Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network

Ashish Raniwala Tzi-cker Chiueh
Computer Science Department, Stony Brook University,
Stony Brook, NY 11794-4400
Email: {raniwala,chiueh}@cs.sunysb.edu

Abstract— **Even though multiple non-overlapped channels exist in the 2.4GHz and 5GHz spectrum, most IEEE 802.11-based multi-hop ad hoc networks today use only a single channel. As a result, these networks rarely can fully exploit the aggregate bandwidth available in the radio spectrum provisioned by the standards. This prevents them from being used as an ISP's wireless last-mile access network or as a wireless enterprise backbone network. In this paper, we propose a multi-channel wireless mesh network (WMN) architecture (called *Hyacinth*) that equips each mesh network node with multiple 802.11 network interface cards (NICs). The central design issues of this multi-channel WMN architecture are channel assignment and routing. We show that intelligent channel assignment is critical to *Hyacinth*'s performance, present *distributed* algorithms that utilize only local traffic load information to dynamically assign channels and to route packets, and compare their performance against a centralized algorithm that performs the same functions. Through an extensive simulation study, we show that even with just 2 NICs on each node, it is possible to improve the network throughput by a factor of 6 to 7 when compared with the conventional single-channel ad hoc network architecture. We also describe and evaluate a 9-node *Hyacinth* prototype that is built using commodity PCs each equipped with two 802.11a NICs.**

Keywords: System design, Distributed Algorithms/Protocols, Simulations, Prototype Implementation.

I. Introduction

Despite significant advances in physical layer technologies, today's wireless LAN still cannot offer the same level of sustained bandwidth as their wired brethren. The advertised 54 Mbps bandwidth for IEEE 802.11a/g wireless LAN interface is the peak *link-layer* data rate. When all the overheads – MAC contention, 802.11 headers, 802.11 ACK, packet errors – are accounted for, the actual goodput available to applications is almost halved. In addition, the maximum link-layer data rate falls quickly with increasing distance between the transmitter and the receiver. The bandwidth problem is further aggravated for multi-hop ad hoc networks due to interference from adjacent hops on the same path as well as from neighboring paths [1] [28]. Fortunately, the IEEE 802.11b/g standards and IEEE 802.11a standard provide 3 and 12 non-overlapped frequency channels, respectively, which could be used simultaneously within a neighborhood. Ability to utilize multiple channels substantially increases the effective bandwidth available to wireless network nodes. Such bandwidth aggregation is routinely used in 802.11-based wireless LANs that operate in infrastructure mode, where traffic to and from wireless nodes is distributed among multiple interfaces of an access point or among multiple access points. However, bandwidth aggregation is rarely applied to 802.11-based LANs that operate in the ad hoc mode. As a result, most ad hoc network implementations use only a single frequency channel, wasting the rest of the spectrum.

In this paper, we describe an IEEE 802.11-based multi-hop wireless ad hoc network architecture (called *Hyacinth*) that employs multiple radio channels simultaneously by equipping each node with multiple NICs. We detail the associated channel assignment and routing algorithms, and present the results of a comprehensive performance study of these algorithms. Although there have been previous research efforts that aimed to exploit multiple radio channels in an ad hoc network, most of them were based on proprietary MAC protocols [2]–[5], and therefore cannot be directly applied to wireless networks using commodity 802.11 interfaces. In contrast, *Hyacinth* is designed to work directly with commodity 802.11-based interfaces, and *requires only systems software modification*. Although our prototype uses 802.11 interfaces, the architecture is also applicable to the *802.16a* networks, where customer premise equipments form a mesh connectivity to reach the base station.

The focus of this paper is on wireless mesh networks (WMNs). A WMN operates just like a network of fixed routers, except that they are connected only by wireless links. WMNs are gaining significant momentum as an inexpensive way to provide *last-mile broadband Internet access* [14]–[18], [30]. In this application, some of the nodes in the WMN are connected to the Internet via physical wires, while the remaining nodes access the Internet through these wired gateways by forming a multi-hop WMN with them. As deployment and maintenance of physical wires is a major cost component in providing high-speed Internet access [17], use of WMN at the last hop significantly brings down the overall system cost and offers an attractive alternative to DSL/cable modem.

