocols that can efficiently maintain connectivity. However, for any static network (e.g., wireless mesh networks [1], [2] and wired networks), load balancing is necessary to avoid hot spots and increase network utilization because poor routes may exist for a long time in a static network and result in congestion and inefficient use of network resources. While load-balancing has been studied in wired networks, the wireless communication channels in mesh networks introduce new challenges. Therefore, the goal of our research is to design efficient load-balanced routing schemes to achieve good performance, such as high throughput and low packet delay, in mesh networks

Mesh networks, motivated by wireless neighborhood networks [1], [2], are composed of static wireless nodes that have ample energy supply. Each of these wireless nodes can be equipped with multiple radios, called a *multi-radio/multi-channel node*, and each of the radios can be configured to a different channel to enhance network capacity. All wireless nodes cooperatively route each other's traffic to the Internet through one or more Internet Transit Access Points (TAPs), which are gateways to the Internet. Nodes may also communicate with each other directly through the mesh network without going through TAPs. Given the static nature of mesh nodes, the main challenge for routing protocols is to support good network performance by balancing the load across all nodes in the network. Although load balancing has been addressed in different types of networks, mesh networks introduce a new combination of wireless nodes in a relatively static network. This combination invalidates the application of solutions targeted at other networks to mesh networks. Essentially, solutions for wired networks do not address the characteristics of the wireless medium and ad hoc networks do not optimize for static nodes.

The shared nature of the wireless channel is the main characteristics of wireless networks not addressed by wired solutions. Since nodes transmitting on the same wireless channel compete for the shared medium, both *inter-flow interference* (interference between nodes carrying different flows) and *intra-flow interference* (interference between nodes carrying the same flow) may affect traffic loads on mesh nodes. The interference between mesh nodes is further complicated by multi-radio/multi-channel nodes since each radio at a may interfere with and affect the load on a different set of neighboring nodes. None of these complex characteristics of interference exist in wired networks. Hence, load-balanced routing schemes for wired networks are not suitable for mesh networks.

Load-balanced routing schemes proposed for ad hoc networks (e.g., MCR [3], LBAR [4], and DLAR [5]) are all based on on-demand routing. While the on-demand route discovery provides strong resistance to mobility-caused link breaks, the long expected lifetimes of links in mesh networks make on-demand route discovery both redundant and very expensive in terms of control message overhead. Hence, load-balanced routing protocols for ad hoc networks are also not appropriate for mesh networks.

Since load-balanced protocols designed for other networks are not efficient in mesh networks, in recent years, protocols that are dedicated for mesh networks have started to emerge (e.g., ETX [6], ETT [7], WCETT [7] and [8]). Unlike on-demand routing protocols in ad hoc networks, these load-balancing protocols for mesh networks are based on proactive routing to support the static nature of mesh nodes. They send route update messages periodically with long update intervals or when the network topology changes. However, none of these protocols captures both the intra-flow and inter-flow interference in mesh networks. Some of the existing schemes may even have the potential for creating routing loops (e.g., WCETT [7]) and network instability (e.g., [8]) (See Section

III for details).

Due to the limitations of existing schemes, new protocols need to be designed to provide effective load blanacing in mesh networks, which is the focus of our research. Different from existing methods, our work addresses both intra-flow and inter-flow interference and does not create network instability or routing loops. There are four unique contributions of our work. First, based on the characteristics of mesh networks, we formulate the goal for load balancing and show how this goal can be optimally achieved in theory. Second, we present the requirements for designing routing protocols to achieve this goal and show that existing protocols have limitations in satisfying these requirements. Third, we design a new routing metric, called *Metric of Interference and Channel-switching (MIC)*, that captures the shared nature of wireless channels and exploits the extra resources available from multi-radio/multi-channel nodes. Fourth, based on this new routing metric, MIC, we design a new proactive routing protocol, called LIBRA (**L**oad and **I**nterference **B**alanced **R**outing **A**lgorithm), that satisfies the requirements for load-balanced routing protocols for mesh networks. Our simulation results show that LIBRA's performance is close to the theoretical optimum for load balancing and is significantly better than several existing approaches.

The reminder of the paper is organized as follows. In Section II, we present the goal of load balancing in wireless mesh networks and discuss how to optimally achieve this goal in theory. In Section III, we present the basic requirements for designing heuristic solutions for mesh networks and show the limitations of several existing heuristic solutions in satisfying these requirements. Section IV introduces our new routing metric, MIC. In Section V, we propose our new routing protocol, LIBRA. In Section VI, we show how to extend LIBRA to consider different interference ranges. In Section VII, we show how to extend LIBRA to consider wireless cards that are able to perform fast dynamic channel reconfiguration. In Section VIII, we evaluate the performance of LIBRA by comparing it with several existing routing schemes through simulation. In Section IX, we conclude our work and discuss future directions.

II. Optimal Load Balancing

The objective of load balancing is essentially to distribute traffic among different paths to avoid creating congested areas and improve network performance. Although there is extensive research about load balancing in wired networks, the differences between wired links and the wireless medium change the objective of load balancing for mesh networks. Hence, in this section, we formulate the objective of load balancing for mesh networks and show how to optimally achieve this objective in theory.

To formulate the objective of load balancing for mesh networks, we make the following assumptions, which are also used for the rest of the paper except Section VII. We assume that a mesh node is equipped with one or more radios, each radio is pre-configured to a certain channel, and radios configured to different channels do not interfere with each other. If a node has multiple radios, these radios are configured to different channels. Several existing algorithms [9] can be used to pre-configure the channels for the radios and the discussion of channel assignment algorithms is beyond the scope of this paper. This assumption is based on the fact that channel reconfiguration is very slow in current wireless cards so that it is not practical for a radio to switch its channel configuration during routing. For future wireless cards that are capable of fast channel reconfiguration, in Section VII, we present an extension of our scheme to capture load balanced routing for dynamic channel reconfiguration.

With these assumptions, the objective of load balancing for mesh networks is very different from wired networks. In wired networks, since each link has its own dedicated bandwidth and traffic on one link does not consume bandwidth from other links, the major goal of load balancing is to reduce link utilization. However, in mesh networks, although a node's connections with its neighbors are also referred to as *wireless links*, the bandwidth of these wireless links cannot be viewed as independent network resources. The dependency comes from the fact that if a node i is communicating with a neighbor using radio r configured to channel c, node i cannot communicate concurrently with other neighbors using the same radio/channel. In addition, when node i's neighboring nodes are actively communicating on channel c, node i cannot use channel c for either receiving or transmitting. Therefore, for wireless networks, the objective for load balancing should be defined in terms of the utilization of wireless *channels* rather than the utilization of wireless *links*. Hence, we define the utilization of each wireless channel configured at a node i as the fraction of time the channel is busy. A channel is busy in two cases. First, the channel is used by node i for communicating with its neighbors. Second, the channel is blocked because node i's neighboring nodes are actively communicating using this channel. Since transmitting packets on highly utilized channels can result in high delay and low throughput, the objective of load balancing in wireless mesh networks should be to avoid over-utilization of channels at every node.

To calculate the utilization of channel c at a node i, it is necessary to define the set of interfering wireless links of channel c at node i. This set, $R_i(c)$, includes any wireless link that uses channel c and has at least one of its end points inside node i's carrier sensing range (including node i itself). With this definition of $R_i(c)$, the utilization of channel c at node i can be defined as:

$$u_i^c = \sum_{(k,l,c)\in R_i(c)} \sum_{t\in V} \frac{f_{klc}^t}{b_{klc}}, \tag{1}$$

where (k,l,c) represents a wireless link from node k to node l using channel c and V is the set of all nodes in the network. f_{klc}^t is the amount of traffic to destination node t that traverses link (k,l,c) and b_{klc} is the physical transmission rate of the link. Essentially, $\sum_{t\in V} \frac{f_{klc}^t}{b_{klc}}$ represents the amount of time that the transmissions on link (k,l,c) block channel c of node i. Similar to the calculations used in [9], [10], our definition of channel utilization is a conservative simplification since it ignores the fact that several neighboring nodes in $R_i(c)$

may transmit concurrently if they are out of each other's interference range. However, this omission is supported by the fact that there is no synchronization between the transmissions of nodes that are not in each other's interference range. Therefore, in the worst case, these neighboring nodes transmit sequentially, which is captured in our definition of u_i^c through the summation of $\sum_{t \in V} \frac{f_{klc}^t}{b_{klc}}$. We choose to use this worst-case-based calculation for the tractability of the optimal load balancing problem. Another formulation proposed in [11] is a best-case-based simplification, which omits the possibility of sequential transmissions of neighboring nodes by assuming an omnipotent MAC layer scheduler that achieves maximum concurrent transmissions among all nodes. This simplification, however, causes the formulation of optimal load-balancing to be NP-complete.

With our definition of channel utilization, the objective for load balancing in mesh networks is to balance the channel utilization of the nodes in the network. This objective can be formulated as a solvable problem as follows:

$$\begin{aligned}
&\textbf{Minimize } \Phi = \sum_{i \in V} \sum_{c \in C(i)} \varphi(u_i^c), \\
&\textbf{subject to} \\
&\sum_{c \in C(i)} \sum_{\{j:(i,j,c) \in E\}} f_{ijc}^t \\
&\quad - \sum_{c \in C(i)} \sum_{\{j:(j,i,c) \in E\}} f_{jic}^t = d_{it}, \quad i, t \in V, i \neq t, \\
&\quad u_i^c = \sum_{(k,l,c) \in R_i(c)} \sum_{t \in V} \frac{f_{klc}^t}{b_{klc}}, \quad i \in V, c \in C(i), \\
&\qquad\qquad f_{ijc}^t \geq 0, \quad (i,j,c) \in E, t \in V,
\end{aligned} \tag{2}$$

where $C(i)$ is the set of channels in node i, E is the set of wireless links in the network and d_{it} is the amount of traffic targeted to destination node t and generated by source node i. $\varphi(\cdot)$ is the cost function of channel utilization, whose definition may vary based on the needs of the system. A commonly used cost function is a piece-wise linear function [12], [13]:

$$\varphi(0) = 0 \text{ and}$$
$$\varphi'(x) = \begin{cases} 1 & \text{for} \quad 0 \leq x < 1/3 \\ 3 & \text{for} \quad 1/3 \leq x < 2/3 \\ 10 & \text{for} \quad 2/3 \leq x < 9/10 \\ 70 & \text{for} \quad 9/10 \leq x < 1 \\ 500 & \text{for} \quad 1 \leq x < 11/10 \\ 5000 & \text{for} \quad 11/10 \leq x < \infty. \end{cases} \tag{3}$$

Figure 1 shows the evolution curve of this cost function. The basic idea is that it is cheap to send flows over a channel with low utilization. As the channel utilization increases, the cost becomes more expensive since packet delay, delay jitter and packet loss ratio at the channel increase and become more sensitive to traffic bursts. As the channel utilization approaches 1.0, the cost function imposes very heavy penalties to prevent the channel from being overloaded. With this cost function, the optimal load-balanced routing problem in Equation (2) becomes a linear program that can be solved in polynomial time to give the optimal traffic allocation $\{f_{ijc}^t\}$ on each link (i, j, c).

