Wireless Netw (2010) 16:527–543
DOI 10.1007/s11276-008-0151-0

Routing in the frequency domain

Jay A. Patel · Haiyun Luo · Indranil Gupta

Published online: 4 December 2008

Abstract The design of single transceiver based multi-channel multi-hop wireless mesh networks focuses on the trade-off between rapid neighbor synchronization and maximizing the usage of all available channels. Existing designs are confined to the MAC layer and scale poorly as the network grows in coverage and density. We recently proposed Dominion as a cross-layer architecture that includes both medium access control and routing. Dominion eliminates the need for neighbor synchronization at the MAC layer and pushes the intelligence up the network stack. At the MAC layer, a node switches channels according to a *deterministic* schedule which guarantees that a node converges with each of its neighbors periodically. At the network layer, the channel-hopping aware routing substrate routes traffic along the *frequency domain*, i.e., packets along a multi-hop route generally traverse via multiple channels. In this paper, we present the complete design, analysis and evaluation of Dominion and make four new contributions. Firstly, we extend Dominion to support goal-oriented routing: source nodes can locally choose to maximize throughput or minimize end-to-end latency without requiring any changes in the network. Secondly, we describe a technique that eliminates intra-flow interference. In absence of extrinsic interference, Dominion now allows network flows to maintain constant throughput and deterministic end-to-end latencies irrespective of distance. Thirdly, via theoretical modeling and analysis, we provide expected throughput and end-to-end latencies for network flows. Finally, via extensive QualNet simulations we show that Dominion achieves 1064% higher throughput than IEEE 802.11 while being 299% fairer.

Keywords Multi-channel networks · Cross-layer design · Wireless mesh networks · Multi-path routing

J. A. Patel (✉) · H. Luo · I. Gupta
Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Avenue, Urbana, IL 61801, USA
e-mail: jaypatel@cs.uiuc.edu

H. Luo
e-mail: haiyun@cs.uiuc.edu

I. Gupta
e-mail: indy@cs.uiuc.edu

1 Introduction

In recent times, numerous wireless mesh networks [4, 6, 20] have been deployed to provide Internet connectivity to their local communities. However, the growth of such networks may have been inhibited due to a scalability problem. As most contemporary wireless mesh networks using IEEE 802.11 (hereon, simply 802.11) operate on the same channel to maintain network connectivity, network flows are prone to intra-flow and inter-flow interferences. This problem is further exaggerated as most mesh networks feature only a small number of *gateway* nodes (often as few as one), where the traffic to and from the Internet converges. The interference severely affects the quality of service to end-users during peak times. To limit service degradation, most mesh networks do not span more than a few hops from a gateway node.

A popular method to scale wireless mesh networks is to exploit an off-the-shelf 802.11 transceivers' capability to switch channels.[1] In a multi-channel network, a node and

[1] We are concerned only with orthogonal channels and not overlapping channels. Presently, 802.11a and 802.11b/g have 12 and 3 such channels respectively.

its neighbor must be switched to the same channel (*synchronized* or *converged*) to exchange data packets. At the same time, disjoint links should remain on distinct channels to maximally exploit multi-channel diversity. An effective neighbor synchronization mechanism that scales with the number of channels remains a challenging problem. Multichannel MAC (MMAC) [26] and Slotted Seeded Channel Hopping (SSCH) [1] are two existing designs that exploit multi-channel diversity using only a single commodity transceiver. MMAC and SSCH are MAC layer solutions that provide distinct neighbor synchronization mechanisms to maintain network connectivity. We detail the challenges faced by both in Sect. 2.

To mitigate issues with neighbor synchronization, we presented a preliminary design of Dominion [23] as a cross-layer architecture for *stationary* wireless mesh networks. The design of Dominion spans both the MAC and the network layers. Dominion entirely avoids the problem of neighbor synchronization at the MAC layer by pushing intelligence up the network stack. At the MAC layer, Dominion divides the network into distinct logical subnetworks. Each subnetwork is assigned a static and deterministic channel hopping schedule from which the member nodes do not deviate. The schedule guarantees that all pairwise subnetworks converge periodically. Nodes converging on a channel access the wireless medium via CSMA using 802.11. At the routing layer, a routing substrate aware of the MAC layer's channel-hopping schedule routes traffic along the frequency domain, i.e., packets along a multi-hop route generally traverse via multiple channels. Dominion's cross-layer architecture additionally simplifies the MAC implementation as both the core logic and the buffers for data packets reside at the routing layer instead of at the MAC layer.

In this paper, we present the complete design, analysis and evaluation of Dominion and make four new contributions. Firstly, we extend the previously proposed graph-theoretic model for the multi-channel wireless mesh network to support goal-oriented routing. A source node may locally select to maximize throughput or minimize end-to-end latency (based on the needs of the application) without requiring any changes in the network. Concretely, routing for a different goal (i.e., increased throughput or reduced end-to-end latency) is as simple as applying a different set of weights to the edges of an abstract graph. Secondly, within the routing substrate itself, we improve the algorithm to eliminate intra-flow interference within multi-hop routes. In absence of extrinsic interference, Dominion end-nodes can now sustain constant throughput between themselves irrespective of their distance, i.e., hop count. By locating and using multiple intra-flow interference free routes, Dominion is able to multiplex the attainable throughput. Thirdly, we thoroughly analyze the properties of the routing algorithm and give expected values for throughput and end-to-end latencies. Finally, using the QualNet [24] network simulator, we present: (i) detailed benchmarks that characterize the effect of the various design space parameters; and (ii) comparative evaluations that highlight the differences between SSCH and Dominion. Not surprisingly, the results closely match our analytical observations. As a baseline comparison, our experimental results show that Dominion improves throughput by 1064% and fairness by 299% over 802.11. Against SSCH, Dominion improves throughput by 93% and fairness by 291%.

The rest of the paper is organized as follows: Sect. 2 presents closely related existing designs. Dominion's MAC layer and the channel-hopping schedule is presented in Sect. 3. Meanwhile, the abstract graph model, goal-oriented routing, and routing strategies are discussed in Sect. 4. Next, in Sect. 5, we analyze the expected throughput and end-to-end latency of routes. The experimental evaluation is presented in Sect. 6. Section 7 reviews other related work, and lastly, we conclude in Sect. 8.

2 Preliminaries

In this section, we briefly discuss existing single transceiver solutions and the challenges faced by each of them.

Since switching a node's transceiver to another channel incurs a delay (around 80 μs [11]), solutions that exploit multi-channel diversity with a single transceiver use *timeslots* to amortize the cost of a channel switch. A node switches to a given channel for the duration of a timeslot during which it may transmit multiple packets. For example, a timeslot duration of 10 ms is sufficient to transmit approximately 28 1024-byte UDP packets at the maximum 802.11a MAC bit rate of 54 mbits/s. It should also be noted that single transceiver solutions that exploit multi-channel diversity assume some form of time synchronization across nodes. As shown by Bahl et al. [1], loose time synchronization (± 1 ms) provided by software-aided methods [10] should adequately suffice.

Based on the 802.11 Power Saving Mechanism, Multichannel MAC (MMAC) [26] nodes converge on a predefined rendezvous control channel at the start of a timeslot. During a convergence period, nodes broadcast their pending traffic queues. Based on information obtained from its neighbors, a node determines the channel it switches to until the next convergence period. A node attempts to converge with the neighbor for which the node has the most pending packets. Due to the need for a control channel, MMAC scales poorly as the number of channels increase.

Slotted Seeded Channel Hopping (SSCH) [1] improves upon MMAC by eliminating the need for a control channel by using optimistic synchronization. Each node generates its own cyclical *schedule* based on a random seed and modulo arithmetic. The schedule guarantees that each node converges with a given neighbor at least once every schedule cycle. As nodes locally broadcast their schedule once every timeslot, nodes learn (or update) the schedule of their neighbors during the periodic convergence. To maintain a network flow, a node synchronizes its schedule (i.e., deviates from its current schedule) with its next hop neighbor. To allow synchronization with multiple neighbors simultaneously, SSCH permits partial synchronization by empirically hopping across 4 different schedules (the authors term this *optimistic synchronization*). Areas where the network density and traffic loads are high, e.g., areas close to the gateways, optimistic synchronization with multiple neighbors becomes increasingly difficult. A node may synchronize with a neighbor, only to find that the neighbor has deviated away from that schedule if the neighbor has its own local traffic to satisfy.

Other related work is discussed in Sect. 7.

3 MAC: channel-hopping schedule

Unlike MMAC and SSCH, Dominion does not use individual schedules per node. Instead, Dominion divides nodes into distinct logical subnetworks. Each subnetwork is assigned a deterministic channel hopping schedule. The schedule guarantees that all pairwise subnetworks converge periodically. The convergence of subnetworks allows a node to communicate with any of its neighbors irrespective of their home subnetworks. Once nodes converge on a given channel, the wireless medium is accessed using 802.11. In this section, we first present a preliminary channel hopping (scheduling) algorithm where the number of available channels k is a prime. Next, we generalize the algorithm to an arbitrary number of channels. Note that the presented algorithm allows each node to independently calculate the channel schedule for any given subnetwork. This allows a node to communicate with a neighboring node as long as it aware of the subnetwork that its neighbor belongs to. As such, lastly, we briefly discuss the assignment of nodes to subnetworks.

