# Interference-aware Loop-free Routing For Mesh Networks

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

## I. Introduction

In recent years, there has been an explosive growth in the development of mesh networks to provide cheap yet high-bandwidth Internet access to areas that does not have coverage of high-bandwidth wired networks [7], [6]. Such mesh networks are composed of static wireless nodes. 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.

To achieve good network performance, a critical design issue for mesh networks is to balance traffic load among nodes. This is because nodes in mesh networks are static. A poor choice of paths may remain unchanged for a long time and create hot spots. Hence, designing path weight functions (also called routing metrics) to facilitate load-balanced routing is very important in mesh networks and is the focus of our work.

To achieve load-balanced routing in mesh networks, path weight functions must reflect the shared nature of wireless channels and support easy calculation of loop-free paths. Existing path weight functions designed for mesh networks, such as hop count, ETX [2] and WCETT [3], do not satisfy these requirements. Hence, their performance is limited and some may cause routing loops. Therefore, in this paper, we present our theoretical studies of these requirements and design a new path weight function, called *Metric of Interference and Channel-switching (MIC)*, that satisfies these requirements. Our preliminary simulation results show that MIC's performance is substantially better than existing approaches. More details of our work can be found in our technical report [11].

The remainder of the paper is organized as follows. In Section II, we present the basic requirements for designing path weight functions for mesh networks and show the limitations of several existing solutions to satisfy these requirements. Sections III and IV introduce our new routing metric MIC and demonstrate how MIC can be used to balance network load. Section V discusses how to extend MIC to capture multi-hop intra-flow interference. Section VI evaluates the performance of MIC. In Section VII, we conclude our work and discuss future directions.

## II. Designing Path Weight Functions

To ensure that routing protocols based on path weight functions have good performance, path weight functions must capture the characteristics of interference in mesh networks, enable efficient algorithms to calculate minimum weight paths, and not create any forwarding loop. In this section, we introduce the theories regarding these requirements and discuss how existing path weight functions fail to satisfy these requirements.

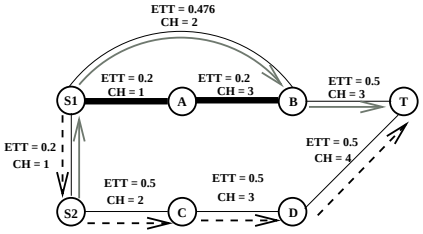



Fig. 1. Example topology for WCETT's non-isotonicity.

### A. Requirements of Path Weight Functions

Due to the shared nature of the wireless medium, both *intra-flow interference* (interference between nodes on the path of the same flow) and *inter-flow interference* (interference between neighboring nodes) exist in mesh networks and affect the load on nodes. Hence, to balance network load, the first requirement for path weight functions is to capture both intra-flow and inter-flow interference.

In addition, a path weight function must have a fundamental property, called *isotonicity* [8], [9].

*Definition 1:* A weight function $W(\cdot)$ is isotonic if $W(a) \leq W(b)$ implies both $W(a \oplus c) \leq W(b \oplus c)$ and $W(c' \oplus a) \leq W(c' \oplus b)$ for all paths $a, b, c, c'$, where operator $\oplus$ represents the concatenation of two paths.

This isotonicity property is needed to ensure efficient routing algorithms, such as link-state or distance-vector algorithms, can find minimum weight and loop-free paths based on the path weight function. Without isotonicity, to find minimum weight and loop-free paths requires algorithms with exponential complexity, which is intractable even for small networks.

### B. Existing Path Weight Functions

Although most existing path weight functions, such as hop count, ETX [2] and ETT [3], are isotonic, they do not consider interference between nodes and hence cannot balance load in mesh networks.

The only interference-aware path weight function is WCETT [3]. Unfortunately, WCETT only considers intra-flow interference and is not isotonic. WCETT of a path $p$ is:

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

where $0 \leq \beta \leq 1$ and $ETT_l$, as introduced in [3], is the expected transmission time of a packet at link $l$. $X_j$ is the number of times that channel $j$ is used along path $p$. While ETT represents the capacity of wireless channels, $X_j$ captures the intra-flow interference of path $p$. However, since inter-flow interference is not considered, WCETT may create severe inter-flow interference by routing flows to dense areas.