Although optimal load balancing can be obtained theoretically by solving Equation (2), this theoretical approach is not practical to use in mesh networks. Essentially, formulating the linear program requires centralized knowledge of the traffic demands between each source and destination pair, which changes frequently and is very difficult to acquire. In addition, the optimal routing solution produced by the linear program requires the ability to split traffic arbitrarily among all paths between a source and a destination. Unfortunately, arbitrary traffic splitting is hard to achieve in reality since it introduces high complexity into the routing mechanism and may also cause out-of-order delivery of TCP traffic [14]. Due to these implementation issues for optimal load balancing, heuristic routing protocols that do not require knowledge of traffic demands and forward all traffic from one flow on the same path must be designed. The performance of these heuristic routing protocols can be compared with optimal load balancing for evaluation purposes.

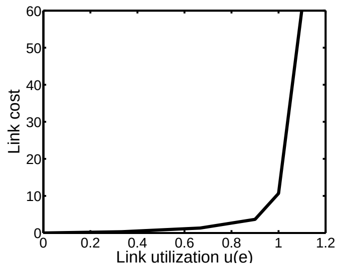


Fig. 1. Link cost $\varphi(u(e))$ as function of link utilization $u(e)$.

III. Heuristic-based Load Balanced Routing

Because of the extreme difficulty of using optimal load balancing in mesh networks, heuristic routing protocols need to be designed to approach the performance of optimal load balancing. The key for a heuristic protocol to achieve good performance is the design of its path weight function (also called routing metrics). In the reminder of the section, we first review the types of routing protocols that may be used in mesh networks and then introduce the theories regarding the requirements of designing path weight functions for these protocols. Finally, we discuss four existing path weight functions in mesh networks and their limitations in satisfying these requirements.

A. Types of Routing Protocols

Depending on how packets are routed, routing protocols in mesh networks can be divided into two categories: source routing and hop-by-hop routing, each with different costs in terms of message overhead and management complexity.

Source routing, such as LQSR [7] and MCR [3], put the entire path of a flow in the packet headers and intermediate nodes forward packets based on these paths. Considering that the packet size in mesh networks is usually very small to cope with the high bit error rate of wireless channels, putting the entire path in the packet header imposes expensive message overhead.

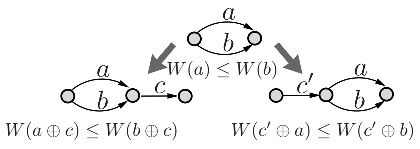
Hop-by-hop routing, on the other hand, only puts the destination address in the packet header and intermediate nodes forward packets based on the destination address. Due to its simple forwarding scheme and low message overhead, hop-by-hop routing is dominant in wired networks. We believe that similar reasons also make hop-by-hop routing the most preferable for mesh networks. However, despite its benefits, hop-by-hop routing requires careful design of path weight functions to ensure that its performance approaches optimal load balancing.

B. Requirements of Routing Protocols

To ensure good performance from heuristic routing protocols, path weight functions must satisfy four requirements. First, the path weight functions must not change frequently to ensure the stability of the network. Second, the path weight functions must capture the characteristics of mesh networks to ensure that minimum weight paths have good performance. Third, the path weight functions must ensure that minimum weight paths can be found by efficient algorithms with polynomial complexity. Finally, the path weight functions must ensure that forwarding loops are not formed by routing protocols. In this section, we introduce the theories regarding these four requirements.

1) Stable Path Weight Function: Frequent path weight changes are very harmful to the performance of any network. Frequent changes can create a high volume of route update messages. They can also disrupt normal network operations since routing protocols may not converge under frequent route updates. Load-sensitive path weight functions, where path weights are related to the traffic loads on the paths, have a high risk of creating network instability if traffic variations on the paths are large and irregular. Experiments conducted in wired networks have already demonstrated such effects [15], which have prevented the deployment of load-sensitive path weight functions in wired networks [16]. For mesh networks, we believe that this problem may be even worse. Since the Internet has a very large number of users, multiplexing smooths out traffic variations and reduces the number of route changes. Mesh networks, however, have a much smaller scale. Hence, link traffic variations may be large and irregular, making it very difficult to use load-sensitive path weight functions while maintaining the stability of the network. Therefore, load-sensitive path weight functions, such as the scheme proposed by Raniwala et. al [8], are not suitable for mesh networks. On the other hand, topology-dependent path weight functions that base path weights on topological properties (e.g., hop count, link capacity, etc.) are more stable and, hence, are preferable for mesh networks.

2) Capturing Network Characteristics: Due to the shared nature of the wireless medium, both *intra-flow interference* and *inter-flow interference* exist in mesh networks. Intra-flow interference refers to the fact that nodes on the path of a same flow compete with each other for channel bandwidth when they transmit on the same channel. Inter-flow interference happens when neighboring nodes carrying different flows compete for channel bandwidth when they transmit on the same channel. Both intra-flow and inter-flow interference affect the load on nodes. Hence, to balance network load, path weight functions must capture both intra-flow and inter-flow interference. In addition, a node may transmit to different neighbors using different transmission rates due to the rate switching ability of current wireless cards and experience different packet loss ratios. Such differences between the wireless links must also be captured by effective path weight functions.



a
b
$W(a) \le W(b)$
a
b
c
$W(a \oplus c) \le W(b \oplus c)$
c'
a
b
$W(c' \oplus a) \le W(c' \oplus b)$

Fig. 2. Example of isotonicity

3) Calculation of Minimum Weight Paths: Both source routing and hop-by-hop routing rely on efficient algorithms, such as Bellman-Ford or Dijkstra's algorithms, to compute routes. To ensure that Bellman Ford and Dijkstra's algorithms can find minimum weight paths, a fundamental property of path weight functions, called *isotonicity*, must be satisfied as shown in the work of Sobrinho [17], [18].

Assuming that for any path a, its weight is defined by a weight function $W(a)$ and the concatenation of two paths a and b is denoted by $a \oplus b$, the definition of isotonicity is:

Definition 1: A weight function $W(\cdot)$ is isotonic if $W(a) \le W(b)$ implies both $W(a \oplus c) \le W(b \oplus c)$ and $W(c' \oplus a) \le W(c' \oplus b)$, for all a, b, c, c' (See Figure 2).

Given this definition of isotonicity, the following relationship exists between the isotonicity property and the optimality of Bellman-Ford and Dijkstra's algorithms [17], [18]:

Theorem 1: Isotonicity is a sufficient and necessary condition for both Bellman-Ford and Dijkstra's algorithm to find minimum weight paths.

Theorem 1 implies that if a path weight function is not isotonic, routing protocols based on Bellman-Ford or Dijkstra's algorithm may not find the minimum weight path between two nodes. The resulting sub-optimal paths may degrade network performance.

4) Loop-free Routing: Since in source routing, the source nodes have complete control over the path of flows, using either Bellman-Ford or Dijkstra's algorithm results in loop-free paths. However, for non-isotonic weight functions in hop-by-hop routing, the algorithm used to calculate routes determines whether routing loops may exist. Specifically, for distance vector routing, which is a hop-by-hop routing protocol based on the Bellman-Ford algorithm, routing loops cannot be created even if path weight functions are not isotonic. However, for link-state routing, which is a hop-by-hop routing protocol combined with Dijkstra's algorithm, Sobrinho's work [17] shows that loop-free forwarding requires isotonicity.

Theorem 2: If Dijkstra's algorithm is used in hop-by-hop routing, isotonicity is a sufficient and necessary condition for loop-free forwarding.

Theorem 2 reveals an important limitation of non-isotonic path weight functions: non-isotonic path weight functions are not usable for link-state routing, which implies that either source routing or distance vector routing must be used. Unfortunately, due to the lack of central management of mesh networks and the unreliable nature of wireless links, it is expected that link breaks or link quality changes in mesh networks can be frequent. In such environments, distance-vector routing converges much slower than link-state routing and can potentially degrade network stability. In addition, as discussed in Section III-A, link-state routing is also better than source routing due to its small message overhead. Hence, it is a non-trivial drawback that non-isotonic weight functions cannot be used in link-state routing.

Hence, to ensure that minimum weight and loop-free routing is achieved in mesh networks, it is very important to check the isotonicity of routing metrics. Therefore, in Section III-C, we use this isotonicity property to examine some existing routing protocols for wireless mesh networks and in Section IV, we use it to check our own protocol.

C. Existing Solutions in Mesh Networks

To satisfy the four requirements of path weight functions, these functions must be isotonic, topology-dependent and must capture the network characteristics. In this section, we discuss four existing path weight functions that have been used in mesh networks and which of these three required properties they lack. These four path weight functions are: hop count, ETX [6], [19], ETT [7], WCETT [7]. All four path weight functions are topology-dependent and each path weight function was proposed as an improvement over the previous one.

1) Hop Count: Hop count is the most commonly used path weight function in existing routing protocols such as DSR [20], AODV [21], DSDV [22] and GSR [23]. Since a hop count weight function is isotonic, efficient algorithms can find loop-free paths with minimum hop counts. However, hop count does not consider the differences of the transmission rates and packet loss ratios between different wireless links. Hence, its performance may not always be good.

2) Expected Transmission Count (ETX): ETX, proposed by De Couto et al. [6], [19], is an isotonic path weight function. It captures the different packet loss ratios at wireless links by measuring the expected number of MAC layer transmissions (ETX) needed to send a unicast packet on a link. The weight of a path is defined as the summation of the ETX's of all links along the path. However, ETX does not consider that different links may have different transmission rates.

3) Expected Transmission Time (ETT): The ETT path weight function, proposed by Draves et al. [7], improves ETX by considering the differences in link rates. It defines the weight of a path p as the summation of the ETT's of the links on the path. The ETT of a link l is defined as:

$$ETT_l = ETX_l \frac{s}{b_l}, \tag{4}$$

where b_l is the transmission rate of link l and s is the packet size. Essentially, ETT_l is the time for a successful transmission of a packet at link l. Similar to ETX, ETT is also isotonic. However, a common drawback of ETT and of all of the path weight functions discussed so far is that they do not fully capture the intra-flow and inter-flow interference in the network. For example, ETT may choose a path that only uses one channel, even though a path with more diversified channels has less intra-flow interference and hence higher throughput.

4) Weighted Cumulative ETT (WCETT): To reduce intra-flow interference, WCETT [7] was proposed by Draves et al. [7] to reduce the number of nodes on the path of a flow that transmit in the same channel. For a path p, WCETT is defined as:

$$WCETT(p) = (1-\beta) \sum_{\text{link } l \in p} ETT_l + \beta \max_{1 \le j \le k} X_j, \tag{5}$$

where β is a tunable parameter subject to $0 \le \beta \le 1$. X_j is the number of times channel j is used along path p and captures the intra-flow interference.

WCETT has two limitations. The first limitation, which is common for all of the existing path weight functions, is that it does not explicitly consider the effects of inter-flow interference. Therefore, WCETT may route flows to dense areas where congestion is more likely and may even result in starvation of some nodes due to congestion.