Another application of WMN is an *enterprise-scale wireless backbone*, where access points inter-connect using wireless links to form a connectivity mesh [19], [20], [31], [32]. Most of today's enterprise wireless LAN deployment is only limited to the *access network* role, where a comprehensive *wired* backbone network is still needed to relay traffic from/to wireless LAN access points. Use of WMN can effectively eliminate the wired backbone and enable truly wireless enterprises.

Although a WMN is similar in concept to a mobile ad hoc network, there are some important differences between the two. Firstly, nodes in a WMN are fixed (i.e. not mobile). Topology changes are therefore infrequent, and occur only due to occasional node failures, node shut-down for maintenance, or addition of new nodes. Secondly, the traffic characteristics, being aggregated from a large number of traffic flows, do not change very frequently, permitting optimization of network based on measured traffic profiles. Thirdly, the traffic

distribution in a WMN is typically skewed, as most of the user traffic is directed to/from a wired network. This happens because users typically want to access resources on the Internet or on the enterprise servers, and both of them most likely reside on the wired infrastructure [22]. Finally, to serve as an effective backbone, a WMN requires *proactive* discovery of paths to reduce packet delays. In contrast, in most mobile ad hoc networks, reactive routing strategies are a normal as additional packet latency due to on-demand route discovery is acceptable.

This paper describes and evaluates a novel multi-channel WMN architecture that is built on 802.11-based wireless LAN hardware and is specifically tailored to multi-hop wireless access network applications. Although the multi-NIC approach has been mentioned in the past, we believe this work represents the first comprehensive study of this approach in the context of a wireless access network. In particular, this paper makes the following research contributions:

- A fully distributed channel assignment algorithm that can adapt to traffic loads dynamically,
- A multiple spanning tree-based load-balancing routing algorithm that can adapt to traffic load changes as well as network failures automatically, and
- A comprehensive performance study that shows significant bandwidth improvements over single-channel WMNs, which are validated through empirical measurements on a fully working prototype.

The rest of the paper is organized as follows. Section II reviews the past work related to this project. Section III discusses the overall system architecture of *Hyacinth*, brings out the research issues in the architecture, and outlines a previously proposed centralized channel assignment algorithm. Section IV presents the distributed channel assignment and routing algorithms for the proposed architecture. Section V evaluates the *Hyacinth* architecture and the effectiveness of the proposed algorithms. Section VI briefly discusses the implementation and empirical evaluation of the 9-node *Hyacinth* prototype. Finally, section VII concludes the paper with a summary of research contributions.

II. Related Work

A. Multi-channel MAC

Several proposals [2]–[5] have been made to modify the MAC layer to support multi-channel networks. The approach taken by most of this body of research is to find an optimal channel for a single packet transmission, essentially avoiding interference and enabling multiple parallel transmissions in a neighborhood. Unlike in all these previous proposals, our architecture does not perform channel switching on a packet-by-packet basis; our channel assignment lasts for a longer duration, such as several minutes or hours, and hence does not require re-synchronization of communicating network cards on a different channel for every packet. This property makes it feasible to implement our architecture using commodity 802.11 hardware.

B. Multi-radio research

The multi-NIC approach has been discussed in some past work [21] [22], however no distributed channel assignment algorithm has been proposed that can indeed realize the true performance potential of this architecture [28]. In [21], authors use multiple 802.11 NICs per node in an ad hoc network by assuming an *identical channel assignment* to all nodes – NIC-1 is assigned channel-1, NIC-2 to channel-2, and so on. This approach can only yield a factor 2 of improvement using 2 NICs, as compared to a factor 6 to 7 improvement possible with our channel assignment scheme. Use of multiple radios in conjunction with directional antennas has been discussed in [22], [32] and [17]. In the current solutions using this approach, a node either requires MAC modifications to support beamforming [17], or requires a separate radio to communicate with each of its neighbors [22] [32]. In contrast, our architecture can give substantial performance improvements using a small number of radios on each node. We first proposed the use of multiple NICs per node, and described how to exploit the performance potential of this approach using centralized channel assignment and routing algorithms in [28]. We also showed why a traffic load-unaware channel assignment algorithm [31] fails to do so. In the current paper, we propose a set of distributed load-aware channel assignment and routing algorithms that can realize the raw performance potential of the multi-NIC architecture in the context of multi-hop wireless access networks. Compared with an earlier version of this multi-channel WMN architecture [11], this version completely eliminates the need of separate control interface, incorporates prioritized channel assignment to emulate a logical fat tree structure, and supports fast failure recovery. In addition, the current paper develops and details overall architecture and algorithms, and provides the results of the first comprehensive performance study of them.