The non-isotonic nature of WCETT can also cause severe problems as shown by the topology in Figure 1. In this figure, two numbers are associated with each link, the ETT and the

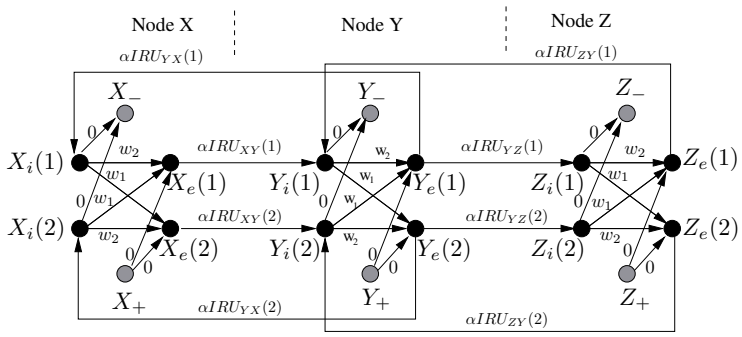



Fig. 2. Virtual nodes

channel number (CH), respectively. Assuming $\beta$ in Equation (1) is set to 0.5, the minimum weight path from $S_1$ to $T$ is $S_1 \rightarrow B \rightarrow T$. However, due to the non-isotonicity 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 1. When node $S_2$ calculates its path to $T$, Dijkstra's algorithm correctly indicates $S_2 \rightarrow S_1 \rightarrow B \rightarrow T$ as the minimum weight path, depicted as the shadowed arrows in Figure 1. Hence, a forwarding loop is formed between $S_1$ and $S_2$.

## III. MIC Path Weight Function

Due to the limitations of existing path weight functions, we propose our novel path weight function, MIC, to support load-balanced routing. MIC not only captures both intra-flow and inter-flow interference, but also can be decomposed into isotonic link weight assignments so that minimum weight paths can be easily found and no forwarding loop can be created.

### A. Definition of MIC

MIC is composed of two metrics: *Interference-aware Resource Usage (IRU)* and *Channel Switching Cost (CSC)*. The IRU metric captures the effects of inter-flow interference and the CSC metric captures the impacts of intra-flow interference.

IRU is defined as follows:

$$IRU_{ij}(c) = ETT_{ij}(c) \times |N_i(c) \bigcup N_j(c)|, \tag{2}$$

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 neighbors that may be interfered with by the transmission activities between node $i$ and node $j$ over channel $c$. IRU captures the aggregated channel time of these neighbors consumed by the transmission of the flow between nodes $i$ and $j$, essentially representing the inter-flow interference that the flow may impose on the network.

To capture intra-flow interference, CSC at a node $X$ is:

$$CSC_X = \begin{cases} w_1 & \text{if } CH(prev(X)) \neq CH(X), \\ w_2 & \text{if } CH(prev(X)) = CH(X), \end{cases} \tag{3}$$

$$0 \leq w_1 < w_2, \tag{4}$$

where $prev(X)$ is the channel used by the previous hop of node $X$ and $CH(X)$ is the channel that node $X$ uses to transmit to the next hop. 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.

Combining both IRU and CSC, we define MIC as follows:

$$MIC(p) = \alpha \sum_{\text{link } l \in p} IRU_l + \sum_{\text{node } i \in p} CSC_i, \tag{5}$$

where $p$ stands for a path in the network and the positive factor $\alpha$ is introduced to represent the trade-off between minimizing intra-flow and inter-flow interference and is defined as follows:

$$\alpha = 1/[N \times \min(ETT)], \tag{6}$$

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 $\alpha$ with Equation (5), IRU is scaled up to be around the same value range as $CSC$ and its weight is normalized to the overall network size.