Besides the lack of consideration of inter-flow interference, WCETT has another unique limitation: there is no efficient algorithm that can calculate the minimum weight path based on WCETT since it is not isotonic. Figure 3 depicts a simple topology that shows that WCETT is not isotonic. In this figure, two numbers are associated with each link, the ETT and the channel number (CH), respectively.

Assuming β in the definition of WCETT (see Equation (5)) is set to 0.5, the minimum weight path from S_1 to T should be $S_1 \rightarrow B \rightarrow T$. However, due to the non-isotonic property of WCETT, when node S_1 uses Dijkstra's algorithm to calculate its path to node T, node S_1 incorrectly chooses $S_1 \rightarrow S_2 \rightarrow C \rightarrow D \rightarrow T$ as the minimum weight path, indicated as the dotted arrows in Figure 3. This is because when running Dijkstra's algorithm at node S_1, the minimum weight path from node S_1 to node B is found to be $S_1 \rightarrow A \rightarrow B$. $S_1 \rightarrow B$, hence, is eliminated from Dijkstra's algorithm's future consideration, although $S_1 \rightarrow A \rightarrow B \rightarrow T$ has a larger weight than $S_1 \rightarrow B \rightarrow T$. This incorrect early discard of $S_1 \rightarrow B$ causes Dijkstra's algorithm to fail to find the minimum weight path $S_1 \rightarrow B \rightarrow T$ from node S_1 to T.

If a link state protocol based on WCETT is used, this incorrect minimum weight path between S_1 and T can cause forwarding loops. When node S_2 calculates its path to T, Dijkstra's algorithm correctly indicates that $S_2 \rightarrow S_1 \rightarrow B \rightarrow T$ is the minimum weight path, depicted as the shadowed arrows in Figure 3. Since S_1 has the incorrect minimum weight path, any packets destined to T are forwarded by S_1 to S_2. S_2 immediately forwards the packets back to S_1 again. Hence, a forwarding loop is formed between S_1 and S_2.

Similar to Dijkstra's algorithm, routing protocols based on the Bellman-Ford algorithm (e.g., distance-vector routing) may not find optimal paths based on WCETT either. Using the same example in Figure 3, since node B's minimum distance to node S_1 is the weight of $B \rightarrow A \rightarrow S_1$, node B only tells its neighbors about the weight of this path. Hence, node T does not have a chance to check the weight of $T \rightarrow B \rightarrow S_1$, which is the correct minimum weight path. Therefore, node T incorrectly sets its distance to S_1 as the weight of $T \rightarrow D \rightarrow C \rightarrow S_2 \rightarrow S_1$ and forwards any packets for S_1 to node D.

Because of WCETT's lack of isotonicity, there is no efficient algorithm with polynomial complexity to calculate minimum weight paths. In addition, the non-isotonicity of WCETT makes it unusable for link-state routing. To ensure loop-free routing, WCETT can only be used in source routing such as LQSR [7] or distance-vector routing. This limitation is non-trivial since both source routing and distance vector routing have significant drawbacks compared to link-state routing (See Section III-B.4).

Because of the limitations of existing path weight functions, we next propose a novel path weight function, MIC, that captures the characteristics of mesh networks. Combined with our LIBRA protocol in Section V, MIC ensures that minimum weight paths can be found by both Bellman-Ford and Dijkstra's algorithms and no forwarding loop can be formed in link-state routing.

IV. New Path Weight Functions

In this section, we propose a novel topology-dependent path weight function, called *Metric of Interference and Channel-switching (MIC)*, which is composed of two metrics: *Interference-aware Resource Usage (IRU)* and *Channel Switching Cost (CSC)*. The two metrics of MIC represent different characteristics of mesh networks. The IRU metric captures the effects of inter-flow interference and the differences in the transmission rates and loss ratios of wireless links, while the CSC metric captures the impact of intra-flow interference. By combining these two metrics, MIC balances CSC and IRU and avoids the drawbacks of existing metrics. Integrated with our novel routing protocol, LIBRA, in Section V, MIC can be used for both distance vector and link state routing. In this section, we first discuss IRU and CSC and then we show how to combine them to form the path weight function MIC. We also show that MIC is not isotonic. However, in Section V, we show how we can decompose MIC into isotonic link weight assignments in a virtual network and hence efficient algorithms can be used to find minimum weight paths.

A. Interference-aware Resource Usage (IRU)

To capture the differences in the transmission rates and loss ratios of wireless links as well as the inter-flow interference, the IRU metric is designed as follows:

$$IRU_{ij}(c) = ETT_{ij}(c) \times |N_i(c) \bigcup N_j(c)|, \tag{6}$$

where $N_i(c)$ is the set of neighbors that node i interferes with when it transmits on channel c. $|N_i(c) \bigcup N_j(c)|$ is the total number of nodes that may be interfered with by the transmission activities (e.g., RTS-CTS-DATA-ACK handshake) between Node i and Node j over channel c. $|N_i(c) \bigcup N_j(c)|$ is introduced to capture inter-flow interference since when a flow is transmitting a packet on link (i, j) over channel c, no neighboring node in $N_i(c) \bigcup N_j(c)$ can use channel c. $ETT_{ij}(c)$, the expected transmission time, is used to capture the differences in transmission rates and loss ratios of links. The overall physical meaning of $IRU_{ij}(c)$ is the aggregated channel time of all nodes consumed by the transmissions of the flow at link (i, j, c) Hence, finding minimum weight paths in terms of IRU path weights essentially minimizes every flow's network resource usage. Therefore, the overall load on the network is reduced and the performance of the network in terms of both delay and throughput is improved. Figure 4 shows a simple example of how IRU can improve the performance of a network. Assuming that ETT_{AD} is slightly smaller than ETT_{AB}, by using ETT metrics, node A would route its traffic to node C through node D. However, since nodes D and E are in each other's interference range, nodes D and E may become hot spots in the network if node E starts to transmit to node F. On the other hand, using IRU, node A chooses $A \rightarrow B \rightarrow C$, eliminating the undesirable contention between nodes E and D.

An important implementation issue of IRU is the estimation of $N_i(c)$. Since mesh networks are static, existing research results have shown that it is possible to measure whether two nodes are in each other's interference range at the time when the network is established. A simple measurement technique proposed by Agarwal et. al [24] exploits the fact that if two nodes are in each other's interference range, their carrier-sensing mechanisms prevent them from transmitting simultaneously. Therefore, if the two nodes start to broadcast consecutive packets at the same time, the transmission rate of each of the nodes should be much smaller than the transmission rate if only one node is broadcasting. Hence, by simply measuring the broadcasting rates of two nodes, it can be determined if the two nodes are in each other's interference range.

B. Channel Switching Cost (CSC)

Although IRU can be used to reduce inter-flow interference, it cannot exploit the multi-channel capabilities of nodes to reduce intra-flow interference. Among two paths that have the same IRU weight, the channel diversified path (i.e., nodes on the path use different channels to transmit to their next hops) can have much less intra-flow interference than the path that always uses the same channel. To capture this, we introduce another path metric, called the *Channel Switching Cost (CSC)*.

To understand CSC, consider a node X, which is on a path of a flow and is equipped with multiple radios, each configured to a different channel. To eliminate the intra-flow interference between node X and its previous hop, $prev(X)$, node X needs to transmit to the next hop node, $next(X)$, using a different channel from the channel it uses to receive from $prev(X)$ (See Figure 5). To capture this, denoting $CH(X)$ as the channel that node X uses to transmit to $next(X)$, the CSC of node X is:

$$CSC_X = \begin{cases} w_1 & \text{if } CH(prev(X)) \neq CH(X) \\ w_2 & \text{if } CH(prev(X)) = CH(X), \end{cases} \tag{7}$$

$$0 \leq w_1 < w_2, \tag{8}$$

where the relationship $w_2 > w_1$ captures the fact that due to intra-flow interference, using the same channel at node X and $prev(X)$ imposes a higher cost than using different channels. w_1 is associated with costs caused by using multiple radios simultaneously. Since such costs are usually very small for nodes in mesh networks, we set $w_1 = 0$. However, in environments where such costs cannot be omitted, such as energy costs for power limited devices, w_1 can be set to represent such costs. Since using the same channel for two consecutive hops can halve the throughput and dramatically

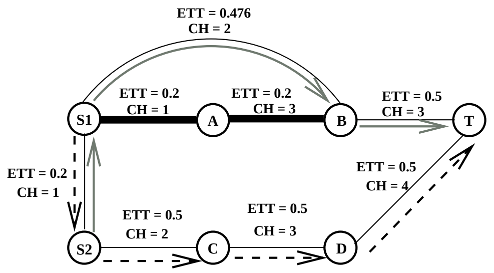


Fig. 3. Non-isotonicity of WCETT

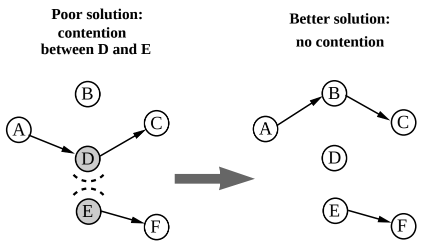

Poor solution: contention between D and E
Better solution: no contention
A
B
C
D
E
F

Fig. 4. Interference-aware routing

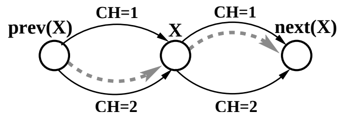


Fig. 5. Example of CSC

increase the delay of a flow, we let $w_2 \gg w_1$. In our simulations in Section VIII, we vary the values of w_2 from 0.3 to 5 and there is no significant impact of w_2 on the performance of our LIBRA algorithm introduced in Section V.

Note that the design of the CSC metric in Equation (7) only considers the intra-flow interference between two consecutive nodes on the path of a flow. The extension of the CSC metric to consider intra-flow interference between nodes that are further away is discussed in Section VI.

C. Metric of Interference and Channel-switching (MIC)

To capture all of the characteristics of mesh networks, we combine IRU and CSC to form a new path weight function called *Metric of Interference and Channel-switching (MIC)* as follows.

$$MIC(p) = \alpha \sum_{\text{link } l \in p} IRU_l + \sum_{\text{node } i \in p} CSC_i, \qquad (9)$$

where p stands for a path in the network. Essentially, the IRU component in the weight function captures how much a flow along p utilizes the overall available bandwidth in the network and is aimed at decreasing the channel utilization of the network. The CSC component represents intra-flow interference, which captures the performance of flows routed through p, and is aimed at decreasing the delay and increasing the throughput of individual flows. These two aims, in some cases, may be conflicting. For example, there may exist two paths p_1 and p_2 between a pair of nodes. p_1 has more diversified channels and hence has higher throughput. On the other hand, p_2 consumes less wireless channel time since it goes through more high bandwidth links, although these links may use the same channel. The positive factor α, hence, is introduced to represent the trade-off between these two benefits. A large α means that balancing the load on the network is more important than obtaining good per-flow performance. A smaller α means that more concern is given to per-flow performance, even if traffic load is not perfectly balanced and some neighboring nodes of flows may be starved due to inter-flow interference.