C. Load-balancing Ad hoc Network Routing

A vast amount of research has been conducted in single-channel multi-hop routing in ad hoc networks [12] [13]. Specifically, in [22] authors propose an algorithm for load-balancing wireless links of a gateway node that connects a wireless access network to the wired network. In this algorithm, the gateway node co-ordinates the movement of all the nodes across the tree to achieve load-balancing. In contrast, our load-balanced routing algorithm does not require such coordination or centralized computation. Various metrics have been proposed to measure an individual "link-load" in a wireless network – using the number of packets [27] or number of paths [26] going through a particular node and its interference zone. We similarly measure the traffic going through each node, and compute the available bandwidth on different links of a path as well as on the gateway node to find a load-balancing route. As different links in a neighborhood can operate on different channels, we separate out each channel load and use it to compute the residual bandwidth available to any link. Finally, in [25] authors consider the problem of load-balancing across different nodes sitting on a narrow strip. The proposed algorithm combines the two greedy strategies – forward each packet to the least-loaded node and forward each packet to the furthest node. We additionally take into account the load on interfering nodes, and determine the bandwidth available to each node.

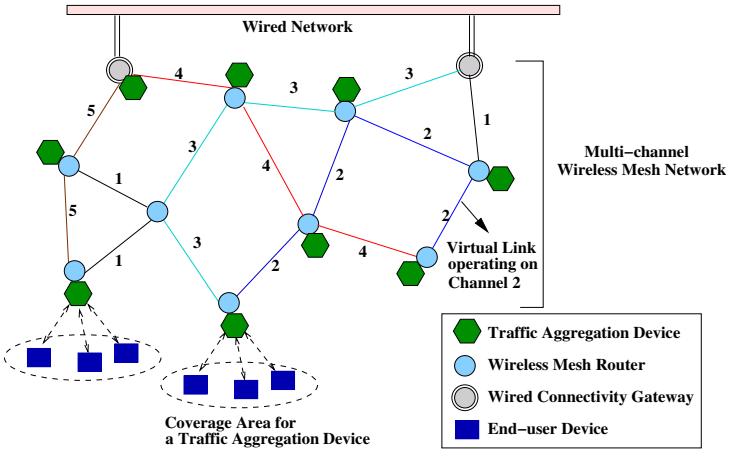


Fig. 1. *The* Hyacinth *architecture consists of a multi-channel wireless mesh network (WMN) core, which is connected to a wired network through a set of wired connectivity gateways. Each WMN node has multiple interfaces, each operating at a distinct radio channel. A WMN node is equipped with a traffic aggregation device (similar to an 802.11 access point) that interacts with individual mobile stations. The multi-channel WMN relays mobile stations' aggregated data traffic to/from the wired network. The links between nodes denote direct communication over the channel indicated by the number on the link. In this example, each node is equipped with 2 wireless NICs. Therefore the number of channels any node uses simultaneously cannot be more than 2; the network as a whole uses 5 distinct channels.*

D. Topology Discovery and Traffic Profiling

The other aspect of any WMN is the topology discovery. Several topology discovery algorithms have been proposed earlier [7] [8]. Specifically in [7], authors discuss the application of mobile agents paradigm to neighbor discovery and traffic information exchange. In our mesh network, the nodes use similar techniques to discover neighboring mesh nodes as well as find load on various channels in the interference zone. To tune the network channel and route assignments to the current network load, we perform traffic profiling on each node, similar to ones discussed in [9] and [10].

III. Problem Formulation

A. System Architecture

As shown in Fig 1, the wireless mesh network (WMN) architecture that this work targets at consists of fixed wireless routers, each of which is equipped with a traffic aggregation access point that provides network connectivity to end-user mobile stations within its coverage area. In turn, the wireless routers form a multi-hop ad hoc network among themselves to relay the traffic to and from mobile stations. Some of the WMN nodes serve as gateways between the WMN and a wired network. All infrastructure resources such as file servers, Internet gateways and application servers, reside on the wired network and can be accessed through any of the gateways. In the most general case, the physical links between gateways and the wired network can be a wired link, or a point-to-point 802.11 or 802.16 wireless link.