3.1 Preliminary schedule

The preliminary scheduling algorithm assumes that the number of available channels k is a prime. The nodes are grouped into k subnetworks (labeled s_0 through s_{k-1}). We use modulo arithmetic to facilitate guaranteed subnetwork convergence. Concretely, the following function $\mathcal{C}$ is used to determine the schedule for subnetwork s_i during timeslot t:

$$\mathcal{C}(s_i,t) = \begin{cases} 0, & \text{if } i=0, t=0; \\ \mathcal{C}(s_0, i-1), & \text{if } 0<i<k; \\ [\mathcal{C}(s_i, t-1)+i] \bmod k, & t>i \end{cases} \qquad (1)$$

The first subnetwork s_0 starts on channel 0 at $t = 0$.[2] Every other subnetwork s_i converges with the first subnetwork at $t = i - 1$ to determine it's *seed*, i.e., starting point. The rest of the schedule (for values of $t > i$) is calculated using modulo arithmetic with the subnetwork identifier i as the additive constant.[3] This implies that subnetwork s_0 remains on channel 0. For all other subnetworks s_i, the channel schedule is calculated based on its schedule for the previous timeslot.

The preliminary scheduling algorithm has the following properties:

- Starting with $t = k$, the *schedule cycle* repeats every k timeslots. The schedule can be minimally represented via one schedule cycle.
- Each of the k subnetworks converges with every other $k - 1$ subnetwork exactly once during the schedule cycle, i.e., every k timeslots. This implies that each subnetwork spends one timeslot (per schedule cycle) in *solitude*, i.e., not converging with a foreign subnetwork. Given this, only $\frac{k+1}{2}$ channels are "occupied" during any given timeslot.

To utilize all k channels simultaneously, the channel hopping schedule can be generated *as if* there were $2k - 1$ available channels. Note that this new schedule assumes that $2k - 1$ is prime, irrelevant of the primeness of k. While this new schedule uses $2k - 1$ total channels, only $\frac{2k-1+1}{2} = k$ channels are occupied during any given timeslot. Thus the number of total channels can be reduced to k on a per-timeslot basis. For each timeslot t, starting with s_0 (and downwards to s_{2k-2}), the values of $\mathcal{C}(s_i, t)$ and $\mathcal{C}(s_j, t)$ are replaced with a value from [0, $k - 2$] in sequential order if $\mathcal{C}(s_i, t) = \mathcal{C}(s_j, t)$, i.e., if the subnetworks converge. As described before, a single subnetwork observing solitude will remain—this subnetwork is assigned to channel $k - 1$.

Assuming a uniform distribution of nodes amongst the subnetworks, only half as many nodes will be switched to channel $k - 1$ as any other channel—every other channel will have exactly two converging subnetworks. To mitigate this slight discrepancy in the nodes-to-channels distribution, we "add" a new subnetwork (i.e., s_{2k-1}) that converges with this *solitudnal subnetwork*. Adding a new subnetwork improves the probability of nodes in the

[2] Channels are 0-offset.

[3] Note that there is no schedule for $t < i$, however, this will soon be mitigated.

previously-solitudnal network having a currently convergent neighbor. More concretely, the probability of a previously-solitudnal node having a converging neighbor improves from $\frac{1}{2k-1}$ to $\frac{1}{k}$, i.e., nodes can now have neighbors converging in the new subnetwork s_{2k-1} instead of in its own subnetwork alone.

3.2 Generalizable schedule

The schedule can be generalized to an arbitrary k channels by first generating a schedule *as if* $P(2k-1)$ channels are available, where $P(x)$ is the smallest prime number greater than or equal to x. Immediately, subnetworks higher than s_{2k-1} are discarded. Next, the number of channels is reduced to k (as described before). However when $2k-1$ is not prime, in addition to the one solitudnal network, there may be a few additional (up to $P(2k-1)-2k-1$) pseudo-solitudnal networks, i.e., these subnetworks converge with the now discarded subnetworks. These solitudnal subnetworks can be converged on the remaining channels (one pair at a time). At worse, given that $P(2k-1) > 2k-1$, some subnetworks converge multiple times during a schedule cycle. Note that each subnetwork still converges with every other foreign subnetwork at least once during the schedule cycle.

The deterministic schedule requires that the network be divided into $S = 2k$ distinct subnetworks. Furthermore, it provides a compact schedule cycle of $T = P(2k-1)$, where $P(x)$ is the smallest prime greater than or equal to x. Since the schedule repeats every T timeslots, we use the notation t_j to indicate all timeslots $t \equiv j (\text{mod}\, T)$. As an example, we present a schedule in Table 1 for a network with 4 available channels. In this example, the schedule cycle duration T is 7 timeslots. Each of the 8 subnetworks converge with the other 7 subnetworks during the schedule cycle. To further illustrate this example, Fig. 1(a) shows a sample 802.11 network with 64-nodes. Figure 1(b)–(h) show the state of a network using the Dominion MAC (nodes are assigned to subnetworks using an uniform random function). The aggregation of the temporal topologies yields the same connectivity as the single-channel 802.11 topology.

Table 1 Schedule for a 4 channel network

$\mathcal{C}(s_i, t_j)$	t_0	t_1	t_2	t_3	t_4	t_5	t_6
s_0	0	0	0	0	0	0	3
s_1	0	3	1	1	1	1	0
s_2	1	0	1	3	2	2	1
s_3	2	1	0	1	2	3	2
s_4	3	2	2	0	1	2	2
s_5	2	2	3	2	0	1	1
s_6	1	1	2	2	3	0	0
s_7	3	3	3	3	3	3	3

Subnet s_i switches to channel $\mathcal{C}(s_i, t_j)$ during timeslot t, where $t \equiv j$ (mod 7)

3.3 Subnetwork assignment

An ideal subnetwork designation assigns an equal number of nodes to each of the S subnetworks either randomly or based on spatial reuse. A one-way uniform hashing function (for example, the SHA-1) may be used to determine a node's home subnetwork. Stated differently, a node A's home subnetwork may be determined by simply using the following mathematical operation: `SHA1(MacAddress(`A`))` mod S. Another way is based on spatial reuse: using a globally optimal assignment [19] (possible for synthetic networks) or locally based on two-hop neighborhood [16].

4 Routing

As a Dominion network simultaneously utilizes multiple channels, it naturally incurs reduced intra-flow and inter-flow interference as compared to 802.11. As a result, Dominion tends to sustain higher throughput when experiencing multiple simultaneous flows. However, a routing strategy oblivious to the underlying channel-hopping schedule may yield undesirably high end-to-end-latencies. An example is presented in Sect. 4.1. In this section, we discuss strategies to minimize end-to-end latency, an approach to eliminate intra-flow interference altogether, and further optimize throughput by using multiple routes. The proposed solution entails the routing substrate to convert the network link-state into an abstract graph, which can be use to calculate routes that either minimize the end-to-end latency or maximize the throughput. The novelty of the proposed routing approach is that changing between the two supported goals is as simple as changing the assignment of edge weights.

4.1 The case for intelligent routing

If data packets need to reach the destination node rapidly ("as soon as possible"), routing for low end-to-end latency remains an important goal. Using a 4 channel network (see Table 1), Fig. 2 illustrates a scenario where the shortest path (based on hop count) incurs a higher end-to-end latency than a longer path. Suppose node A (belonging to s_3) needs to send a single packet to a neighboring node B (in s_4) at t_0. Subnetworks s_3 and s_4 do not converge until t_6. Hence, node A "stalls" for 6 timeslots. However, if node A relays the packet via intermediate node C (in s_5), the packet can reach the node B at timeslot t_1.

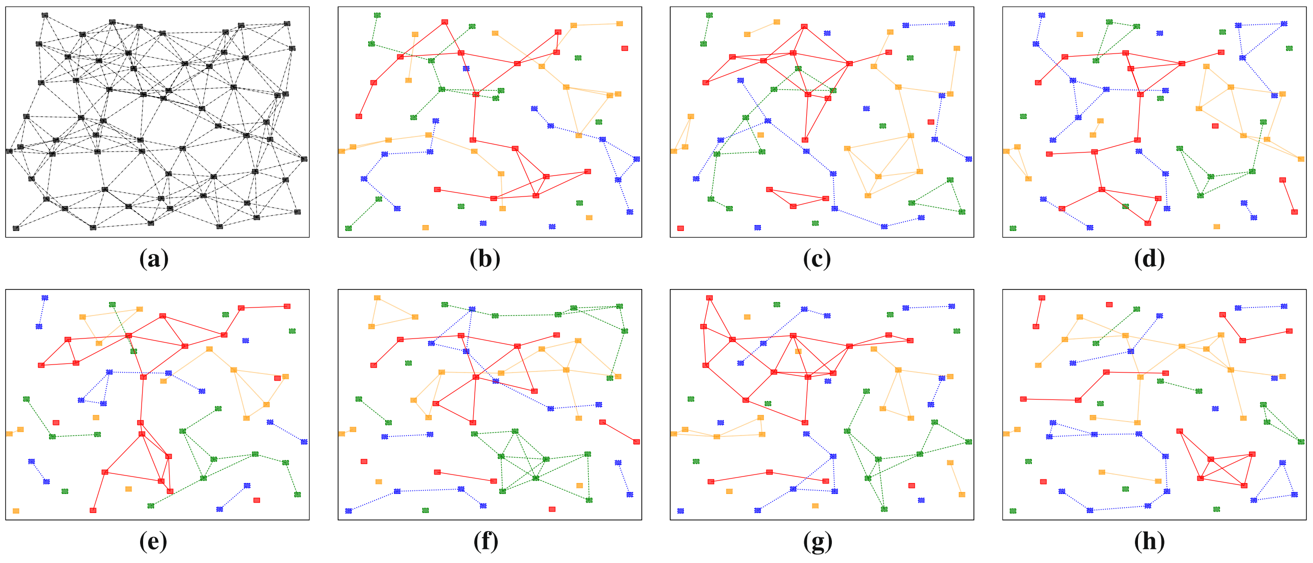

Fig. 1 A Dominion network maintains the same connectivity as a single-channel network, while fully utilizing all available channels (in this example, 4 channels). Note that the aggregation of Dominion's temporal topologies yields the same connectivity as the single-channel 802.11 topology. **a** 802.11 topology, **b** Dominion topology at t_0, **c** Dominion topology at t_1, **d** Dominion topology at t_2, **e** Dominion topology at t_3, **f** Dominion topology at t_4, **g** Dominion topology at t_5, **h** Dominion topology at t_6

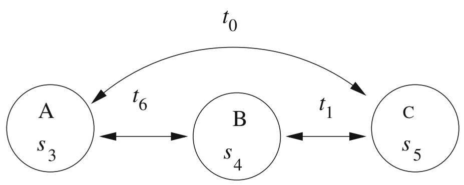

Fig. 2 A longer route may minimize the end-to-end latency

Due to the constant channel hopping, the route that minimizes end-to-end delay is time variant. For example, if node A desired to send a packet to node B at time t_1 (instead of at t_0), the single hop route AB will minimize the end-to-end latency. However, there are at most T unique routes (one per timeslot) that minimize end-to-end latency.