In our simulations, we set α as a trade-off value as follows.

$$\alpha = \frac{1}{N \times \min(ETT)}, \qquad (10)$$

where N is the number of nodes in the network and $\min(ETT)$ is the smallest ETT in the network, which can be estimated based on the lowest transmission rate of wireless cards. By combining this α with Equation (9), $\min(ETT)$ scales up the ETT in IRU (See Equation (6)) to be larger than 1 so that IRU is around the same value range as our settings of (CSC). N in Equation (10) normalizes the $|N_i(c) \bigcup N_j(c)|$ part in IRU, which is the number of neighbors the flow interferes with, to the total number of nodes in the network. The rationale behind this normalization is that if the network is large compared to the number of nodes interfered with by a flow, focusing on per-flow performance may be better since even if a flow causes congestion at some of its neighbors, the effects are only local and have a small impact on overall network performance.

While the physical meaning of MIC is obvious, we need to check if it is isotonic so that efficient algorithms can be used to find minimum weight paths. The example in Figure 6, however, shows that MIC is not isotonic if used directly. In the example, assuming that link $(A, B, 1)$ has a slightly smaller IRU than link $(A, B, 2)$, the weights of paths $(A, B, 1)$ and $(A, B, 2)$ satisfy: $MIC((A, B, 1)) < MIC((A, B, 2))$. However, due to the reuse of channel 1 on path $(A, B, 1) \oplus (B, C, 1)$ ($\oplus$ means concatenation of two paths), $MIC((A, B, 1) \oplus (B, C, 1)) > MIC((A, B, 2) \oplus (B, C, 1))$. Hence, based on the definition of isotonicity, MIC is not an isotonic path weight function if used directly in real networks. In the next section, we show how the problem of MIC's non-isotonic property can be solved.

V. LIBRA Routing Protocol

Although MIC is not isotonic, our novel routing protocol, named **L**oad and **I**nterference **B**alanced **R**outing **A**lgorithm (LIBRA), can create a virtual network from a real network and decompose MIC into isotonic link weight assignments on this virtual network. By using Bellman-Ford or Dijkstra's algorithm on this virtual network, LIBRA can calculate minimum MIC weight paths of the real network so that flows can be routed though minimum weight paths and no forwarding loops are created for either distance-vector or link-state routing.

The design of LIBRA includes four components.

1) Neighbor Management: each node must discover and measure the ETTs to its neighbors using each of its channels. We assume that a node knows the number of nodes in its interference range as discussed in Section IV-A.
2) Routing Control: Each node propagates/exchanges routing information with other nodes in the network.

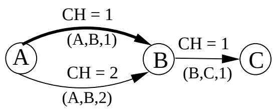


Fig. 6. Non-isotonicity of MIC

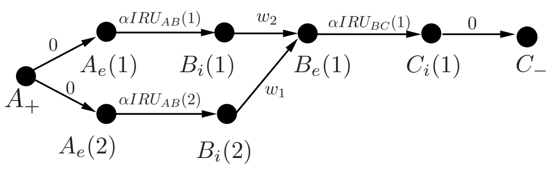


Fig. 7. Isotonic decomposition of MIC into the virtual network of Figure 6.

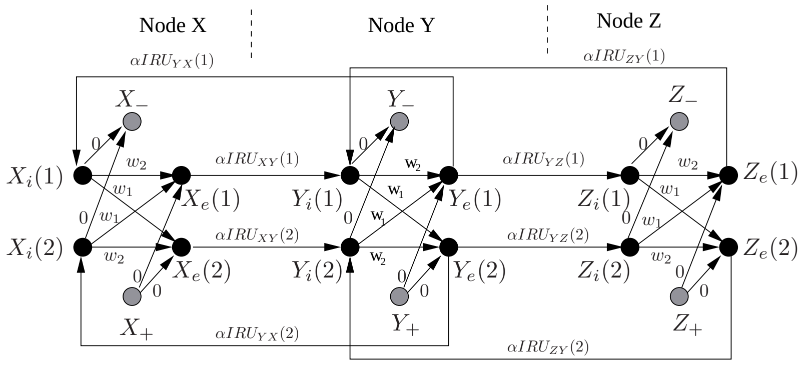


Fig. 8. Virtual nodes

3) Route Determination: Each node calculates minimum weight paths based on the information from all other nodes.
4) Routing Table Management: Each node builds routing tables based on the minimum weight paths.

The design of the Neighbor Management and Routing Control components of LIBRA is similar to those presented in [7], where ETT is measured by periodic hello messages between neighboring nodes so that the IRU cost can be calculated. If link-state routing is used, each node collects topology and IRU information from all other nodes to determine minimum weight paths. If distance-vector routing is used, a node maintains its distances (the minimum weight among currently known paths) to other nodes and exchanges these distances with neighbors to find minimum weight paths to all nodes in the network.

The design of the Route Determination and Routing Table Management components in LIBRA, however, is unique. Briefly speaking, Route Determination is based on the decomposition of MIC into isotonic link weight assignments on a virtual network, which is an image of the original real network. Then, by using distance-vector or link-state routing in this virtual network, LIBRA can find minimum weight paths in terms of MIC between any pair of nodes. Routing Table Management utilizes the constructed minimum weight paths to build routing tables and forwards packets accordingly. In the rest of this section, we focus on the design of these two components.

A. Decomposition of MIC into a Virtual Network

By carefully mapping a real network into a virtual network, we can decompose MIC into isotonic link weight assignments in the virtual network. The goal of this decomposition is to ensure that LIBRA can use efficient algorithms to find minimum weight paths without creating forwarding loops.

The decomposition of MIC is based on a deep understanding of why MIC is not isotonic. Figure 6 demonstrates that the non-isotonic behavior of MIC is due to the fact that the additional weight that link $(B, C, 1)$ brings to a path not only depends on link $(B, C, 1)$'s own status, but is also related to the channel assignment of the link that precedes link $(B, C, 1)$. Due to the common channel used by links $(A, B, 1)$ and $(B, C, 1)$, adding link $(B, C, 1)$ to path $(A, B, 1)$ introduces a higher cost than adding link $(B, C, 1)$ to path $(A, B, 2)$. Hence, even though $MIC((A, B, 1)) < MIC((A, B, 2))$, $MIC((A, B, 1) \oplus (B, C, 1)) > MIC((A, B, 2) \oplus (B, C, 1))$.

To cope with this problem, note that for MIC, whether a cost increment will be different by adding a link is only related to the channel assignment of the previous link on the path. The possible assignments of channels for the precedent link are limited by the number of radios that a node has since each radio is only configured to one channel. Hence, by introducing several virtual nodes to represent these possible channel assignments for the precedent link, we can translate MIC into isotonic weight assignments to the links between these virtual nodes. More specifically, for every channel c that a node X's radios are configured to, two virtual nodes $X_i(c)$ and $X_e(c)$ are introduced. $X_i(c)$ represents that node $prev(X)$ transmits to node X on channel c. $X_e(c)$ indicates that node X transmits to its next hop on channel c. The subscript i stands for "ingress" and the subscript e stands for "egress". Figure 8 shows an example of the virtual nodes for nodes X, Y and Z. (Virtual nodes X_+ and X_- in Figure 8 will be explained in Section V-B.)

Links from the ingress virtual nodes to the egress virtual nodes at node X are added and the weights of these links are assigned to capture different CSC costs as shown in Figure 8. Link $(X_i(c), X_e(c))$ represents that node X does not change channels while forwarding packets and hence weight w_2 is assigned to this link. Similarly, weight w_1 is assigned to link $(X_i(c), X_e(c1))$, where $c \neq c1$, to represent the low cost of changing channels while forwarding packets.

Links between the virtual nodes belonging to different real nodes are used to capture the IRU weight. If node X is able to communicate with its neighbor node Y through channel c, two links $(X_e(c), Y_i(c))$ and $(Y_e(c), X_i(c))$ are added for each common channel between nodes X and Y (e.g., links $(X_e(1), Y_i(1))$ and $(Y_e(1), X_i(1))$ in Figure 8). The weights of these links are $\alpha IRU_{XY}(c)$ and $\alpha IRU_{YX}(c)$, respectively.

By building the virtual network from a real network, we essentially decompose MIC in the real network into weight assignments to the links between virtual nodes. This is because the MIC weight of a real path in a real network can be reconstructed by aggregating all of the weights of the virtual links on the corresponding virtual path. The IRU part of MIC is reflected in the weight of the links between virtual nodes in different real nodes. The CSC costs are captured by routing through different virtual links inside real nodes.

For example, the original real paths in Figure 6 are mapped to the virtual network in Figure 7. The real path $(A,B,1) \oplus (B,C,1)$, which has MIC weight $\alpha IRU_{AB}(1) + \alpha IRU_{BC}(1) + w_2$, is mapped to a virtual path $A_e(1) \rightarrow B_i(1) \rightarrow B_e(1) \rightarrow C_i(1)$, whose aggregated link weight is also $\alpha IRU_{AB}(1) + \alpha IRU_{BC}(1) + w_2$. The real path $(A,B,2) \oplus (B,C,1)$ with MIC weight $\alpha IRU_{AB}(2) + \alpha IRU_{BC}(1) + w_1$ is mapped to path $A_e(2) \rightarrow B_i(2) \rightarrow B_e(1) \rightarrow C_i(1)$, whose aggregated link weight is also $\alpha IRU_{AB}(2) + \alpha IRU_{BC}(1) + w_1$. Similarly, real paths $(A,B,1)$ and $(A,B,2)$ are mapped to $A_e(1) \rightarrow B_i(1)$ and $A_e(2) \rightarrow B_i(2)$, respectively and the MIC weights of the paths are maintained in the mapping.

It is worth noting that WCETT's non-isotonicity is also caused by the dependence of its $\max X_j$ component (Equation (5)) on the channel assignments of multiple links. However, WCETT cannot be decomposed into isotonic metrics by introducing virtual nodes since the value of $\max X_j$ is related to the channel assignments on all links. Essentially, the weight increment of adding a link c to a path p depends on how many times each channel has appeared in path p. As the length of p increases, the combination of channel assignments can become infinite. Hence, it is impossible to introduce virtual nodes to represent all channel assignment states. In addition, WCETT's consideration of the channel assignment on the entire path is not necessary since a node usually does not interfere with other nodes that are more than two hops away even if they share the same channel.

B. Finding Minimum MIC Weight Paths

The purpose of building the virtual network is to ensure that efficient algorithms can find minimum MIC weight paths. This can be achieved since the MIC of a real path is the same as the aggregated link weight of the corresponding virtual path and the weight assignments of virtual links are isotonic due to the fact that the weight of each virtual link is independent of the status of any other virtual link. Therefore, running either Bellman-Ford or Dijkstra's algorithm on the virtual network produces minimum weight paths between any pair of virtual nodes.