Each node in a multi-channel WMN is equipped with multiple 802.11-compliant NICs, each of which is tuned to a particular radio channel for a relatively long period of time, such as several minutes or hours. For direct communication, two nodes need to be within *communication range* of each other, and need to have a common channel assigned to their interfaces. A pair of nodes that use the same channel and are within *interference range* may interfere with each other's communication, even if they cannot directly communicate. Node pairs using different channels can transmit packets simultaneously without interference. For example, in Fig 1, each node is equipped with 2 NICs. The "virtual links" shown between the nodes depict direct communication between them, and the channel used by a pair of nodes is shown as the number associated with the connecting link. This example network totally uses 5 distinct channels. Note that mobile nodes have only a single NIC, and the interaction between mobile nodes and a traffic aggregation device is similar to the infrastructure mode operation of the IEEE 802.11 standard.

B. Channel Assignment Problem

Intuitively, the goal of channel assignment in a multi-channel WMN is to bind each network interface to a radio channel in such a way that the available bandwidth on each virtual link is proportional to the load it needs to carry. This problem is different from the channel assignment problem in cellular networks [23], because adjacent base stations in a cellular network are connected through wired networks, whereas adjacent nodes in a WMN can only communicate with each other through wireless links. Therefore, if one simply assigns a least-used channel to a WLAN interface, there is no guarantee that the resulting mesh network is even connected. A *Hyacinth* node needs to share a common channel with each of its communication-range neighbors with which it wishes to set up a virtual link or connectivity. On the other hand, to reduce interference a node should minimize the number of neighbors with whom to share a common channel. More generally, one should break each collision domain into as many channels as possible while maintaining the required connectivity among neighboring nodes.

The channel assignment problem can actually be divided into two subproblems: (1) neighbor-to-interface binding, and (2) interface-to-channel binding. Neighbor-to-interface binding determines through which interface a node uses to communicate with each of its neighbors with whom it intends to establish a virtual link. Because the number of interfaces per node is limited, each node typically uses one interface to communicate with multiple of its neighbors. Interface-to-channel binding determines which radio channel a network interface should use. The main constraints that a channel assignment algorithm needs to satisfy are

- The number of distinct channels that can be assigned to a WMN node is bounded by the number of NICs it has.
- Two nodes that communicate with each other directly should share at least one common channel.
- The raw capacity of a radio channel within an interference zone is limited.
- The total number of non-overlapped radio channels is fixed.

At a first glance, this problem appears to be a graph-coloring problem. However, standard graph-coloring algorithms cannot really capture its requirements and constraints. A node-multi-coloring formulation [24] fails to capture the second constraint where communicating nodes need to be assigned a common

color. An edge-coloring formulation fails to capture the first constraint where no more than q (number of NICs per node) colors can be incident onto a node. While a constrained edge-coloring might be able to roughly model the remaining constraints, it is still incapable of satisfying the third constraint of limited channel capacity. Conceptually, links that need to support higher traffic load should be given more bandwidth than others. This means that these links should use a radio channel that is shared among a fewer number of nodes. An ideal load-aware channel assignment would distribute radio resource among links in a way that matches their expected traffic loads.

C. Load-Balancing Routing Problem

Channel assignment depends on the load on each virtual link, which in turns depends on routing. The traffic distribution of a WMN is skewed – most of the WMN nodes communicate primarily with nodes on the wired network. This is the case because most users are primarily interested in accessing the Internet or enterprise servers, both of which are likely to reside on the wired network [22]. The goal of the routing algorithm is thus to determine route(s) between each traffic aggregation device and the wired network in such a way that balances the load on the mesh network, including the links to the wired network. Load balancing helps avoid bottleneck links, and increases the network resource utilization efficiency.