### B. Isotonic Decomposition of MIC

Although MIC captures interference, it is not isotonic if used directly. Therefore, in this section, we map a real network into a virtual network and decompose MIC into isotonic link weight assignments in this virtual network. Due to this isotonic decomposition, efficient algorithms, such as Bellman-Ford and Dijkstra's algorithms, can easily calculate minimum weight paths using $MIC$ without creating forwarding loops.

To decompose MIC, several virtual nodes are used to represent the possible arrival/departure channels of packets at node $X$. Then, MIC can be translated into isotonic link weight assignments between these virtual nodes as shown in Figure 2.

For each channel $c$ of a real node $X$, two virtual nodes $X_i(c)$ and $X_e(c)$ are introduced, where $X_i(c)$ represents that packets arrive at node $X$ from channel $c$ and $X_e(c)$ stands for that node $X$ transmits packets on channel $c$. Link $(X_i(c), X_e(c))$ represents that node $X$ does not switch channel while forwarding a packet and hence weight $w_2$ is assigned to this link to capture the CSC value in this case. Similarly, weight $w_1$ is assigned to link $(X_i(c), X_e(c1))$, where $c \neq c1$. To represent the IRU metric, if node $X$ is able to transmit to node $Y$ using channel $c$, link $(X_e(c), Y_i(c))$ with weight $IRU_{XY}(c)$ is added between $(X_e(c)$ and $Y_i(c))$. Node $X$ also has two additional virtual nodes: $X_-$ and $X_+$. $X_-$ is the destination virtual node for flows destined to the real node $X$ and every $X_i(c)$ has a link with weight 0 to $X_-$. $X_+$ is the source virtual node for flows that start at the real node $X$ and $X_+$ has a link with weight 0 to every $X_e(c)$.

By creating this virtual network, we essentially decompose MIC into isotonic link weight assignments on the virtual links between virtual nodes. Therefore, running Dijkstra or Bellman-Ford algorithms on this virtual network creates minimum weight paths for $MIC$ and no forwarding loop can be formed.

### C. Implementation of MIC

An important implementation issue of $MIC$ is the calculation of $IRU$, which includes the estimation of both $ETT$ and $N_i$. The calculation of $ETT$ can be achieved by periodically broadcasting hello messages at each node in the network as shown by [2], [3]. These hello messages do not necessarily increase the control message overhead of the network since they can be piggybacked on the route update messages of routing protocols. For $N_i$, since mesh networks are static, we can determine whether two nodes are in each other's interference range by measuring the correlations between the broadcasting rates of these two nodes at the time when the network is established (see the work of Agarwal et. al [1] for details).

## IV. Using MIC in Routing Protocols

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 Section IV-A, we discuss the architecture of routing tables and in Section IV-B, we discuss how to build the routing tables.

### A. Routing Table Architecture

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.

*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. 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 MIC, such an increase is acceptable since the number of radios in a node is usually very small, resulting in a small number of routing tables that a node needs to store. In addition, each routing table is not very large due to the relatively small size of mesh networks. 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.

### B. Building Routing Tables

The major benefit of MIC compared to WCETT is that it can be used with any routing algorithm, including both Bellman-Ford or Dijkstra's algorithms, and is guaranteed to find the minimum weight paths without creating any forwarding loops.

*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 and is able to construct the virtual network according to the description in Section III-B and use this virtual network to build its routing tables.

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 $\langle Z, Y, c_0 \rangle$. Similarly, the central routing table $T_+$ for node $X$'s own traffic, can be built by running Dijkstra's algorithm at virtual node $X_+$.

*2) Using Distance-Vector Routing:* In traditional distance-vector routing, a node $Y$ maintains and informs its neighbors about the weights of its minimum weight paths to every node in the network. When using MIC in distance-vector routing, node $Y$ maintains and informs its neighbors about the weights of its minimum weight paths from its ingress virtual nodes (e.g., $Y_i(c)$ for some channel c) to every node's destination virtual node (e.g., virtual node $Z_-$ at node $Z$). The information should have a form of $\{Y_i(c) \rightarrow Z_-$, weight $MIC(Y_i(c), Z)\ \}$ so that upon receiving this information from Node $Y$, node $X$, a neighboring node of node $Y$, can update its minimum weight paths from its own ingress virtual nodes (e.g. $X_i(c)$) to every node's destination virtual node (e.g., $Z_-$ at node $Z$).