4.2 Eliminating intra-flow interference

A big drawback of single-channel routing protocols is that all transmissions share the same medium, which results in intra-flow interference within multi-hop routes. Intra-flow interference not only affects the throughput, but also adversely effects the end-to-end latency due to delays caused by contention. Multi-channel routing protocols are also prone to intra-flow interference (albeit to a lesser degree) if nodes along a route transmit on the same channel at the same time. Within Dominion, as nodes within a subnetwork operate on the same channel, intra-subnetwork hops induce intra-flow interference. We can eliminate intra-flow interference: (i) by assigning a specific time at which a hop should be *active*, i.e., when a node should transmit; and (ii) by selecting hops along a route that do not interfere with previous hops. Even though distant nodes may exploit spatial reuse, we conservatively require the later. Below, we present an overview of how these two goals can be achieved. The actual mechanism to derive an intra-flow interference path is described in Sects. 4.3 and 4.4.

With Dominion's deterministic channel-hoping schedule, a link is only active during certain timeslots. This can be verified by examining the (channel, time)-tuples set that describe the channel and timeslots when a link is active. For example, in Fig. 2, as nodes A and B converge on channel 2 at t_6, the link AB is characterized by the (channel, time)-tuples set $\{(2, t_6)\}$. The size of such a set depends on the link itself. For example, an intra-subnetwork link is characterized by T distinct tuples. However, most inter-subnetwork links are characterized by exactly one tuple. As mentioned in Sect. 3, only when $T \neq 2k - 1$, a few inter-subnetwork converge multiple times during a schedule cycle. In Sect. 4.3, we describe how a link that converges multiple times during a schedule cycle is assigned exactly one (channel, time)-tuple.

Given k available channels, and T as the length of the schedule cycle, there are $k{\cdot}T$ distinct (channel, time)-tuples. If every hop along a route is characterized by a distinct tuple, intra-flow interference is eliminated—as none of nodes will transmit *on the same channel, at the same time*. Section 4.4 presents the algorithm that selects such routes.

Ignoring channel switching delays and in absence of extrinsic interference, a single flow can maintain a constant throughput with a non-interfering route. The throughput would be approximately $\frac{1}{T}$th the MAC bit rate *irrespective* of distance (hops). Further, if there are m non-interfering routes such that none of hops of the m routes interfere with each other, the throughput increases to $\frac{m}{T}$th the MAC bit rate. Stated differently, throughput of Dominion flows depends on the number of non-interfering routes rather than on distance. Based on this premise, Dominion's routing substrate seeks to locate as many non-interfering paths as possible.

4.3 Abstract network model

A new abstract graph is constructed locally by a source node whenever it has a data packet for a previously unencountered destination node. Each physical node A is replaced with $T + 1$ *virtual nodes* in the abstract graph. The first T virtual nodes, labeled A^0 through A^{T-1}, are the *temporal nodes*. Each temporal node uniquely represents the physical node during each of the T timeslots of a schedule cycle. *Temporal edges* cyclically connect these temporal nodes, i.e., there is a one-way temporal edge from each node A^t to $A^{(t+1) \bmod T}$ for all $t < T$. Temporal edges represent the passage of one timeslot. Figure 3 expands node A (in a 4 channel network) from Fig. 2.

For each link in the physical topology, we add a virtual *connectivity edge* from temporal nodes A^t to B^t for all t during which their respective physical nodes converge. In the illustrated example, the physical links from node A to nodes B and C are translated to connectivity edges from A^6 to B^6 and A^0 to C^0, respectively. Note that a physical link with multiple convergence periods is now represented by multiple virtual edges, i.e., each connectivity edge is associated with exactly one (channel, time)-tuple. The final virtual node, labeled A^T, is a *base node* that represents the physical node itself. As routes start and terminate only end nodes, base nodes are required only for source and destination nodes. A base node is connected to its respective temporal nodes via *base edges*.

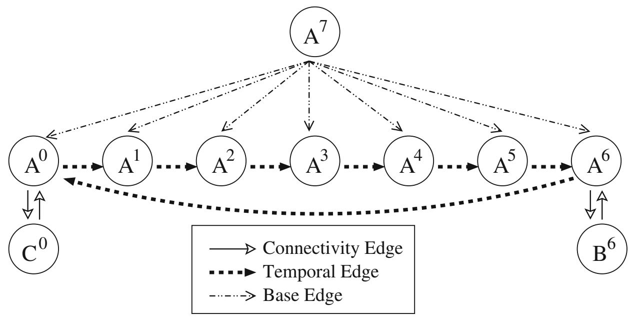

Fig. 3 Expansion of a physical node in the abstract graph model

4.4 Goal oriented routing

A node may want to minimize end-to-end latency (e.g., for multimedia traffic) or maximize throughput (e.g., for large file transfers). The routing substrate empowers a source node to make such decisions *locally* at run time, by simply changing edge weights of the abstract graph *on a per-flow basis*, and without *any changes* in the network. We assign edge weights based on the following plausible simplifications:

- The entire network is dormant, i.e., there is no network activity except for a *single packet* being transmitted from source node A to destination node B. This assumption allows us to ignore delays due to medium contention (which are not trivial to calculate in a distributed system).
- As both the time to switch to another channel and the time to transmit a single packet are substantially less than the duration of a timeslot, we ignore such delays. This allows us to focus on minimizing the aggregate stall time.

Next, we define the two primary goal-oriented routing substrates and describe their respective edge weight assignments.

- *Low Latency (LL) routing*: The goal is to minimize the aggregate stall time at each intermediate node. Hence, the weight assignment should aid the routing substrate in selecting a route with the minimum end-to-end delay. To this effect, each temporal edge is given a weight of 1 as the packet is stalled for one timeslot. Whereas each connectivity edge is given a cost of 0 as the packet is being currently transmitted.
- *High Throughput (HT) routing*: The goal of HT routing is to select the best quality links. Hence, each connectivity edge is weighted equivalent to the multiplicative inverse of the underlying physical link's probability to successfully forward a packet. Whereas each temporal edge is given a weight of 0 as it is irrelevant. This weight assignment can be used to locate the route with the smallest ETX cost [8]. With this weight assignment scheme, by giving a weight of 0 to temporal edges, it may seem that we are reducing the abstract graph back to the network link state. However, the abstract graph allows us to assign a schedule to the network hops (used to eliminate intra-flow interference—described next) and break ties between two similar routes. To break a tie between routes, the other weight system is used as a secondary key.

For both LL and HT routing, Dijkstra's algorithm [7] is used to find the shortest path between the respective base nodes of the source and the destination in the abstract graph. This path represents the shortest route from the source node A to destination node B, with each hop along the route *assigned* exactly one (channel, time)-tuple. To eliminate intra-flow interference, if the path contains two or more connectivity edges with the same tuple, all but one of them are *temporarily discarded*. If possible, the edges in the middle of the path are chosen to preserve the connectivity of source and destination nodes to their neighbors. This is a heuristical approach and may be suboptimal: it may prune the edge that actually yields the optimal solution. Nonetheless, in practice, we find that this approach suffices.

Dijkstra's algorithm is run repeatedly (up to a certain limit, which is discussed further in Sect. 5), until a non-interfering path is found. If a non-interfering path cannot be found, the shortest path is used as a fall-back. We show in Sect. 5 a non-interfering path can be found with high probability. If a non-interfering path is found, it is stored as a valid path and the process is re-run to locate more paths until no more non-interfering paths can be found. In essence, Dominion divides a network flow into multiple *subflows*. However prior to computing the next path, all temporarily discarded links are restored. Further, for each connectivity edge in the last located path, all *congruent* connectivity edges are pruned from the abstract graph. Two edges are said to be congruent if they are assigned the same (channel, time)-tuple. Pruning congruent edges eliminates "inter-subflow interference" and allows throughput to scale with the number of routes found.

The routing substrates described above are based on a centralized algorithm. It is made distributed by using a link state protocol, i.e., maintaining the link-state at each node.

4.5 Additional issues

TCP performance over single-channel multi-hop wireless networks remains a formidable challenge [2]. As such, we do not directly evaluate TCP performance in this paper. However, we take certain steps in improving Dominion to deliver packets in order. The ideas mentioned here along with tasks left as future work (mentioned in Sect. 8.1) should allow TCP to work fluidly atop a Dominion network.