However, the purpose of routing is not to find the minimum weight paths between virtual nodes. Instead, the purpose is to find minimum MIC weight paths between real nodes. For example, in Figure 7, the path $A_e(1) \rightarrow B_i(1) \rightarrow B_e(1) \rightarrow C_i(1)$ is the minimum weight path between $A_e(1)$ and $C_i(1)$. In the real network, this means that if node A starts transmitting on channel 1 and wants node C to receive its packets through channel 1 as well, the minimum weight path goes through links $(A,B,1)$ and $(B,C,1)$ and does not switch channels at node B. However, this is not what node A is interested in since node A simply needs to know the minimum weight path between itself and C regardless of what channels are used at the start and end points of the path. To avoid this issue, for every real node X, we introduce two additional virtual nodes, X_- and X_+ (see Figure 8). X_- is used as the virtual destination node for flows destined to node X. Hence, every ingress virtual node of node X has a link with weight 0 pointing to X_-. X_+ is used as the virtual source node for flows starting at node X. Hence, X_+ has a link with weight 0 pointing to each $X_e(c)$. Calculating the minimum MIC weight path between a real node X and any other real node Z simply requires finding the minimum weight virtual path between X_+ and Z_-.

C. Routing Table Architecture

In theory, we have shown that MIC can be decomposed into isotonic path weights in a virtual network so that minimum weight virtual paths can be translated into minimum MIC weight real paths. For implementation purposes, however, it is still unclear how the routing tables can be built based on the minimum weight virtual paths and how packets are routed accordingly. In this section, we discuss the following implementation issues: what information should be stored in routing tables and how packets should be forwarded based on these routing tables. In Section V-D, we discuss how to build the routing tables.

1) Routing Entries: Each entry in a routing table of node X is a tuple $\langle dst, nexthop, channel \rangle$, where dst is the destination address and $nexthop$ is the address of the next relaying node to the destination. The $channel$ entry represents the channel that node X should use to send packets to $nexthop$.

2) Routing Tables: To ensure that packets follow minimum weight paths, the forwarding decisions at node X must depend on the channel assignment of the previous hop. Assuming that node X has m radios configured to m different channels, each of the m radios should have its own routing table. For example, routing table $T(c)$ is used to represent the routing choices for packets arriving from channel c.

In addition, since in mesh networks every node can initiate traffic, node X must also know how to route traffic initiated by itself. Since node X is the source node, there is no channel assignment from the previous hop to affect the path weight. Hence, the routing table for the minimum weight path starting from node X is different from the routing tables for relaying other nodes' traffic. Therefore, we introduce another routing table for forwarding node X's own packets, called the central routing table (T_+). In total, a node has $m+1$ routing tables.

Although the number of routing tables increases in LIBRA, we believe that such an increasing is acceptable. This is because each routing table is not very large due to the relatively small size of mesh networks. In addition, since the number of radios in a node is usually very small, the number of routing tables that a node needs to store is very limited. Therefore, the memory used by routing tables should not create a problem for mesh nodes. Furthermore, this increase in the number of routing tables does not hurt packet forwarding performance since node X does not need to search all routing tables to make a forwarding decision. When a packet arriving from a channel c needs to be forwarded, only one routing table $T(c)$ is searched to forward the packet.

D. Building Routing Tables

The major benefit of LIBRA compared to WCETT is that it can be implemented with either Bellman-Ford or Dijkstra's algorithms. Hence, using either link-state routing or distance-vector routing, LIBRA is guaranteed to find the minimum MIC weight paths. In addition, because of the isotonic decomposition of the MIC metric, LIBRA does not create any forwarding loops if it uses link-state routing.

1) Using Link-state Routing: Similar to traditional link-state routing, each node propagates its connectivity information with its neighbors to the whole network. The connectivity information includes which channel the node can use to communicate with its neighbors and the IRUs of these links. By gathering connectivity information from other nodes, each node obtains global knowledge of the network topology, including the IRU and channel assignments corresponding to each link in the network. Hence, a node is able to construct the virtual network according to the description in Sections V-A and V-B and use this virtual network to build its routing tables.

To build the central routing table T_+ for node X's own traffic, node X runs Dijkstra's algorithm at virtual node X_+. The resulting minimum weight virtual path between X_+ and any other node Z's virtual node Z_- essentially represents the minimum weight real path from X to Z and can be used to fill T_+ at Node X as follows. The minimum weight virtual path has the form $X_+ \rightarrow X_e(c) \rightarrow Y_i(c) \rightarrow \cdots \rightarrow Z_-$, where Y can be any node. Hence, by observing the third virtual nodes on the virtual path, the routing table entry can be constructed as $\langle Z, Y, c\rangle$.

For example, in Figure 7, running Dijkstra's algorithm at A_+ returns $A_+ \rightarrow A_e(2) \rightarrow B_i(2) \rightarrow B_e(1) \rightarrow C_i(1) \rightarrow C_-$. Hence, node A's central routing table T_+ has a route entry $\langle C, B, 2\rangle$, which shows the traffic initiated by node A to destination C should be forwarded to node B using channel 2.

To build routing tables for relaying traffic at node X, node X runs Dijkstra's algorithm at each of its ingress virtual nodes, $X_i(c)$, to build routing table $T(c)$ for packets received from channel c. The minimum weight path should be $X_i(c) \rightarrow X_e(c_0) \rightarrow Y_i(c_0) \rightarrow \cdots \rightarrow Z_-$, where c_0 may or may not be the same channel as c. By checking the third virtual nodes on the virtual path, the routing entry can be built as $\langle Z, Y, c_0\rangle$.

For example, for node B in Figure 7, running Dijkstra's algorithm at $B_i(2)$ shows that the minimum weight path to node C_- is $B_i(2) \rightarrow B_e(1) \rightarrow C_i(1) \rightarrow C_-$. Hence, node B has entry $\langle C, C, 1\rangle$ in its $T(2)$ table. When a packet of node A arrives from channel 2, node B searches routing table $T(2)$ and finds the entry $\langle C, C, 1\rangle$. Hence, node B forwards the packet to node C through channel 1.

2) Using Distance-Vector Routing: In traditional distance-vector routing, a node X maintains and informs its neighbors about its distances to every node in the network. In the distance-vector version of LIBRA, node X maintains and informs its neighbors about the distances of its ingress virtual nodes (e.g., $X_i(c)$ for some channel c) to every node's destination virtual node (e.g., virtual node Z_- at node Z). Using the example in Figure 7, node C tells node B that virtual node $C_i(1)$ has distance 0 to virtual node C_-. Upon receiving this information, node B knows that $B_e(1)$ can reach C_- through $C_i(1)$ with distance $\alpha IRU_{BC}(1)$ plus the distance between $C_i(1)$ and C_-. Therefore, node B knows that $B_i(1)$ can reach C_- first through $B_e(1)$ and then through $C_i(1)$. Hence, $B_i(1)$'s distance to C_- is $w_2 + \alpha IRU_{BC}(1)$ and node B can add entry $\langle C, C, 1\rangle$ to its routing table $T(1)$. Similarly, the virtual node $B_i(2)$'s distance to C_- can be calculated as $w_1 + \alpha IRU_{BC}(1)$ with the corresponding entry $\langle C, C, 1\rangle$ for $T(2)$. Finally, node B propagates $B_i(1)$ and $B_i(2)$'s distances to C_- to node A. Node A updates node $A_e(1)$'s distance to C_- as $\alpha IRU_{AB}(1) + w_2 + \alpha IRU_{BC}(1)$ and node $A_e(2)$'s distance to C_- as $\alpha IRU_{AB}(2) + w_1 + \alpha IRU_{BC}(1)$. Since node A_+'s distance to C_- through $A_e(2)$ is shorter than through $A_e(1)$, node A chooses $A_e(2)$ and $B_i(2)$ as node A_+'s path to node C_-. Hence, an entry $\langle C, B, 2\rangle$ is added to the central routing table T_+ of node A.

VI. Extension to Multihop intra-flow Interference

In the design of the CSC metric for capturing intra-flow interference, we only consider the interference between two consecutive nodes on a path. However, depending on the carrier-sensing range, intra-flow interference may also exist between nodes that are further away. In this case, considering the channel assignments at more hops before forwarding a packet may further reduce intra-flow interference and improve network performance. To do so, extensions to both the definition of CSC and the design of LIBRA are needed. These extensions introduce costs in terms of increasing computation complexity and memory consumption at nodes.

If intra-flow interference exists between nodes that are two hops away, node X interferes with both nodes $prev(X)$ and $prev^2(X)$, where $prev^2(X)$ stands for the node that is the two hop precedent of node X in a path. To capture the two-hop interference on node X, Equation (7) is extended to:

$$CSC_X = \begin{cases} w_2, & \text{if } CH(prev^2(X)) \neq CH(X) = CH(prev(X)), \\ w_3, & \text{if } CH(prev^2(X)) = CH(X) \neq CH(prev(X)), \\ w_2 + w_3, & \text{if } CH(prev^2(X)) = CH(X) = CH(prev(X)), \\ w_1, & \text{otherwise,} \end{cases} \tag{11}$$

$$0 \leq w_1 \ll w_3 < w_2, \tag{12}$$

where w_3 captures the intra-flow interference between nodes $prev^2(X)$ and X and w_2 captures the intra-flow interference between nodes $prev(X)$ and X. $w_3 < w_2$ since the further away that two nodes are, the less interference exists between them. In our simulation, we set $w_3 = 0.6 w_2$ to capture this relationship.

To incorporate this new CSC definition, LIBRA's construction of the virtual networks and the actual routing procedures needs to be extended. To build virtual networks to reflect the new CSC definition, different virtual nodes are needed to represent the channel assignments at both $prev^2(X)$ and $prev(X)$. Consider a node X with m radios, each configured to a different channel. For each channel c at node X, $m+1$ egress virtual nodes $X_e(c^a, c)$ are introduced. $X_e(c^a, c)$ represents that a flow arrives at node X from channel c^a and is relayed by node X through channel c. c^a has two types of values: it is either the same as one of the m channels configured at node X or -1 to represent that the flow is

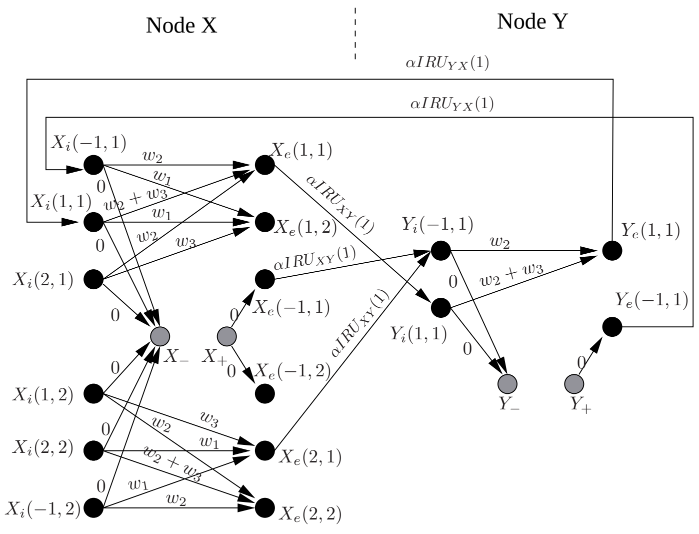


Fig. 9. Virtual Nodes for multihop intra-flow Interference

initiated by node X itself so that there is no $prev^2(X)$ on the path. Another $m+1$ ingress virtual nodes $X_i(c^p, c)$ are also introduced for each channel c at node X. $X_i(c^p, c)$ represents that a flow travels from $prev^2(X)$ to $prev(X)$ through channel c^p and arrives at node X through channel c. c^p has two types of values: it is either the same as one of the m channels configured at node X or -1 to represent that $prev^2(X)$ uses a channel that is different from all channels configured at node X. Figure 9 shows an example of the virtual nodes at two nodes X and Y. Node X has two radios configured to channels 1 and 2, respectively. Hence, Node X has six ingress virtual nodes and six egress virtual nodes. Node Y has one radio configured to channel 1. Therefore, Node Y has only two ingress virtual nodes and two egress virtual nodes.