D. Evaluation Metric

The ultimate goal of the channel assignment and routing algorithms is to maximize the overall network goodput, or the number of bytes it can transport between the traffic aggregation devices and the wired connectivity gateways within a unit time. To formalize this goal, we define the *cross-section goodput* of a network as

$$X = \sum_{a} min(\sum_{i} C(a, g_i), B(a)) \quad (1)$$

where $C(a, g_i)$ is the *useful* network bandwidth available between a traffic aggregation device a and a gateway node g_i. If the bandwidth requirement between a traffic aggregation device a and the wired network is $B(a)$, then only up to $B(a)$ of the bandwidth between node a and all the gateway nodes is considered useful. This criteria ensures that only the usable bandwidth of a network is counted towards its cross-section throughput, hence the term cross-section goodput. The goal of the channel assignment and routing algorithms is to maximize this cross-section goodput X.

E. Centralized Channel Assignment Algorithm

Although the notion of *network-wide load balancing* [28] is conceptually simple, it is rarely used in previous load balancing routing algorithms because it is surprisingly difficult to capture quantitatively. Even with a complete knowledge of network topology and traffic matrix, the channel assignment problem is *NP-hard*. We prove its hardness by reducing the *multiple subset sum* problem to the channel assignment problem [28]. To establish a baseline, we develop a greedy centralized algorithm to the channel assignment/routing problem in multi-channel WMNs [28]. The algorithm works by first estimating the load imposed on each virtual link by each traffic flow in the given traffic matrix, and thus the total expected load on each virtual link. The channel assignment algorithm then visits all the virtual links in decreasing order of their expected loads. Upon visiting a particular virtual link, the algorithm greedily assigns it a channel that leads to minimum interference and contention with neighboring nodes in the interference zone whose WLAN interfaces have already been assigned to specific channels.

IV. Distributed Routing/Channel Assignment Algorithm

In this section, we present a distributed routing/channel assignment algorithm that utilizes only local topology and local traffic load information to perform channel assignment and route computation. This information is collected from a $(k + 1)$-hop neighborhood, where k is the ratio between the interference and communication ranges, and is typically between 2 and 3.

A. Load-Balancing Routing

As most of the traffic on a WMN is directed to/from the wired network, each WMN node needs to discover a path to reach one or multiple wired gateway nodes. Logically, each wired gateway node is the root of a spanning tree, and each WMN node attempts to participate in one or multiple such spanning trees. These spanning trees are connected to each other through the wired network. When each WMN node joins multiple spanning trees, it can distribute its load among these trees and also use them as alternative routes when nodes or links fail. However, a WMN node may need additional wireless network interfaces to join multiple trees. In this paper, we restrict our focus on the case where each node is actively associated with only one of the trees and uses the other trees only for failure recovery.

1) Routing Tree Construction: The basic tree construction process is similar to IEEE 802.1D's spanning tree formation algorithm [6] with two major differences – (a) the metric used by each WMN node to determine a parent is dynamic to achieve better load balancing, and (b) load-aware channel assignment technique is used to automatically form a fat-tree where more relay bandwidth is available on virtual links closer to the roots of the trees, i.e., wired gateways.

Assume a node X has already discovered a path to the wired network. It periodically, every T_a time units, broadcasts this reachability information to its one-hop neighbors using an ADVERTISE packet. Initially, only the gateway nodes can send out such advertisements because of direct connectivity to the wired network. Over time, intermediate WMN nodes that have a multi-hop path to one of the gateway nodes can also make such advertisements. The ADVERTISE packet that X sends out contains the "cost" of reaching the wired network through X. Upon receiving an advertisement, X's neighbor, say node Y, can decide to join X if Y does not have a path to the wired network, or the cost to reach the wired network through X is less than Y's current choice. To join node X, Y sends a JOIN message to X. On receiving the JOIN message, X adds Y to its children list, and sends an ACCEPT message

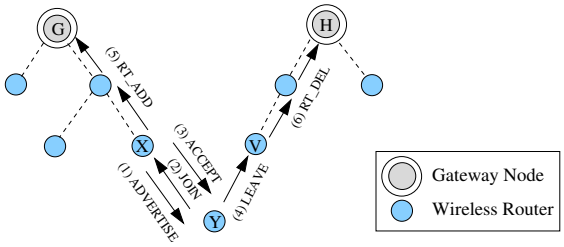


Fig. 2. *A distributed route discovery/update protocol used to establish routes between multi-channel WMN nodes and wired gateways.*

to Y with information about channel(s) and IP address to use for forwarding traffic from Y to X. In terms of the routing tree, X is now the *parent* of Y, and Y is one of the *children* of X. Finally, Y sends a LEAVE message to its previous parent node, say V. From this point on, Y also broadcasts ADVERTISE packets to its own one-hop neighbors to further extend the reachability tree. Fig 2 shows the message exchange sequence.