As two subflows may have different end-to-end latencies packets may arrive out-of-order. Particularly, a subflow may originate at an earlier timeslot and deliver packets to the destination node later than another subflow. This poses a problem for transport protocols such as TCP which expect their packets to arrive in order. As a result, for LL routing an additional subflow is started only if it delivers packets in order. As is the case with locating an additional intra-flow interference path, a connectivity edge in the middle of the route is temporarily discarded and the algorithm is recomputed if an additional path delivers packet out-of-order. While this optimization reduces the number of packets arriving out-of-order substantially, the problem is not completely eliminated. Packets remaining in the buffer of an intermediate node will have to wait until the next schedule cycle to be forwarded—resulting in out-of-order delivery. As packet timeliness is only of secondary importance with HT routing, the routing substrate (or alternatively, the application itself) at the destination node can be directed to lazily resequence the packets.

If a node only needs to transmit a few packets (e.g., TCP ACKs), a variant of LL routing, Low Latency Now (LLN) routing can be used. The key idea is that since only a few packets need to be sent, it may be acceptable to have minimal subflow interference as throughput is of only secondary importance. Due to the cyclical channel hopping schedule, the shortest path (based on end-to-end latency) between a source and a destination node is time variant. We can simply use the shortest path available at the given instance—*now*—instead of computing and using a path that is part of a multi-path solution. To compute a LLN route, all base edges except one for the source node are pruned—the base edge to the current temporal node is maintained. The destination node's base edges remains unaffected. If packets cannot be delivered within the same timeslot, the remnant packets may use a different route causing minimal intra-subflow interference. However, packets will generally arrive in-order.

5 Analysis

In this section, we first present the expected end-to-end delay for both high throughput (HT) and low latency (LL) routing. To simplify our analysis, we focus on the shortest path located by both routing substrates. Stated differently, we do not address multi-path routing or the impact of intra-flow free routes. The experimental results presented in Sect. 6 show that empirical results with these options match the simpler analytical results. Next, to address the issue of algorithm complexity, we calculate the probability of a path being intra-flow interference free. With this information, we calculate an effective limit that Dijkstra's algorithm must be rerun to locate an intra-flow interference-free path. We finish by analyzing the time complexity of Dijkstra's algorithm on the abstract graph.

5.1 Expected end-to-end latency

The lowest (or best) delay for a path depends on its hop count. In the best case, a packet can depart the source node at the time of packet origin and continue its traversal to the destination node during the same slot. Therefore, the lowest delay for a route R is simply $\Omega(\text{delay}) = 0$. The highest (or worst) delay for a route is when each hop induces a stall time of $T - 1$ slots. The highest delay for a route R is $O(\text{delay}) = |R| \cdot (T - 1)$, where $|R|$ is the hop count.

The expected delay for a route is in between these two extremes. As the next hop for HT routing is chosen without regard to the delay, the expected end-to-end delay for the shortest HT route R is:

$$Ex[\text{delay}_{\text{HT}}] = |R| \cdot \frac{T}{2} \tag{2}$$

To validate our analysis, we plot the above result along side empirical results in Sect. 6.3.

The expected delay for a LL route depends on many variables but most importantly on the number of acyclic routes between a source and destination node. This is a non-trivial calculation for real topologies. As such, we simplify the problem by solving for the expected end-to-end delay between a source and any destination node $|R|$ hops away. This gives us the lower-bound on the expected end-to-end delay with LL routing. Further, we make the following assumptions: (i) the nodes are located as lattice points on a torus, (ii) the transmission range r is equivalent at each node, (iii) random assignment of nodes to subnetworks, and (iv) $2k - 1$ is prime, i.e., $T = 2k - 1$. Assumptions (i) and (ii) imply that each node will have exactly α neighbors, where the value of α depends on r and density of the lattice. Assumption (iv) implies that a subnetwork will converge with another foreign subnetwork exactly once during the schedule cycle. It is an acceptable assumption as it is true for both $k = 3$ and $k = 12$ (the maximum number of channels in 802.11b/g and 802.11a respectively).

We begin by calculating the expected delay for a node one hop away, and generalizing to a node $|R|$ hops away. However, before we delve further, the difference between the number of "available" neighbors for a source node and any of the intermediate nodes along a route should be noted. As routes are acyclic, an intermediate node only has at most $\alpha - 1$ available neighbors, i.e., one of the neighbors is already a predecessor in the path. We use $\alpha'(h)$ to imply the number of neighbors. In other words, $\alpha'(0) = \alpha$ for the first hop and $\alpha'(h) = \alpha - 1$ for any intermediate hop (i.e., $0 < h < |R|$) along the route.

With LL routing, a node transmits a packet to next node as soon as possible after packet reception (for a source node, the packet reception time is the packet generation time). We denote the time of packet reception as $t = 0$ for a given node. This stall time could be as low as zero if the two nodes are presently converging. However, this only occurs if the next node is part of the same subnetwork or is part of the presently converging foreign subnetwork. The probability that a node is converging with a *particular* neighbor at the time of packet reception is $\frac{1}{k}$ (i.e., 2 out of $2k$ subnetworks). The probability that *any* of the $\alpha'(h)$ neighbors belongs to either of these two subnetworks is given by $P_{\text{conv}(h,t=0)} = 1 - \left(1 - \frac{1}{k}\right)^{\alpha'(h)}$. If the hop does not converge at the time of packet origin, LL routing will select the neighbor that is first to converge. For any given slot t, where $t > 0$, the probability that a link to any one of its neighbor is active is dependent on: (i) the probability that the hop has not already converged, i.e., $1 - P(\text{conv}(h, t - 1)$; and (ii) the probability that there is a neighbor that converges with the currently converging subnetwork. For all slot t, where $t > 0$, the number of available subnetworks reduces by one as there is exactly one foreign subnetwork that converges per slot. This follows so that when $t = T - 1$, there is exactly one subnetwork remaining and convergence is guaranteed. Hence, the probability that a hop converges at time t is:

$$P_{\text{conv}(h,t)} = \begin{cases} 1 - \left(1 - \frac{1}{k}\right)^{\alpha'(h)}, & \text{if } t = 0; \\ \left(1 - P_{\text{conv}(h,t-1)}\right) \cdot \left(1 - \left(1 - \frac{1}{T-t}\right)^{\alpha'(h)}\right), & \text{otherwise.} \end{cases}$$

The expected delay at a single hop is the sum of all possible values of t multiplied by the probability that the hop will be active at this time. To get the total end-to-end latency for the route, this value is multiplied by the number of hops.

$$Ex[\text{delay}_{\text{LL}}] = \sum_{h=0}^{|R|-1} \sum_{t=0}^{T-1} P_{\text{conv}(h,t)} \cdot t \tag{3}$$

5.2 Intra-flow interference free path

Dijkstra's algorithm is repeatedly executed to locate an intra-flow interference-free path. To derive a mathematically intuitive limit for rerunning Dijkstra's, we first determine the probability that a selected path is intra-flow interference free. The HT routing metric chooses paths based on the ETX quality of connectivity edges and is oblivious to it's (channel, time)-tuple assignment. Recall from Sect. 3, there are total of $k \cdot T$ distinct (channel, time)-tuples. Generally, a connectivity edge is assigned a given tuple with probability $\frac{1}{k \cdot T}$ if using random assignment of nodes to subnetworks. There is an exception to this: a route with two consecutive intra-subnetwork hops features two consecutive connectivity edges with the same (channel, time)-tuple. However, this is trivially mitigated by picking one of the other $T - 1$ connectivity edges for the physical link. However, we take the conservative approach (which also simplifies the problem) and ignore such possibilities.

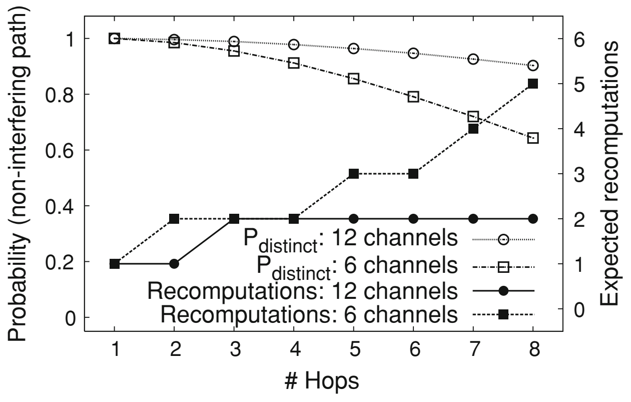

Fig. 4 The probability of a selected path being intra-flow interference free and the estimated number of computations to locate such a path

Hence, the probability of the shortest HT route R being intra-flow interference-free is:

$$P_{\text{distinct}} = \begin{cases} 0, & \text{if } |R| > k \cdot T; \\ \prod_{h=0}^{|R|-1} \left(1 - \frac{h}{k \cdot T}\right), & \text{otherwise.} \end{cases} \quad (4)$$

Figure 4 shows the probability of a selected path being intra-flow interference free on the primary y-axis (the left side). With 12 available channels, a route with $|R| = 6$ hops has a 0.95 probability of being intra-flow interference free. With 6 channels, the probability reduces to 0.79. If an interfering path is found, the algorithm is recomputed after temporarily discarding one of the offending connectivity edges. While this increases the probability of finding a non-interfering path (as the offending edge is removed), for our calculations, we maintain the same (lower) probability of additional paths being intra-flow interference free. With this conservative approach, we estimate the number of times Dijkstra's must be recomputed to have a 99% probability of finding a intra-flow interference free route. The results are shown on the secondary y-axis (on the right side) of the aforementioned figure. While the practical upper bound on the number of recomputations is less than half a dozen, we take a pessimistic approach and use a much higher limit of 100 for our implementation.

5.3 Algorithm complexity

Given a graph $G = (V, E)$, Dijkstra's algorithm implemented using Fibonacci heaps completes in $\mathcal{O}(|E| + |V| \log |V|)$ [7]. We can compute the computational complexity of the routing algorithm by counting the total number of edges and nodes in the abstract graph.