Based on the new CSC definition in Equation (12), the virtual links between virtual nodes are added as follows. For every pair of virtual nodes $X_i(c_1, c_2)$ to $X_e(c_2, c_3)$ in node X, a link from $X_i(c_1, c_2)$ to $X_e(c_2, c_3)$ is added. If $c_1 \neq c_3 = c_2$, the weight of the link is w_2. If $c_1 = c_3 \neq c_2$, the weight of the link is w_3. If $c_1 = c_2 = c_3$, the weight of the link is $w_2 + w_3$. Otherwise, the weight of the link is w_1. X_+ has a link with 0 weight to every $X_e(-1, c)$ and every $X_i(c_1, c_2)$ has a link with 0 weight to X_-. For every pair of virtual nodes $X_e(c_1, c_2)$ and $Y_i(c_1, c_2)$ belonging to two neighboring nodes X and Y, a link from $X_e(c_1, c_2)$ to $Y_i(c_1, c_2)$ with weight $\alpha IRU_{XY}(c_2)$ is added to capture the IRU weight. In addition, for virtual node $X_e(c_1, c_2)$, if c_1 is not one of the channels configured at node Y, a link from $X_e(c_1, c_2)$ to $Y_i(-1, c_2)$ with weight $\alpha IRU_{XY}(c_2)$ is also added. With all of the links between virtual nodes, we can then build a routing table $T(c_1, c_2)$ for each of the ingress nodes $X_i(c_1, c_2)$ similar to the procedure in Section V-D.

Since the decision about packet forwarding now is based on two-hop channel assignments, the actual routing procedure of LIBRA also changes. A packet now needs to have an extra field $prev\_chan$ in its packet header to carry the channel number that $prev^2(X)$ used to forward it. When a node X forwards a packet, it stamps $prev\_chan$ with the channel from which this packet arrived at node X. Therefore, when the next hop node $Next(X)$ receives this packet from node X through channel c, $Next(X)$ knows that the packet traveled first through channel $prev\_chan$ and then channel c before reaching $Next(X)$. Therefore, $Next(X)$ can forward the packet using the corresponding routing table. More specifically, if $prev\_chan$ equals one of $Next(X)$'s channels, table $T(prev\_chan, c)$ is used. Otherwise, table $T(-1, c)$ is used.

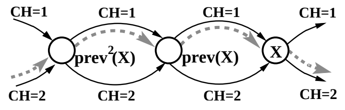


Fig. 10. Routing when radios dynamically reconfigure their channel assignment. Each node only has one radio.

Although the above extensions are designed for two-hop intra-flow interference, similar approaches can be used to capture intra-flow interference between nodes that are further away. However, the cost for considering intra-flow interference between multiple hops is the increased number of routing tables and calculation complexity for computing routing tables. To consider two-hop intra-flow interference, the number of routing tables for a node with m radios is $(m+1)m+1$, which is much larger than the $m+1$ routing tables needed for considering 1-hop intra-flow interference. Therefore, there is a trade-off between the cost and benefit of considering multi-hop intra-flow interference. In our simulations, we show that even though the interference range is two-hop, the improvement of considering multihop interference in LIBRA is not significant. Given the computation and memory overhead associated with capturing multihop intra-flow interference, considering multi-hop interference in LIBRA may not be necessary.

VII. Dynamic Channel Configuration

In all our discussions so far, we assume that each wireless radio is configured to a fixed channel. Therefore, for a node to switch transmission channel, the node must transmit on different wireless radios. This assumption is mainly due to the fact that reconfiguring channel assignment in current wireless cards are very slow and it is not practical to let a wireless card dynamically reconfigure its channels while forwarding packets. However, in some recent studies [3], it has been suggested that future wireless cards may be able to reconfigure their channels very fast. In such case, instead of being fixed to a single channel, a wireless card can dynamically switching between multiple channels while forwarding packets, which potentially can further boost up the network capacity [3].

However, intra-flow interference still exists even if a wireless card receives and transmits packets on different channels through dynamic channel reconfiguration. This is because a single wireless card cannot transmit/receive on multiple channels simultaneously. Figure 10 shows an example where every node only has one radio and each radio can dynamically configure to two channels. In this example, if node $prev(X)$ uses a channel that is different from $prev^2(X)$ to transmit to node X, node $prev(X)$ needs to dynamically reconfigure the channel assignment of its only radio to enable it to transmit and receive from different channels. This dynamic channel reconfiguration does not reduce the intra-flow interference between nodes $prev^2(X)$ and $prev(X)$ since node $prev(X)$ cannot receive and transmit simultaneous on two different channels of the same radio. Considering the costs associated with reconfiguring the radio to switch channels, such as delay and computational power, channel reconfiguration at the radio of $prev(X)$ may even be harmful. However, if through channel reconfiguration, nodes $prev^2(X)$ and X use different channels, the interference between these two nodes can be greatly reduced. This indicates that channel reconfiguration should only be performed for every other hop instead of every hop as depicted by the dotted arrows in Figure 10.

Based on the above analysis, for radios with the ability of fast channel reconfiguration, the CSC of any node X on a path p should be based on the channels used by its previous two hop radios. In such case, denoting $RD_r(X)$ as the radio that node X uses for receiving and $RD_t(X)$ as the radio that node X uses for transmitting, the definition of CSC in Equation (12) should be extended to:

$$CSC_X = \begin{cases} w_2, & \text{if } CH(prev^2(X)) \neq CH(X) = CH(prev(X)), \\ w_3, & \text{if } CH(prev^2(X)) = CH(X) \neq CH(prev(X)) \\ & \text{and } RD_r(X) \neq RD_t(X), \\ w_3 + w_4, & \text{if } CH(prev^2(X)) = CH(X) \neq CH(prev(X)) \\ & \text{and } RD_r(X) = RD_t(X), \\ w_2 + w_3, & \text{if } CH(prev^2(X)) = CH(X) = CH(prev(X)), \\ w_1, & \text{if } CH(prev^2(X)) \neq CH(X) \neq CH(prev(X)) \\ & \text{if } RD_r(X) \neq RD_t(X) \\ w_1 + w_4, & \text{if } CH(prev^2(X)) \neq CH(X) \neq CH(prev(X)) \\ & \text{if } RD_r(X) = RD_t(X) \end{cases} \quad (13)$$

$$0 \leq w_1 \ll w_3 < w_2, \quad (14)$$

$$0 < w_4, \quad (15)$$

where w_4 captures the cost, such as delay and computation overhead, associated with channel reconfiguration.

Similar to Section VI, based on the above definition of CSC, we can construct virtual nodes and virtual links between these virtual nodes to represent the CSC metric. Figure 11 shows the virtual nodes for a node X that has two radios: radio 1 and radio 2. Radio 1 is able to dynamically reconfigure to both channels 1 and 2. Radio 2 is fixed on channel 1. In this figure, the superscript of each virtual node represents the radio number used by node X. For example, virtual node $X_i^r(c^p, c)$ represents that a flow travels from $prev^2(X)$ to $prev(X)$ through channel c^p and arrives at node X through channel c of radio r. $X_e^r(c^a, c)$ represents that a flow arrives at node X from channel c^a and is relayed by node X through channel c of radio r. For every pair of virtual nodes $X_i^{r_1}(c_1, c_2)$ to $X_e^{r_2}(c_2, c_3)$ in node X, a link from $X_i^{r_1}(c_1, c_2)$ to $X_e^{r_2}(c_2, c_3)$ is added. If $c_1 \neq c_3 = c_2$, the weight of the link is w_2. If $c_1 = c_3 \neq c_2$ and $r_1 \neq r_2$, the weight of the link is w_3. If $c_1 = c_3 \neq c_2$ and $r_1 = r_2$, the weight of the link is $w_3 + w_4$. If $c_1 = c_2 = c_3$, the weight of the link is $w_2 + w_3$. If $c_1 \neq c_3 \neq c_2$ and $r_1 = r_2$, the weight of the link is $w_1 + w_4$. If $c_1 \neq c_3 \neq c_2$ and $r_1 \neq r_2$, the weight of the link is w_1. X_+ has a link with 0 weight to every $X_e^r(-1, c)$ and every $X_i^r(c_1, c_2)$ has a link

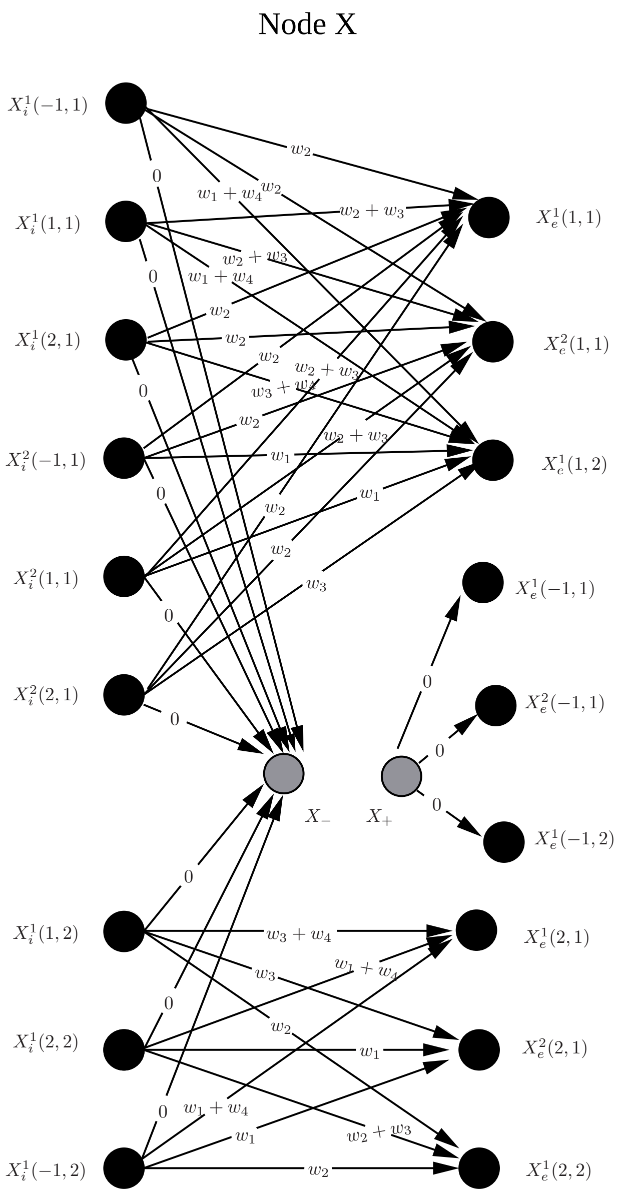


Fig. 11. Virtual nodes for dynamic channel configuration at node X. Radio 1 of node X can dynamically reconfigure to channels 1 and 2. Radio 2 is fixed at channel 1.

with 0 weight to X_-.