## V. 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 routing protocols that use MIC 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 (3) is extended to:

$$CSC_X = \begin{cases} w_2, & \text{if } CH(prev^2(X)) \neq CH(X) = CH(prev(X)), \\ w_3, & \text{if } CH(prev^2(X)) = CH(X) \neq CH(prev(X)), \\ w_2 + w_3, & \text{if } CH(prev^2(X)) = CH(X) = CH(prev(X)), \\ w_1, & \text{otherwise,} \end{cases} \tag{7}$$

$$0 \leq w_1 \ll w_3 < w_2, \tag{8}$$

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 $w_3 = 0.6w_2$ to capture this relationship.

To incorporate this new CSC definition, the construction of the virtual networks and the actual routing procedures of MIC need to be extended. Briefly speaking, 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. At least $m(m+1)$ ingress virtual nodes and $m(m+1)$ egress virtual nodes are need for this node. This means that this node must maintain $m(m+1)+1$ routing tables to consider two-hop intra-flow interference (For details see our technical report [11]). To capture intra-flow interference between nodes that are further away, the number of routing tables and calculation complexity for computing routing tables increase even further. 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 MIC is not significant. Given the computation and memory overhead associated with capturing multihop intra-flow interference, considering multi-hop interference in MIC may not be necessary.

## VI. Evaluation

To demonstrate the effectiveness of our new path weight function MIC, our evaluation includes two parts using NS2 simulations [4]. First, we demonstrate the effectiveness of MIC by comparing MIC's performance with hop count, ETT and WCETT. Then, we examine the sensitivity of MIC's performance to the choice of $w_2$ and $w_3$ in CSC definition.

### A. Evaluation Metrics

To evaluate the performance of MIC, we adopt four different evaluation metrics. The first two metrics are total network throughput and average end-to-end packet delay. Although these two metrics essentially reflect the average network service quality to users, they cannot capture the load distribution in the network. It is important to distribute the network load evenly so that each user can have their fair amount of channel utilization and no user is starved due to local congestion. Here, the channel utilization of channel $c$ at node $i$'s location, denoted as $u_i^c$, is the fraction of time that this channel is busy due to communication activities at node $i$ or node $i$'s neighboring nodes. We uses two additional metrics: the *maximum channel utilization ($M$)* and the *cost of channel utilization ($\Phi$)* to reflect the routing protocol's ability in achieving even distribution of channel utilization. These two metrics are defined as follows:

$$M = \max_{i \in V, c \in C(i)} u_i^c \tag{9}$$

$$\Phi = \sum_{i \in V} \sum_{c \in C(i)} \varphi(u_i^c), \tag{10}$$

where $C(i)$ is the set of channels in node $i$ and $V$ is the set of nodes in the network. $\varphi(\cdot)$ 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 [5], [10]:

$\varphi(0) = 0$ and

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

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 represent traffic congestion at this channel.

### B. Performance of MIC

To examine MIC performance compared to hop count, ETT, WCETT, we randomly generate ten $1000m \times 1000m$ networks, each with 100 nodes and 1 TAP. Every node has two radios and each radio can be configured to one of three channels. In each network, we have 20 flows destined to the Internet through the TAP. The sources of the flows are randomly located and all flows are CBR flows with 512 Byte packets. The routing protocol is distance vector routing. For MIC, we set $w_2 = 0.5$ and $w_3 = 0.3$ (See Equations (3) and (7)). 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.