Assuming $T = 2k - 1$, the abstract graph has $2k$ (i.e., $T + 1$) virtual nodes for every physical node, for a total of $2|V|k$ or $\mathcal{O}(|V|k)$ nodes. There are $2k - 1$ connectivity edges for each intra-subnetwork link, whereas there is exactly 1 connectivity edge for all inter-subnetwork links. With random subnetwork assignment, the probability of such an intra-subnetwork link is $P_{\text{intra-subnet}} = \frac{1}{2k}$. Hence, the expectation for the total number of connectivity edges, $|E_c|$, in the abstract graph is:

$$\begin{aligned} Ex[|E_c|] &= P_{\text{intra-subnet}} \cdot |E| \cdot (2k-1) + (1 - P_{\text{intra-subnet}}) \cdot |E| \cdot 1 \\ &= \frac{1}{2k} \cdot |E| \cdot (2k-1) + \left(1 - \frac{1}{2k}\right) \cdot |E| \cdot 1 \\ &= \frac{|E| \cdot (2k-1)}{k} \\ &< 2|E| \end{aligned}$$

Further, there are $2k - 1$ temporal edges for each physical node for a total of $|V|(2k - 1)$ temporal edges. Finally, the virtual base nodes of the physical source and destination nodes are connected to each of their $2k - 1$ respective temporal nodes. Thus, there are a total of $2|E| + |V|(2k - 1) + 2(2k - 1)$, or $\mathcal{O}(|E| + |V|k)$ edges. As a result, the cost of a single iteration of Dijkstra's algorithm on the abstract graph is $\mathcal{O}(|E| + |V|k \log(|V|k))$. This step is executed up to 100 times to locate each intra-flow interference-free path. The process is repeated until all paths are found. At best, there are $2k - 1$ non-interfering paths between two nodes, i.e., one per slot. Therefore, the complexity of locating multiple routes is $\mathcal{O}(|E|k + |V|k^2 \log(|V| \cdot k))$.

6 Experiments

We evaluate Dominion using the Qualnet v3.9 network simulator [24]. The first set of experiments comprehensively evaluate the various design parameters of Dominion. Next, the comparative experiments evaluate Dominion against SSCH [1] and 802.11 under various scenarios. We find that Dominion outperforms both 802.11 (by an order of magnitude) and SSCH in aggregate throughput and fairness.

6.1 Implementation

We implement Dominion using the Qualnet network simulator v3.9 [24]. Most of the code resides at the network layer with a few hooks in to the MAC layer. To promptly switch channels when they are due, the MAC layer attempts only 1 DCF (vs. 7 for 802.11) transmission at a time for each packet. A packet is dropped after 14 successive DCF failures, akin to two 802.11 retries. Dominion uses source routing akin to DSR [15]. Each packet includes a Dominion header specifying the hop count, the sequence of intermediate nodes, and the channel and time at which the intermediate nodes must transmit. As an example, a packet traversing 5 hops would incur an overhead of 35 bytes due to source routing.

A node maintains a packet buffer equivalent to the maximum number of the packets that can be transmitted in a single timeslot. FIFO queuing of buffered packets does not bode well for intermediate nodes. A packet arriving at an intermediate node maybe not be scheduled to depart until a later timeslot. To prevent head-of-line blocking, separate per-timeslot FIFO queues are maintained.

However, per-timeslot FIFO queues still present a problem of flow starvation. Imagine a scenario where both node A and node B have an ongoing flow to destination node D via intermediate node C. Node A converges with node C at t_1 and node B converges with node C at t_2. Lastly, node C converges with node D at t_3. The flow with an earlier convergence to node C (i.e., node A at timeslot t_1) will fill up node C's buffer for timeslot t_3, starving the flow from node B. Hence, we maintain per-timeslot, per-flow FIFO queues. During a given timeslot, packets from the various per-flow FIFO queues are transmitted using a round-robin scheduler.

Yet a small problem remains—at node C, packets for timeslot t_3 are arriving twice as fast (incoming at both t_1 and t_2) as they can be forwarded. In such a scenario, packets may be dropped at intermediate nodes. To prevent buffer overflows, we use flow control to block the arrival of new packets. Once a per-flow queue contains as many packets as can be transmitted within a single timeslot, the flow of packets is blocked until the next schedule cycle. As the 2-byte time stamp field is generally unused by 802.11 ACK packets, we use it to deliver the "buffer is now full" directive to the previous hop neighbor.

Lastly, we acquired the code for SSCH from its authors. The original SSCH code was implemented in Qualnet v3.6, which we ported to v3.9. However, the current implementation of SSCH has an intrinsic problem—it requires the number of channels to be prime. Thus to make a fair comparison, we use 11 as the number of channels for all comparative experiments.

6.2 Methodology

The physical layer uses the two-ray path-loss model and the constant shadowing model. As a result, the physical links exhibited temporal variance. The `TX-POWER` and `RX-SENSITIVITY` of transceivers use the specifications of the popular Wistron NeWeb DCMA-82 802.11a/g adapter [28]. The transceiver can transmit at 22.5 dBm using the maximum 802.11a MAC bit rate (54 mbits/s). The nodes also used a low-powered omnidirectional antenna (5 dBi at 80% efficiency). Both virtual carrier sensing (i.e., no RTS-CTS packets) and autorate fallback (i.e., nodes always transmit at 54 mbits/s) were disabled. All 12 802.11a channels were used. The cost of a switching to another channel is 80 μs [11].

While Dominion was evaluated using various parameters, we establish a set of default settings. These settings are validated in Sect. 6.3. We use these default settings for all our experiments unless mentioned otherwise. The default topology uses 100 nodes placed in a 1 km^2 square, with the node locations chosen using uniform random placement. The maximum shortest distance between any two nodes in the default topology (with the configured transceiver settings) is 7 hops. The timeslot duration for both Dominion and SSCH is set to 10 ms. Nodes used the HT routing substrate.

A dynamic routing protocol induces substantial overhead for route discovery and maintenance (or recovery) inhibiting maximum attainable throughput [3]. As a result, we use static routing to eliminate the routing overhead.[4] Prior to the experimentation, we execute a bootstrap process to gauge the "steady-state" quality of each link.

During the bootstrap process, the network is converged to a single channel. Each node broadcasts a few HELLO messages at random intervals. At the completion of the bootstrap process, the quality of each link is calculated based on the proportion of HELLO messages received by each neighbor. Only the set of high quality links (i.e., forward transmission probability of 85% or higher) is used to determine routes for Dominion. This is done to reduce the memory and computational requirements for calculating routes, and is a strategy we envision real Dominion deployments to use. On the other hand, for our experiments, we calculate static ETX [8] routes for both 802.11 and SSCH using all the links. Note that ETX only selects the best available links unless a "worse link" provides a shorter path—in our experiments, more than 96% of links selected by ETX had a forward transmission probability of 85% or greater.

Lastly, based on the link state information, each Dominion node is assigned to a subnetwork to minimize the number of nodes in each subnetwork within their local 2-hop neighborhood. Subnetworks can also be assigned dynamically if each node piggybacks information about its neighbors atop periodic HELLO messages [16].

We believe that in a real deployment, the bootstrap process can be repeated at low frequency (e.g., every few schedule cycles) to proactively maintain the link state and perform dynamic subnetwork assignment [16]. Note that only differences in the link state would need to be propagated. Section 8.1 discusses another approach to maintaining the link state.

The experiments were performed using constant bit rate (CBR) flows over UDP. A 1024-byte packet was offered

[4] We partially evaluate 802.11 with DSR to verify this claim.

every 100 μs safely saturating the maximum MAC bit rate. Each individual simulation lasted 30 s, with network flow(s) starting at the 15 s mark and terminating at the end. To reduce statistical anomalies, each throughput-based benchmark was evaluated 5 times with a different set of flows. All results are described using the mean of the experimental runs.

6.3 Dominion characteristics

In the first two experiments, we study the effect of route length. For these experiments, we execute only one flow per trial. Figure 5 shows that the goodput of a network using 802.11 reduces inversely with increasing hop count. Note that the maximum goodput is not equal to the native MAC bit rate (54 mbits/s) primarily due to packet acknowledgment overhead. On the other hand, Dominion (with 12 channels) using only a single route maintains a constant goodput approximately equivalent to $\frac{1}{23}$th the maximum goodput irrespective of the hop count. This matches the theoretical observation made in Sect. 4.2. Using multiple routes, Dominion achieves higher throughput by locating multiple non-interfering routes. Note that 802.11 outperforms Dominion for shorter routes, but Dominion excels with longer routes. Throughput of Dominion flows depends on the number of non-interfering routes rather than on the distance between nodes.

As shown before in Sect. 5.1, the end-to-end latencies in Dominion depend on the route length. To effectively evaluate the end-to-end latencies, we randomly select 50 different routes of each possible path length. As mentioned previously, the longest route in our default topology is 7 hops. However, due to the scarcity of long routes, the evaluation of end-to-end latencies was limited routes 6 hops long. To precisely measure the end-to-end latency, all network flows transmit only one data packet. Further, these flows are time-staggered to not cause interference with each other. We use a timeslot of 2 ms for this experiment,

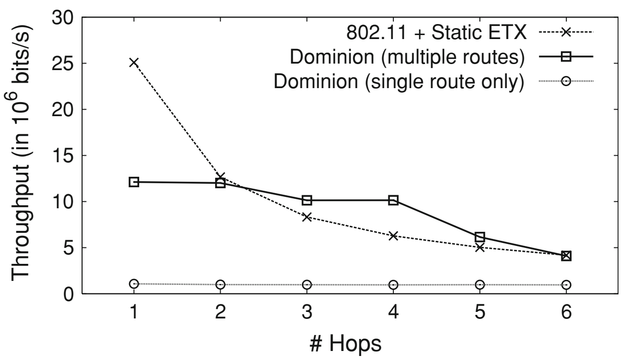

Fig. 5 Route length vs. throughput

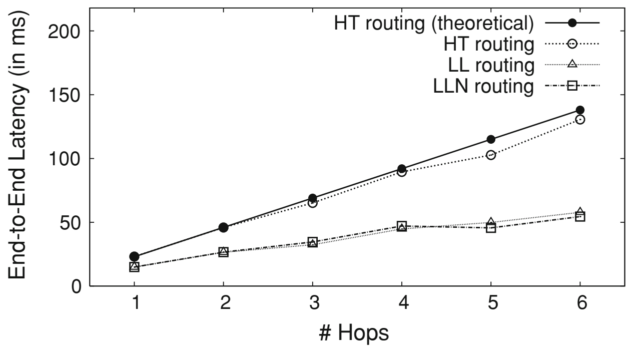

Fig. 6 Route length vs. latency

however, results can be multiplied by a linear factor to approximate results for other durations.