Based on the new virtual nodes and virtual links, we can extend LIBRA using a similar approach discussed in Section VI to support dynamic channel reconfiguration.

VIII. Evaluation

To demonstrate the effectiveness of our new path weight function MIC and the performance of LIBRA, our evaluation includes four parts. First, to demonstrate the effectiveness of the IRU part of MIC in balancing network load and reducing inter-flow interference, we compare LIBRA with ETT and optimal routing produced by solving the linear program in Equation (2) in single-radio/single-channel networks and omit the CSC part of MIC. Since in single-radio/single-channel networks every node has only one radio and every radio in the network is configured to the same channel, there is no channel switching to reduce intra-flow interference. Hence, WCETT, which is aimed to reduce intra-flow interference, is not used in this part of our evaluation. In the second part of our evaluation, we evaluate the performance of LIBRA with MIC metrics in multi-channel/multi-radio networks to further measure LIBRA's ability to reduce both inter-flow and intra-flow interference. We demonstrate the effectiveness of LIBRA by comparing LIBRA's performance with hop count, ETT, optimal routing and WCETT. WCETT is used in the second part of our evaluation since with multi-channel/multi-radio nodes, WCETT can use channel switching to reduce intra-flow interference. In the third part of our evaluation, we examine the sensibility of LIBRA's performance to the choice of w_2 and w_3 in CSC definition. In the final part of our evaluation, we study the performance of LIBRA's extension on networks when radios are able to dynamic reconfigure their channel assignments.

All of our simulations are performed in the NS2 simulator [25]. The topologies of simulations are randomly generated. Although our LIBRA protocol supports communication between any pair of nodes, since we expect that most of the traffic in a real mesh network will be traffic to/from the wired network, in our simulations, all flows are destined to the Internet through one to four TAPs. The sources of the flows are randomly located in the mesh network. All flows are CBR flows with 512 Byte packets. Distance vector routing is used to simulate ETT, WCETT and hop count performance. The evaluation of all protocols is based on the performance of the system after the routing tables are stabilized. The transmission range is 250m while the carrier-sensing range is 550m. The transmission rates between neighboring nodes are related to the distance between the nodes as shown in Table I.

A. Single Channel/Single Radio Environments

In the first set of simulations, we randomly generated six $1500m \times 1500m$ networks with 160 nodes, 15 flows and 4 TAPs. To evaluate the effectiveness of using IRU to perform load balanced routing, every node in the network logs the fraction of channel busy time at its location, which is the indication for channel utilization in Equation (1). Figure 12 depicts how the aggregated cost of channel utilization of the whole network (Φ in Equation (2)) increases as the per-flow rate increases. The optimal paths obtained by solving the linear optimization problem in Equation (2) have the lowest aggregated cost, which is expected since the objective function of the optimization problem is to minimize the total cost of channel utilization. However, as discussed in Section II, solving the optimization problem to find optimal paths is not practical to implement. Among the three practical solutions, the channel utilization cost of IRU is the lowest due to the IRU metric's excellent ability to balance network load and reduce inter-flow interference.

To understand the benefits of balancing network load, Figures 13(a), 13(b) and 13(c) show the maximum channel utilization among nodes, the total network throughput and the average end-to-end packet delay, respectively. Again, optimal routing has the best performance since it has the lowest maximum channel utilization, the highest total network throughput and the smallest average end-to-end packet delay. The performance of LIBRA with IRU, however, is not far from optimal routing and is much better than the performance of ETT and hop count due to its ability of balance the load in the network.

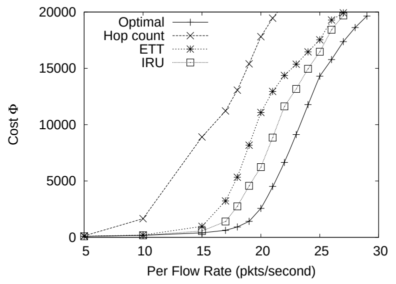


Fig. 12. Cost (Φ) in $1500m \times 1500m$ 160 node 1-radio/1-channel networks

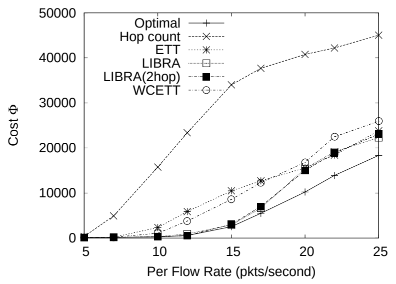


Fig. 14. Cost (Φ) in $1000m \times 1000m$ 100 node 2-radio/3-channel networks

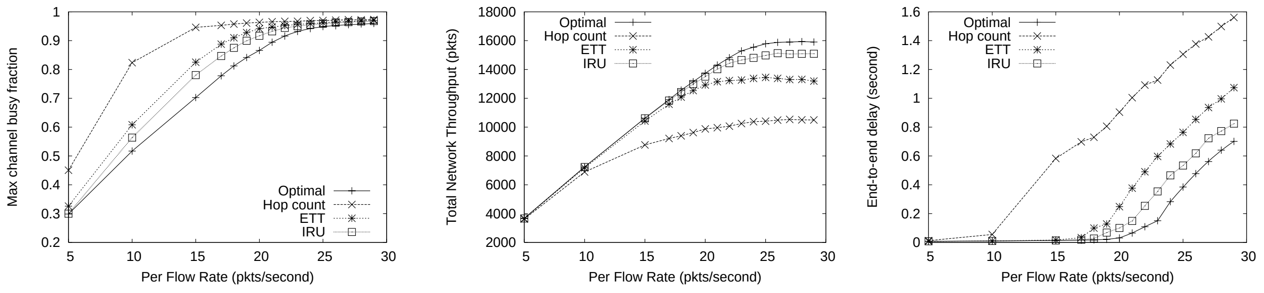
B. Multi-Channel/Multi-Radio Environment

To examine LIBRA's performance for multi-channel/multi-radio networks, in the second set of simulations, every node has two radios and each radio can be configured to one of three channels. The full MIC metric, which includes both IRU and CSC, is used, with $w_2 = 0.5$ and $w_3 = 0.3$ (See Equations (7) and (12)). The performance of LIBRA under other settings of w_2 and w_3 is presented in Section VIII-C. We randomly generate ten $1000m \times 1000m$ networks, each with 100 nodes, 20 flows and 1 TAP. Compared to the first set of simulations, the capacity of the network is increased by having multiple channels. Therefore, we use a higher node density, a larger number of flows and a smaller number of TAPs to increase network load.

Figure 14 depicts the aggregated cost of channel utilization (Φ in Equation (2)), where LIBRA (2hop) represents the extended version of LIBRA for considering two hop intra-flow interference. Again, the optimal paths obtained by solving the linear optimization problem in Equation (2) are shown to have the lowest aggregated cost. Among all of the other practical solutions, the channel utilization cost of LIBRA and LIBRA (2hop) is the lowest since LIBRA balances network load and reduces both intra-flow and inter-flow interference. The difference between LIBRA and LIBRA (2hop) is very small and hard to distinguish. Figures 15(a), 15(b) and 15(c) show the maximum channel utilization among nodes, the total network throughput and the average end-to-end packet delay, which further confirm that optimal routing has the best

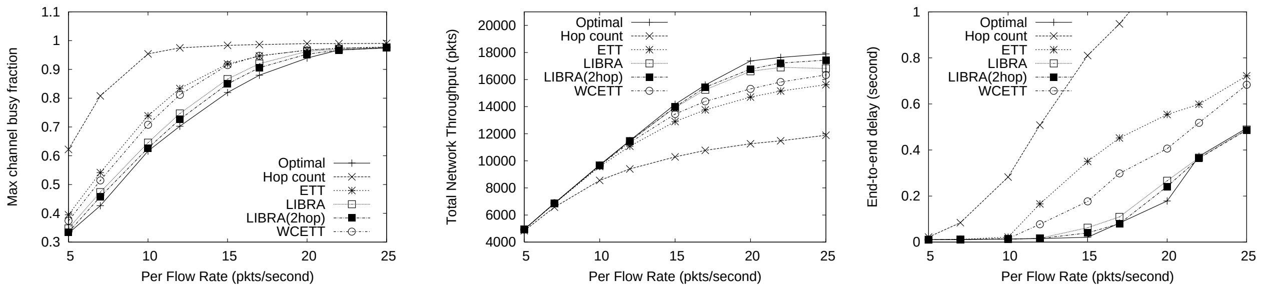
TABLE I
DISTANCE/RATE RELATIONSHIPS

Distance(m)	25	50	75	100	125	150	175	200	225	250	>250
Rate(Mbps)	54	48	36	24	18	12	9	6	2	1	0



(a) Maximum channel utilization (b) Total network throughput (c) Average end-to-end packet delay

Fig. 13. $1500m \times 1500m$ 160 node single channel/single radio networks



(a) Maximum channel utilization (b) Total network throughput (c) Average end-to-end packet delay

Fig. 15. $1000m \times 1000m$ 100 node 2-radio/ 3-channel networks

performance. The performance of LIBRA and LIBRA (2hop), however, is not far from optimal routing and are much better than the performance of ETT, WCETT and hop count due to the balancing of traffic load. LIBRA (2hop) has a slightly better performance than LIBRA due to its consideration of two-hop intra-flow interference.

C. Sensitivity to w_2 and w_3 weight in CSC

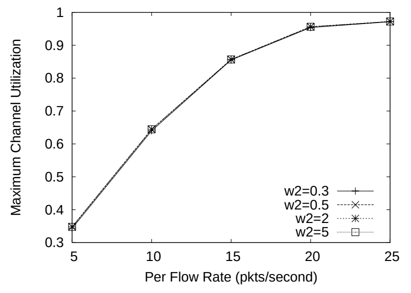
To understand whether the performance of LIBRA is sensitive to the choice of w_2 and w_3 weight in the definition of CSC, in this set of simulation, we test different w_2 and w_3 configurations on ten randomly generated $1000m \times 1000m$ networks, each with 100 nodes, 20 flows and 1 TAP. The range of w_2 changes from 0.3 to 5 and the range of w_3 changes from 0.18 to 3. Figures 16 and 17 show how the performance of LIBRA and LIBRA(2hop), including the maximum channel utilization, total network throughput and average end-to-end packet delay, changes as the values of w_2 and w_3 vary. The variations of w_2 values has almost no impact on the performance of LIBRA. However, LIBRA(2hop) is more sensitive to the changes in w_2 and w_3 and when w_2 and w_3 becomes too large (e.g., $w_2 = 5$, $w_3 = 3$), LIBRA(2hop)'s performance can be even worse than LIBRA. Considering the fact that LIBRA(2hop) is more expensive and its improvement over LIBRA is not significant even if w_2 and w_3 is chosen appropriately, we conclude that LIBRA is more preferably in real networks.