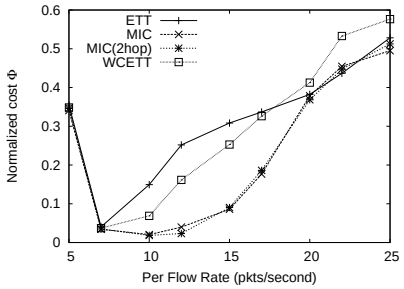



Fig. 3. Cost ($\Phi$) of $MIC$, $MIC(2hop)$, $ETT$ and $WCETT$, normalized to the cost of hop count.

Figure 3 depicts the cost of channel utilization ($\Phi$), where MIC (2hop) represents the extended version of MIC for considering two hop intra-flow interference. Among all of the solutions, the node utilization cost of MIC and MIC (2hop) is the lowest since MIC balances network load and reduces both intra-flow and inter-flow interference. The difference between MIC and MIC (2hop) is very small and hard to distinguish. Figures 4(a), 4(b) and 4(c) show the maximum channel utilization among nodes, the total network throughput and the average end-to-end packet delay. The performance of MIC and MIC (2hop) is much better than the performance of ETT, WCETT and hop count due to the balancing of traffic load. MIC (2hop) has a slightly better performance than MIC due to its consideration of two-hop intra-flow interference.

### C. Sensitivity to $w_2$ and $w_3$ weight in CSC

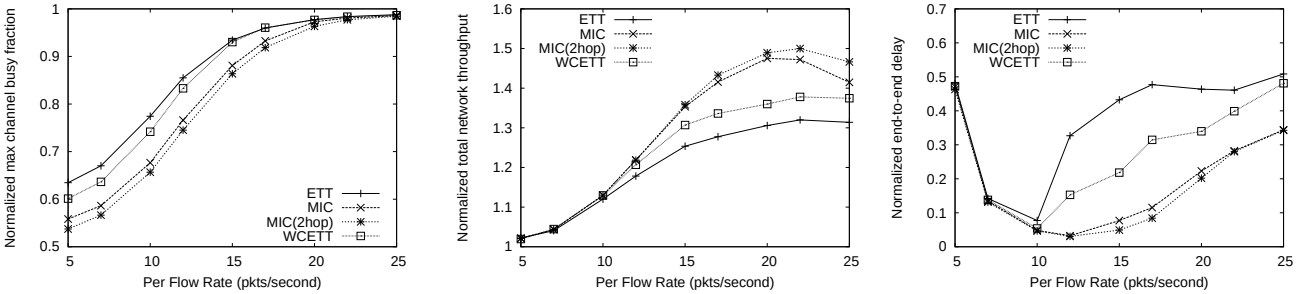
To understand whether the performance of MIC is sensitive to the choice of $w_2$ and $w_3$ weight in the definition of CSC, in this set of simulation, we test different $w_2$ and $w_3$ configurations on ten randomly generated $1000m \times 1000m$ networks, each with 100 nodes, 20 flows and 1 TAP. The range of $w_2$ changes from 0.3 to 5 and the range of $w_3$ changes from 0.18 to 3. Figures 5 and 6 show how the performance of MIC and MIC(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 MIC. However, MIC(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., $w_2 = 5$, $w_3 = 3$), MIC(2hop)'s performance can be even worse than MIC. Considering the fact that MIC(2hop) is more expensive and its improvement over MIC is not significant even if $w_2$ and $w_3$ is chosen appropriately, we conclude that MIC is more preferably in real networks.

## VII. Conclusions and Ongoing Work

In this work, we have proposed for the first time a path weight function, MIC, that captures both inter-flow and intra-flow interference. MIC enables efficient calculation for minimum weight path and loop-free routing. Hence, MIC is compatible with both Bellman-Ford and Dijkstra's algorithms, which are