Figure 6 shows the mean end-to-end latency experienced by HT, LL and LLN routing substrates. Notice that the end-to-end latencies for LLN routing are under 50 ms for routes as long as 6 hops. We believe this is an acceptable delay for web traffic loads. For HT routing, the experimental end-to-end latencies closely match the theoretical results derived in Sect. 5.1. The end-to-end latencies for HT routing are approximately 3 times worse than LL and LLN routing.

For the next set of experiments, we use 50 random flows of varying lengths, to evaluate aggregate network throughput. To mitigate the effect of varying route lengths, we present distance-normalized results, i.e., each flow's throughput is given a weight equivalent to the distance between its end nodes. The reasons for distance-normalization are further clarified in the next section (Sect. 6.4), however, for consistency, we use distance-normalization for the rest of the paper.

Figure 7 shows the performance of Dominion if the maximum number of subflows is 3, 6, and unlimited (i.e., 23 with 12 channels). Unlimited subflows provides the best result when there are fewer simultaneous flows, but the 6 subflows and later the 3 subflows outperform unlimited subflows as the number of simultaneous flows increase. This is likely because a as increase in active subflows leads to greater contention for the medium.

Automatically adjusting the number of subflows utilized at run time remains an interesting idea for future extensions.

Next, we evaluate two of the subnetwork assignment schemes discussed in Sect. 3.3. Figure 8 shows that assigning nodes to subnetwork based on a 2-hop geographic neighborhood [16] provides a noticeable improvement over random subnetwork assignment.

While we use the default timeslot duration of 10 ms, a smaller timeslot duration should theoretically only affect the throughput of the network marginally. With a 10 ms

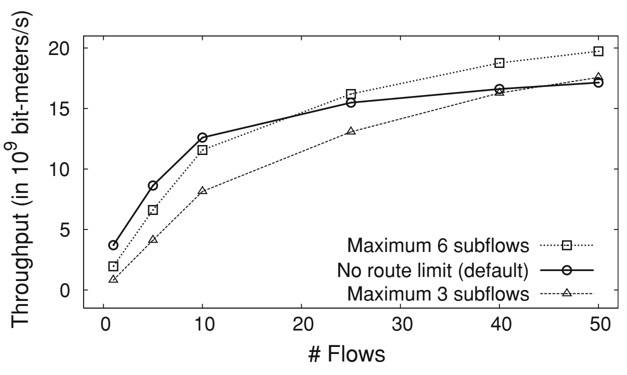

Fig. 7 Parameter: number of subflows

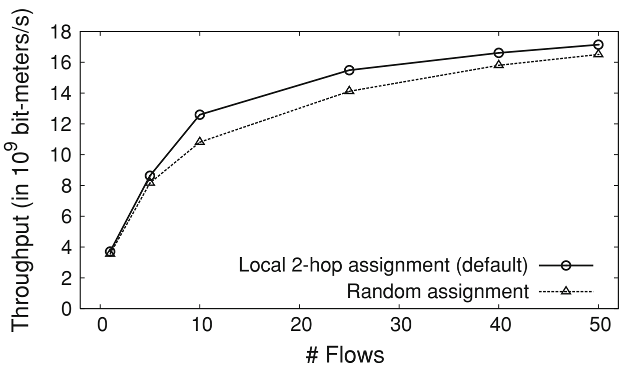

Fig. 8 Parameter: subnetwork assignment

timeslot duration, a 80 µs switching delay implies a relatively negligible 0.8% penalty. With a smaller timeslot duration, the switching penalty increases linearly. However, Fig. 9 shows that the overhead does not remain linear with decreasing timeslot duration as theoretically expected. This is because the transceiver cannot switch to the designated channel until the packet in flight is delivered to (or received from) a neighbor. As mentioned earlier, Dominion minimize this delay by only attempting 1 DCF transmission per attempt. While a 2 ms timeslot incurs noticeable degradation, intermediate timeslots of 3, 4, and 5 ms provide interesting trade-off points between high throughput and reduced end-to-end latencies.

Finally, Fig. 10 shows the effect of maintaining link state of varying link qualities. Link quality has minimal effect with HT routing, as ETX based routing naturally selects high quality links However, the performance of LL routing varies significantly as it is oblivious to link quality. The best LL route attempts to minimize end-to-end latency, and may select poor quality links when low quality links are used. It should be noted that this plot also shows the trade-off between HT and LL routing in terms of throughput. HT routing provides significantly better throughput than LL routing. However, the question of an appropriate minimal link quality cannot be answered without evaluating the end-to-end latencies. Figure 11 shows that LL routing with higher quality links (75, 85, and 95%) performs noticeably better than with lower quality links (50%). This may seem surprising as a shorter temporal route can be chosen with the numerous additional links provided by a lower quality link state. However due to the increased packet drop rates, and hence packet retransmissions, lower quality links are ineffective. We settle on a links with 85% or greater probability of successful transmission as effective links.

6.4 Comparative evaluation

Figure 12 shows that both Dominion and SSCH achieve an order of magnitude higher aggregate throughput than 802.11 with random flows (note that the result is not distance-normalized here). The throughput increases substantially as the numbers of flows increase. It should be noted that 802.11 is included only as a strawman as it operates on a single channel only (observe that a more "sophisticated" 802.11 is not possible as using multiple channels partitions the network).

SSCH does achieve higher actual throughput than Dominion for a small number ($\leq$25) of simultaneous flows.

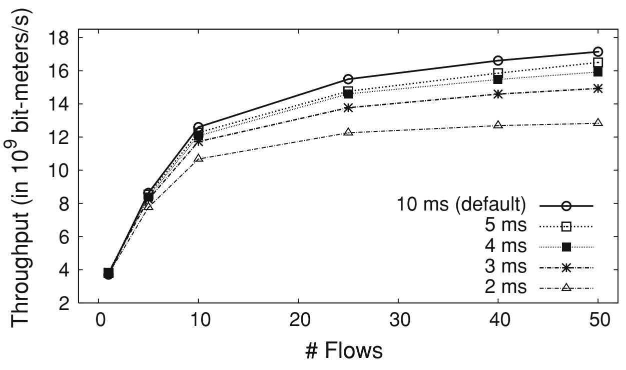

Fig. 9 Parameter: timeslot duration

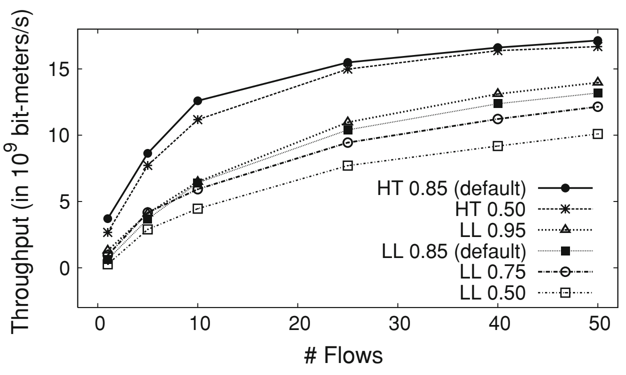

Fig. 10 Parameter: link quality (vs. throughput)

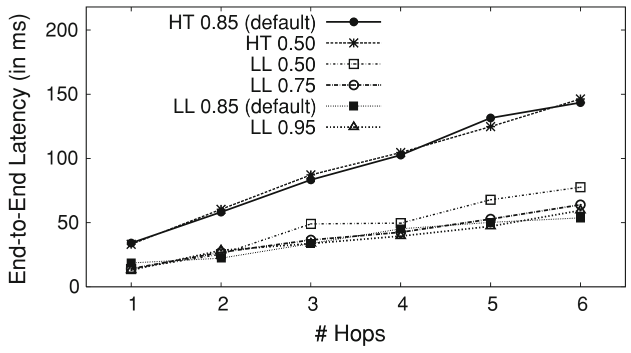

Fig. 11 Parameter: link quality (vs. latency)

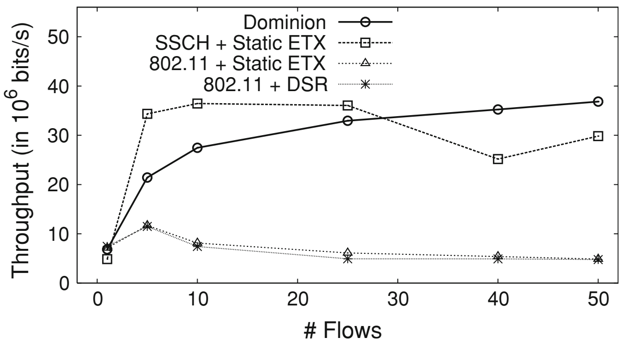

Fig. 12 Actual aggregate throughput

However, examining the same results using distance-normalization (i.e., each flow's throughput is given a weight equivalent to the distance between its end-nodes) and Jain's fairness index exposes a different story. The next two plots show that SSCH achieves good aggregate throughput by favoring a handful of shorter flows at the expense of longer flows.