D. Dynamic Channel Reconfiguration

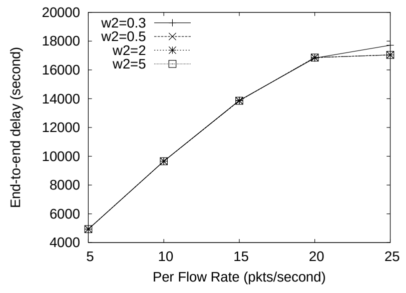
To evaluate LIBRA's performance for networks where radios can dynamically reconfigure their channels, we made a simple extension to the IEEE 802.11 implementation in NS2. In this extension, a common control channel is used by all nodes for RTS/CTS handshakes. The channel number that should be used for DATA/ACK packets is carried in the RTS/CTS handshake for channel reservation. After finishing the RTS/CTS handshake in the common channel, the sender and receiver nodes switch to the data channel and perform the DATA/ACK handshake. [1] Since WCETT does not support dynamic radio reconfiguration, WCETT is not included in the simulations in this section.

In the first set of simulations, we examine LIBRA's performance for networks where each node only has one radio and each radio is able to dynamically switching between three channels, one control channel and two data channel. The simulations are performed in six $1000m \times 1000m$ networks with 100 nodes, 17 flows and 2 TAPs. The radio used in the simulations is configured as Radio 1 in Table II. Figures 18(a) and 18(b) show the total network throughput and average end-to-end packet delay. For LIBRA, both the total network throughput and the average packet delay are very close to

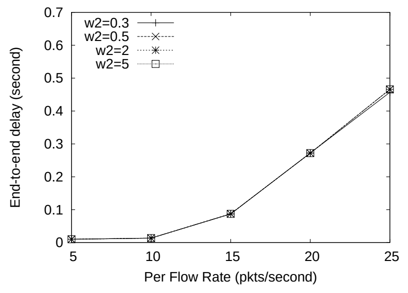
[1]We use this scheme for its simplicity of implementation and to test the performance of LIBRA. For the design of real multi-channel networks, more efficient schemes [26], [27], [28] may be used.



(a) Maximum channel utilization

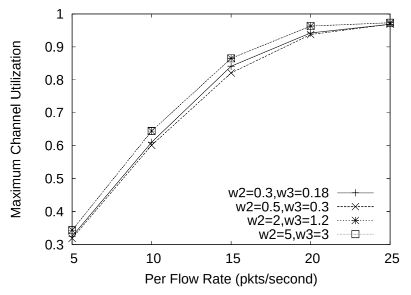


(b) Total network throughput

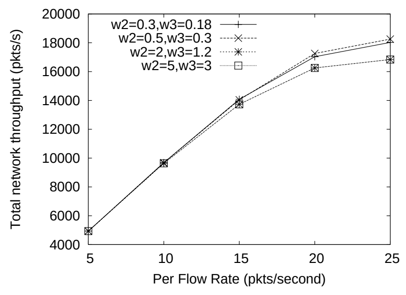


(c) Average end-to-end delay

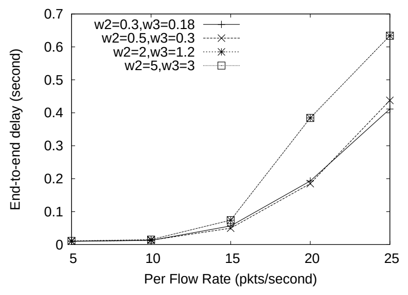
Fig. 16. Sensitivity of LIBRA to weight w_2



(a) Maximum channel utilization



(b) Total network throughput



(c) Average end-to-end delay

Fig. 17. Sensitivity of LIBRA(2hop) to weight w_2 and w_3

TABLE II

DISTANCE/RATE RELATIONSHIPS FOR SIMULATIONS OF MULTI-CHANNEL/MULTI-RADIO NETWORKS

Distance (m)	60	120	180	250	>250				
Radio 1 Rate (Mbps)	11	5.5	2	1	0				
Distance(m)	10	20	30	40	50	60	70	80	>80m
Radio 2 Rate(Mbps)	54	48	36	24	18	12	9	6	0

optimal routing and are much better than the performance of ETT and hop count, demonstrating LIBRA's excellent ability to exploit multi-channel capabilities of nodes.

In the second set of simulations, we examine LIBRA's performance in networks where 70% of the nodes are equipped with two radios: radio 1 and radio 2 (Table II), while the rest of the nodes are equipped with only radio 1. Radio 1 can switch between channel 1,2 and 3. Radio 2 can switch between channel 4,5 and 6. The simulations are performed in six $1000m \times 1000m$ networks with 100 nodes, 17 flows and 2 TAPs. Figures 19(a) and 19(b) show the total network throughput and average end-to-end packet delay. Again, LIBRA's performance is close to optimal routing and is much better than both ETT and hop count.

IX. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have presented a comprehensive study of load balancing in mesh networks. We formulate the goal of load balancing in mesh networks and show how it can be optimally achieved in theory. We then propose a practical heuristic solution to approach the optimal theoretical result. Our solution includes a new path weight function, MIC, and a routing protocol, LIBRA. The novelties of MIC are that it can balance the overall network load and at the same time exploit the multi-channel/multi-radio abilities of nodes to improve per-flow performance. The combination of MIC and LIBRA is efficient since it is compatible with both Bellman-Ford and Dijkstra's algorithms, which are the most efficient algorithms for calculating minimum weight paths. By comparing the performance of our solution with existing solutions using simulations, we find that our heuristic solution can approach the theoretical optimal routing's performance.

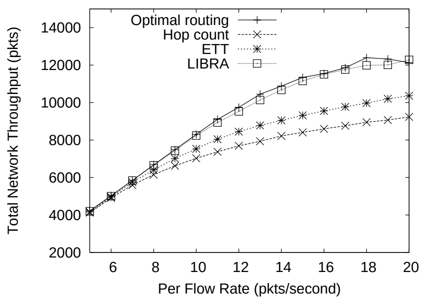
Our future work for LIBRA and MIC is to investigate the trade-off of setting the w's in Equation (7) and α in Equation (9). We will investigate what are the appropriate w's for real mesh networks based on actual hardware measurements. We also want to further study how α affects the delay and throughput of flows and the overall load on the network. Finally, we will investigate how to integrate LIBRA with mesh networks that have both mobile and static nodes.

REFERENCES

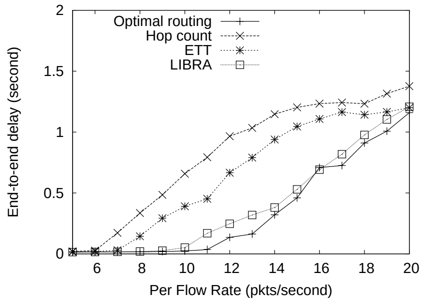
[1] Roger Karrer, Ashutosh Sabharwal, and Edward Knightly, "Enabling Large-scale Wireless Broadband: The Case for TAPs," in *Proceedings of HotNets*, Cambridge, MA, 2003.

[2] Violeta Gambiroza, Bahareh Sadeghi, and Edward Knightly, "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks," in *ACM Mobicom*, 2004.

[3] Pradeep Kyasanur and Nitin Vaidya, "Multi-Channel Wireless Networks: Capacity and Protocols," Tech. Rep., University of Illinois at Urbana-Champaign, 2005.

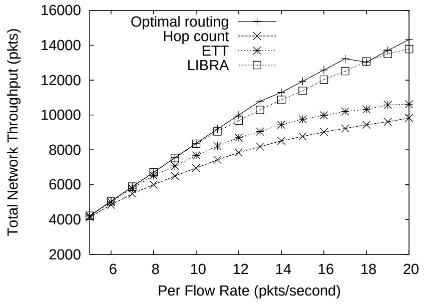


(a) Total network throughput

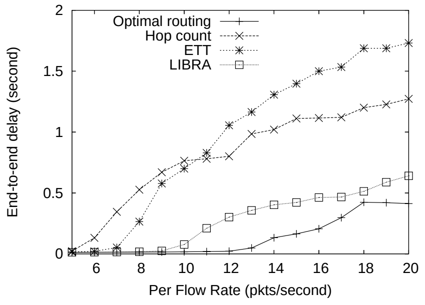


(b) Average end-to-end packet delay

Fig. 18. $1000m \times 1000m$, 100 nodes, 1 radio per node



(a) Total network throughput



(b) Average end-to-end packet delay

Fig. 19. $1000m \times 1000m$, 100 nodes. 70% nodes have 2 radios, 30% nodes have 1 radios.

[4] Hossam Hassanein and Audrey Zhou, "Routing with Load Balancing in Wireless Ad hoc Networks," in *ACM MSWiM*, 2001.

[5] Sung-Ju Lee and Mario Gerla, "Dynamic Load-Aware Routing in Ad hoc Networks," in *IEEE ICC*, 2001.

[6] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *ACM Mobicom*, 2003.

[7] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM Mobicom*, 2004.

[8] Ashish Raniwala and Tzi cker Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *IEEE INFOCOM*, 2005.

[9] Jian Tang, Guoliang Xue, and Weiyi Zhang, "Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks," in *ACM Mobihoc*, 2005.

[10] Yaling Yang and Robin Kravets, "Contention-Aware Admission Control for Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 4, pp. 363–377, 2005.

[11] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance," in *ACM MobiCom*, 2003.

[12] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *IEEE INFOCOM*, Tel-Aviv, Israel, 2000.

[13] A. Sridharan, R. Guerin, and C. Diot, "Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks," in *IEEE INFOCOM*, 2003.

[14] Jun Wang, Yaling Yang, Li Xiao, and Klara Nahrstedt, "Edge-based Traffic Engineering for OSPF Networks," *Elsevier Journal on Computer Networks*, vol. 48, no. 4, pp. 605–625, 2005.

[15] Atul Khanna and John Zinky, "The Revised ARPANET Routing Metric," in *ACM SIGCOMM*, 1989.

[16] E. Anderson and T. Anderson, "On the Stability of Adaptive Routing in the Presence of Congestion Control," in *IEEE INFOCOM*, 2003.

[17] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Inernet," in *IEEE INFOCOM*, 2001.

[18] J. L. Sobrinho, "Network Routing with Path Vector Protocols: Theory and Applications," in *ACM SIGCOMM*, 2003, pp. 49–60.

[19] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *ACM SIGCOMM*, 2004.

[20] David B Johnson and David A Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*. 1996, vol. 353, Kluwer Academic Publishers.

[21] Charles Perkins, "Ad-hoc on-demand distance vector routing," in *MILCOM panel on Ad Hoc Networks*, 1997.

[22] Charles Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM*, 1994.

[23] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Schemes for Ad-hoc Wireless Networks," in *IEEE ICC*, 1998.

[24] S. Agarwal, J. Padhye, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill, "Estimation of Link Interference in Static Multihop Wireless Networks," in *ACM Internet Measurement Conference (IMC)*, 2005.

[25] Kevin Fall and Kannan Varadhan, "NS notes and documentation," in *The VINT Project, UC Berkely, LBL, USC/ISI, and Xerox PARC*, 1997.

[26] Paramvir Bahl, Ranveer Chandra, and John Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Netowkrs," in *ACM Mobicom*, 2004.

[27] Jungmin So and Nitin Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver," in *ACM Mobihoc*, 2004.

[28] R. Rozovsky and P. R. Kumar, "SEEDEX: A MAC protocol for ad hoc network," in *ACM Mobihoc*, 2001.