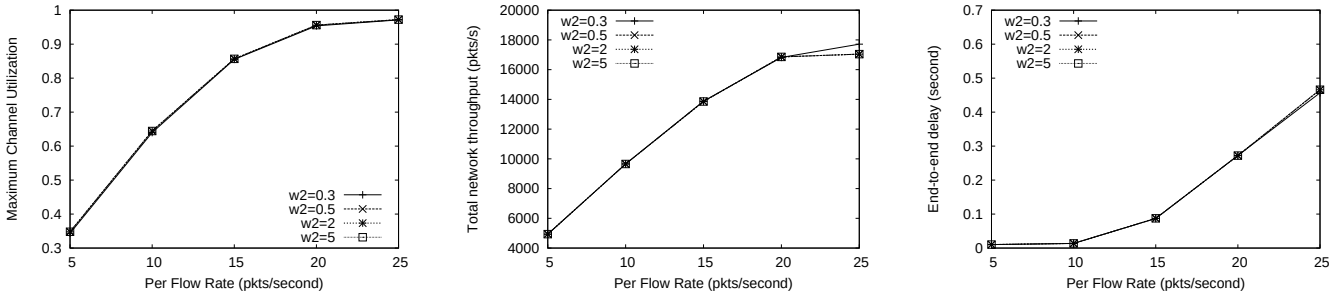
TABLE I
DISTANCE/RATE RELATIONSHIPS

| 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 |



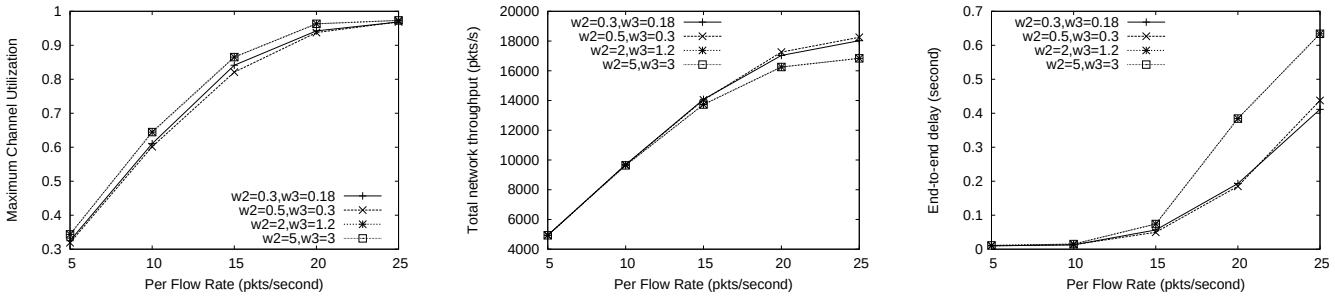

(a) Maximum channel utilization (b) Total network throughput (c) Average end-to-end packet delay

Fig. 4. Performances of $MIC$, $MIC(2hop)$, $ETT$ and $WCETT$, normalized to the performance of hop count.




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

Fig. 5. Sensitivity of MIC to weight $w_2$




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

Fig. 6. Sensitivity of MIC(2hop) to weight $w_2$ and $w_3$

the most efficient algorithms for calculating minimum weight paths. Our future work for MIC is to investigate the trade-off of setting the $w$'s in Equation (3) and $\alpha$ in Equation (5). 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 $\alpha$ affects the delay and throughput of flows and the overall load on the network. Finally, we will investigate how to integrate MIC with mesh networks that have both mobile and static nodes.

## REFERENCES


[1] 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.

[2] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *ACM Mobicom*, 2003.

[3] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *ACM Mobicom*, 2004.

[4] K. Fall and K. Varadhan. NS notes and documentation. In *The VINT Project, UC Berkely, LBL, USC/ISI, and Xerox PARC*, 1997.

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

[6] V. Gambiroza, B. Sadeghi, and E. Knightly. End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks. In *ACM Mobicom*, 2004.

[7] R. Karrer, A. Sabharwal, and E. Knightly. Enabling Large-scale Wireless Broadband: The Case for TAPs. In *Proceedings of HotNets*, Cambridge, MA, 2003.

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

[9] J. L. Sobrinho. Network Routing with Path Vector Protocols: Theory and Applications. In *ACM SIGCOMM*, pages 49–60, 2003.

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

[11] Y. Yang, J. Wang, and R. Kravets. Interference-aware Load Balancing for Multihop Wireless Networks. Technical Report UIUCDCS-R-2005-2526, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.