Firstly, using distance-normalization, Fig. 13 shows that both Dominion and SSCH perform similarly with fewer flows (<15) but Dominion outperforms SSCH with increasing concurrent flows. With 50 simultaneous random flows, on average, Dominion achieves 1064 and 93% higher distance-normalized aggregate throughput than 802.11 and SSCH respectively.

Secondly, Jain's fairness index [14] is used to evaluate throughput distribution amongst individual flows. In Jain's Fairness Index, a 1.0 represents optimal fairness. Figure 14 shows that both SSCH and 802.11 fare poorly compared to Dominion. With 50 simultaneous random flows, Dominion is 299 and 291% fairer than both 802.11 and SSCH respectively.

Lastly, for the same experiment, we show that shows that Dominion performance is more consistent than SSCH. Figure 15 plots the actual results from five different runs. Recall that the flows selected differ for each run. The high variance in the performance of SSCH is due to the mechanism in which nodes synchronize with their neighbors: when given multiple options, a node chooses to synchronize with the neighbor for which it has the most packets. As a result, a source node almost continually synchronizes with its next hop neighbor, reducing the throughput for any flow for which it is an intermediate node.

Next we evaluate the effect of varying density on Dominion and SSCH. For this experiment, we generate two new topologies—one with a total area of 0.5 km^2 and another with a total area of 2 km^2. The number of nodes remains the same at 100. As a result, the maximum shortest distance between any two nodes is 4 hops for the smaller topology and only 10 hops for the larger topology. Figure 16 shows that Dominion achieves higher aggregate throughput than SSCH under both topologies. Figure 17 is another experiment that compares the effect of using a different number of channels for Dominion and SSCH. Dominion outperforms SSCH with both 7 channels and 3 channels. Quite surprisingly, Dominion with 3 channels matches the throughput of SSCH with 7 channels when the network experiences more than 40 simultaneous flows.

Generally, most flows in wireless mesh networks either originate, pass-through, or terminate at centrally located gateway nodes. Using only a single gateway node, we evaluate the performance of Dominion (and others) with up to 30 simultaneous bi-directional gateway flows. Figure 18 shows that with 30 simultaneous gateway flows, Dominion

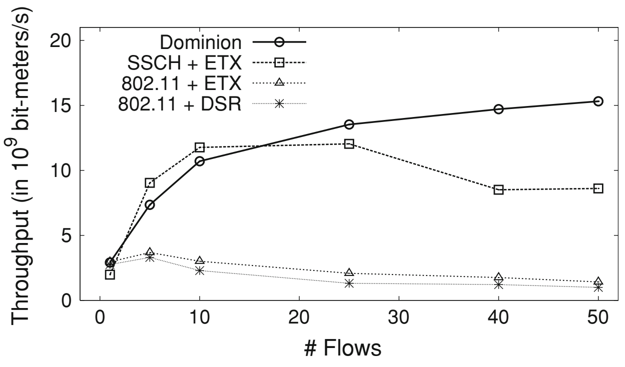

Fig. 13 Normalized aggregate throughput

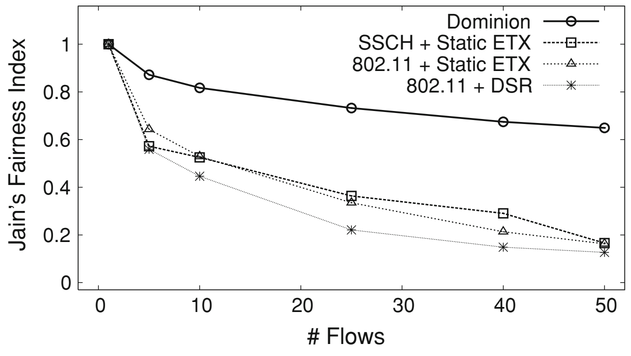

Fig. 14 Jain's fairness index

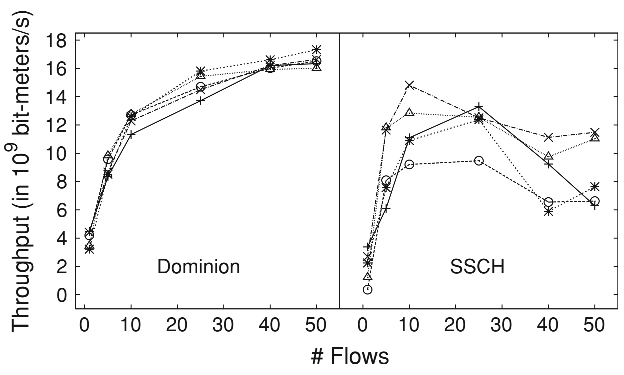

Fig. 15 Variation in flow selection: individual runs

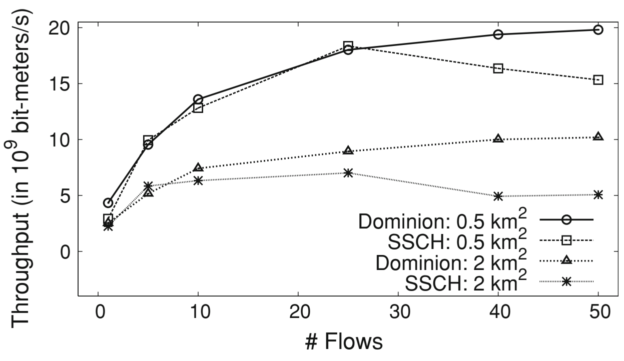

Fig. 16 Parameter: network density

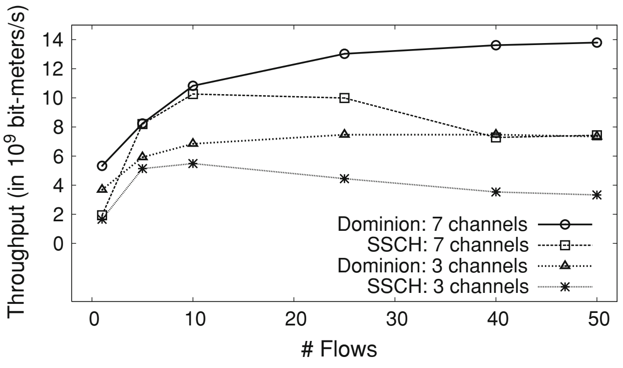

Fig. 17 Parameter: channels utilized

achieves 55 and 64% higher aggregate throughput than 802.11 and SSCH respectively. To further improve performance vs. 802.11, Dominion and SSCH may utilize multiple transceivers per node to further improve performance. Because neither support it intrinsically, we use multiple co-located nodes to mimic a gateway node with multiple transceivers. Note that *only* the gateway node has multiple transceivers. Dominion$_{3T}$ (i.e., subscript indicates number of transceivers) performs remarkably better than the other solutions. In fact, Dominion with a single transceiver achieves higher throughput than SSCH$_{3T}$ after 15 flows.

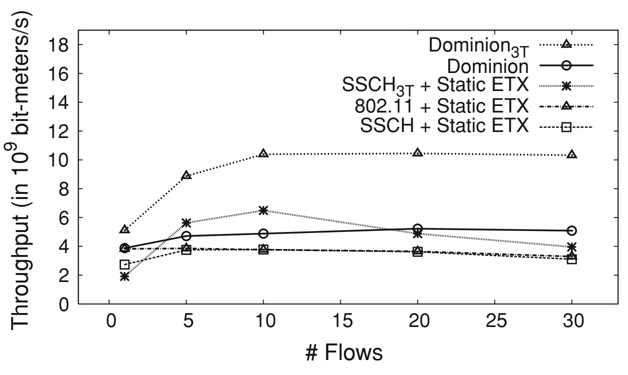

Fig. 18 Bi-directional gateway flows

7 Related work

Previous research efforts can be divided into two broad categories: (i) solutions that exploit multi-channel diversity; and (ii) solutions that exploit multi-user diversity with multi-path routing.

Solutions that exploit multi-channel diversity with only a single commodity transceiver are MMAC [16], SSCH [1] and Local Coordination-based MAC (LCM MAC) [18]. MMAC and SSCH were described in Sect. 2. Local LCM MAC performs coordinated channel negotiations and channel switching to provide multichannel support. In essence, it improves MMAC while following similar design principles. LCM MAC eliminates the need for strict time synchronization by requiring nodes in designated nodes to converge to a common control channel when idle.

The next set of solutions utilize multiple commodity transceivers. Dynamic Channel Assignment (DCA) [29] requires two transceivers per node, one for the control channel communication and the other for actual transmission of data packets. Jain et al. [13] propose a similar scheme to DCA except the receiver decides which channel to switch to next. Draves et al. [9] propose a multi-radio routing protocol that uses two transceivers: one operating on 802.11a and the other on 802.11b/g. Hyacinth [25] requires two transceivers per node, both of which are assigned different static channels such that the network remains connected. Kyasanur and Vaidya [16] propose a routing protocol that works with two transceivers: the first transceiver is assigned a static home channel, whereas the second transceiver channel hops to transmit packets.

Another set of solutions utilize (at least) one specialized transceiver. Nasipuri et al. [21, 22] propose two closely related multi-channel CSMA protocols that assume a transceiver can listen to all available channels simultaneously. Node pairs choose the least busy channel to exchange data packets. Extended Receiver Directed Transmission Protocol (xRDT) [18] is a recently proposed protocol that uses an additional “busy tone” interface (coupled with a commodity transceiver) to resolve the multichannel hidden terminal and deafness problem.

So and Vaidya [27] propose a routing-later solution to exploit multiple channels using only a single commodity transceiver. The proposed protocol assigns specific channels to routes. Whenever a node initiates route discovery, a certain channel is assigned to route (i.e., all nodes along the route are switched to that channel). The scheme fails to exploit maximal multi-frequency diversity when the network is experiencing multiple non-disjoint flows as routes start converging to a dominant channel.

Existing solutions that exploit multi-path routing include Opportunistic Multi-path Scheduling (OMS) [5], which uses multiples routes via intelligent scheduling. Extremely Opportunistic Routing (ExOR) [2] forwards a series of packets through nodes, deferring the choice of next hop until after the previous node has transmitted the packet. pTCP [12] and R-MTP [17] are two closely related solutions that improve TCP performance over multi-path routing.

8 Conclusion

In this paper, we make four new contributions. Firstly, in Sect. 4, we present the graph-theoretic model for the multichannel wireless mesh network that supports goal-oriented routing. Nodes can locally choose to maximize throughput or minimize end-to-end latency without requiring any changes in the network. Secondly, in the same section, we describe a technique to remove intra-flow interference. Section 6.3 validates our theoretical observation that in absence of extrinsic interference, network flows can maintain constant throughput irrespective of distance. Next, in Sect. 5 via theoretical modeling and analysis, we provide expected end-to-end latencies for network flows. Finally, in Sect. 6.4, the extensive QualNet simulations show that Dominion is able to achieve 93% higher throughput and improve fairness by 291% over SSCH.

8.1 Future work and discussion

A big drawback that can hamper adoption of Dominion is its TCP performance. In preliminary experimentation, we find that most of TCP problems arise due to multi-path routing. As subflows can terminate at arbitrary time slots, the packet inter-arrival rate is highly variant (i.e., jitter). For example, numerous packets can arrive in a short burst during a time slot in which a subflow terminates, followed by a period of no new packet arrivals until the next time slot in which another subflow terminates. To mitigate this problem, we believe we can leverage previous work that aggregate packets over multiple paths via a “middleware” packet re-sequencer [12, 17].

To support routing primitives, we would also like to investigate the ability to efficiently broadcast messages. For example, SSCH [1] takes a probabilistic approach for broadcasting: each node broadcasts a message for only a part of its schedule cycle. An alternate method is to add one or more time slots (per schedule cycle) for network-wide convergence, i.e., forming a single channel 802.11 network. This is likely to reduce the capacity of the network slightly, but on the other hand, make primitives such as broadcast and clock synchronization easier to implement. An added bonus is that such network-wide convergence allows for easier maintenance and dissemination of the network link state.

Another optimization is the run-time adjustment of the number of channels being utilized. While a higher number of channels allow for greater network capacity, fewer number of channels may provide two key benefits. Firstly, fewer channels leads to reduction of the schedule cycle, and thus lower end-to-end latency. Secondly, a single subflow achieves higher throughput as the number of channels being utilized decreases. This is because the channel schedule is repeated more frequently and a node can transmit packets for a greater proportion of time. Given this, networks with low density and low parallelism (of flows) may achieve higher throughput with fewer channels (depending on number of subflows found in relation to the length of the schedule cycle). An alternate mechanism that can provide higher throughput and lower end-to-end latency simultaneously is via the usage of multiple transceivers per node—with each radio assigned distinct subnetworks.

Acknowledgments We are thankful to Ranveer Chandra for providing the code for the original SSCH implementation. This research was supported in part by NSF CAREER grant CNS-0448246, and in part by NSF ITR grant CMS-0427089.

References

1. Bahl, P., Chandra, R., & Dunagan, J. (2004). SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of ACM MobiCom* (pp. 216–230), Philadelphia, PA, USA, September 2004.
2. Biswas, S., & Morris, R. (2005). ExOR: Opportunistic routing in multi-hop wireless networks. In *Proceedings of ACM SIGCOMM* (pp. 133–144), Philadelphia, PA, USA, August 2005.
3. Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., & Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of ACM MobiCom* (pp. 85–97), Dallas, TX, USA, October 1998.
4. Camp, J. D., Knightly, E. W., & Reed, W. S. (2005). Developing and deploying multihop wireless networks for low-income communities. In *Proceedings of Digital Communities*, Napoli, Italy, June 2005.

5. Cetinkaya, C., & Knightly E. (2004). Opportunistic traffic scheduling over multiple network paths. In *Proceedings of IEEE INFOCOM* (pp. 1928–1937), Hong Kong, March 2004.
6. Champaign-Urbana wireless network. Website: http://www.cuwireless.net/.
7. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Single-source shortest paths, chapter 24* (2nd ed., pp. 580–619). MIT Press.
8. De Couto, D. S. J., Aguayo, D., Bicket, J., & Morris, R. (2003). A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MobiCom* (pp. 134–146), San Diego, CA, USA, September 2003.
9. Draves, R., Padhye, J., & Zill, B. (2004). Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of ACM MobiCom* (pp. 114–128), Philadelphia, PA, USA, September 2004.
10. Elson, J., Girod, L., Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (pp. 147–163), Boston, MA, USA, December 2002.
11. Herzel, F., Fischer, G., & Gustat, H. (2003). An integrated CMOS RF synthesizer for 802.11a wireless LAN. *IEEE Journal on Solid-State Circuits, 38*(10), 1767–1770.
12. Hsieh, H.-Y., & R. Sivakumar. (2002). pTCP: An end-to-end transport layer protocol for striped connections. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)* (pp. 24–33), Paris, France, November 2002.
13. Jain, N., Das, S. R., & Nasipuri, A. (2001). A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks. In *Proceedings of IEEE International Conference on Computer Communications and Networks (ICNP)* (pp. 432–439), Scottsdale, AZ, USA.
14. Jain, R. K., Chiu, D.-M. W., & Hawe, W. R. (1984). *A quantitative measure of fairness and discrimination for resource allocation in shared computer systems*. Technical Report TR-301. Digital Equipment Corporation.
15. Johnson, D. B., Maltz, D. A., & Broch, J. (2001). *DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks, chapter 5* (pp. 139–172). Addison-Wesley.
16. Kyasanur, P. & Vaidya, N. H. (2005). Routing and interface assignment in multi-channel multi-interface wireless networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 2051–2056), New Orleans, LA, USA.
17. Magalhaes, L., & Kravets, R. (2001). Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)* (pp. 165–171), Riverside, CA, USA, November 2001.
18. Maheshwari, R., Gupta, H., & Das S. R. (2006). Multichannel MAC protocols for wireless networks. In *Proceedings of IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)* (pp. 393–401), Reston, Virginia, USA, September 2006.
19. Mishra, A., Banerjee, S., & Arbaugh, W. (2005). Weighted coloring based channel assignment for wlans. *ACM SIGMOBILE Mobile Computing and Communications Review, 9*(3), 19–31.
20. MIT Roofnet. Website: http://pdos.csail.mit.edu/roofnet/.
21. Nasipuri, A., & Das, S. R. (2000). Multichannel CSMA with signal power-based channel selection for multihop wireless networks. In *Proceedings of IEEE Vehicular Technology Conference (VTC)* (pp. 211–218), Boston, MA, USA, September 2000.
22. Nasipuri, A., Zhuang, J., & Das, S. R. (1999). A multichannel CSMA MAC protocol for multihop wireless networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1402–1406), New Orleans, LA, USA, September 1999.
23. Patel, J. A., Luo, H., & Gupta, I. (2007). A cross-layer architecture to exploit multi-channel diversity with a single transceiver. In *Proceedings of IEEE INFOCOM (Minisymposium)*, Anchorage, AK, USA, May 2007.
24. Qualnet user manual v3.9. Website: http://www.qualnet.com/.
25. Raniwala, A., & Chiueh, T. (2005). Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proceedings of IEEE INFOCOM* (pp. 2223–2234), Miami, FL, USA, March 2005.
26. So, J., & Vaidya, N. H. (2004). Multi-channel MAC for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of ACM MobiHoc* (pp. 222–233), Tokyo, Japan, May 2004.
27. So, J., & Vaidya, N. H. (2004). *A routing protocol for utilizing multiple channels in multi-hop wireless networks with a single transceiver*. Technical report. University of Illinois at Urbana-Champaign.
28. Wistron NeWeb DCMA-82 mini-PCI adapter. Website: http://www.wneweb.com/wireless/wireless_mini-pci.htm.
29. Wu, S.-L., Lin, C.-Y., Tseng, Y.-C., & Sheu, J.-P. (2000). A new multi-channel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *Proceedings of IEEE International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)* (pp. 232–237). Dallas, TX, USA, December 2000.

Author Biographies

Jay A. Patel is a Ph.D. student at the University of Illinois at Urbana-Champaign (UIUC). He previously received a B.S.C.S. from The University of Texas-Pan American (UTPA). His primary research interests include designing, implementing, and evaluating new and interesting distributed protocols, with secondary interests in areas of wireless networks. He is a student member of both the ACM and the IEEE. On separate occasions, he has served as both the Chair and Treasurer of the UTPA student chapter of the ACM.

Haiyun Luo completed his Ph.D. in Computer Science from the University of California at Los Angeles. His research interests include the design, imple-mentation, and evaluation of wireless network architectures, protocols, and applications; specifically, wireless Internet access, intelligent wireless protocols, self-organized wireless systems, scalable wireless mesh, and future embedded networks. Earlier in his career, he served as an Assistant Professor of Computer Science at the University of Illinois at Urbana-Champaign.

Indranil Gupta completed his Ph.D. in Computer Science from Cornell University in 2004. He presently leads the Distributed Protocols Research Group in the CS Department at UIUC. He is interested in research on distributed protocols, large-scale distributed systems, monitoring and management for distributed systems, and sensor networks. He is recipient of the NSF CAREER award in 2005.