As a result of the exchange of JOIN/ACCEPT/LEAVE messages, the routing tables on the involved nodes are updated. First, the default routing entry of Y points to X as the next hop. All nodes in the tree from V upwards to the corresponding gateway node *delete* the forwarding entries pointing to Y and its children, if any. On the other hand, all nodes in the tree from X upwards to the gateway node *add* a forwarding entry for packets destined to Y and its children. To perform these route updates, the RT_ADD/RT_DEL messages are sent up to the root of the corresponding trees, as shown in Fig 2.

Delivery of some protocol messages such as JOIN, ACCEPT and LEAVE needs to be reliable for consistent network operations. These reliable connections are built as part of the *Neighbor Discovery Protocol*, wherein a new node broadcasts a HELLO message to its one-hop neighbors. Upon receiving this HELLO message from a new node, each of its neighbors establishes a reliable connection with the new node and also sends an ADVERTISE message to expedite the route discovery for the new node. A reliable connection is built on top of the UDP layer and is used for delivering all control messages that require reliability. The HELLO message itself can be lost, and is thus broadcasted multiple times to minimize the probability of message loss. The ADVERTISE message, in contrast, is sent as an unreliable broadcast packet for efficiency reasons.

2) Routing Metric: The "cost" metric carried in the ADVERTISE messages determines the final tree/forest structure. We explore three different cost metrics. First is the *hop count* between a WMN node and the gateway node associated with an ADVERTISE message. This metric enables a WMN node to reach the wired network using the minimum number of hops, but does nothing to balance network load. An advantage of using the hop-count metric is rapid convergence, as the minimum hopcount from a node to a wired network is determined by physical topology and is thus mostly static.

The second cost metric is the *gateway link capacity*, which indicates the residual capacity of the *uplink* that connects the root gateway of a tree to the wired network. Residual capacity of any link is determined by subtracting the current usage of the link from its overall capacity. In case the total bandwidth of a gateway's wireless links is smaller than its uplink, we take the wireless links' bandwidth as the gateway link capacity. The third cost metric is the *path capacity*, which represents the minimum residual bandwidth of the path that connects a WMN node to the wired network. Path capacity is more general than gateway link capacity because the former assumes that the bottleneck of a path can be any constituent link on the path, rather than always the gateway link. The *capacity of a wireless link* is approximated by subtracting the aggregate usage of the link's channel within its neighborhood from the channel's raw capacity which is assumed to be fixed within any collision domain.

The latter two metrics are dynamic, and can result in route flaps and a non-convergent network behavior. Route flaps occur when multiple nodes discover and switch to an underutilized path at the same time. Such simultaneous switching results in overloading of the originally underutilized path. This problem is similar to the route flapping problem observed on the wired Internet [29]. One can use similar measures as in BGP to dampen these route flaps reactively. We prevent route flaps by introducing a slight modification of the protocol. Since gateway is the only node aware of its latest link load, we propagate JOIN message up to the gateway. The gateway can now send an ACCEPT or a REJECT back to the newly joining node based on the gateway's latest residual link capacity. In addition, any intermediate node can also send a REJECT message to new requests if its capacity has decreased because other nodes switched to join its subtree. Additionally, the RT_ADD message also updates the current link usage on each hop of the path. This protocol modification effectively addresses the route flap problem at its source itself. As shown in the performance section, the bandwidth overhead introduced by this protocol change is fairly small.

B. Distributed Load-Aware Channel Assignment

The neighbor discovery and routing protocol in the previous subsection allows each WMN node to connect with its neighbors and identify a path to the wired network. We now discuss the mechanisms through which a WMN node can decide how to bind its interfaces to neighbors and how to assign radio channels to these interfaces without global coordination, as in the case of centralized algorithm.

1) Neighbor-Interface Binding: The key problem in the design of a *distributed* channel assignment algorithm is *channel dependency* among the nodes, which is illustrated in Fig 3. In this example, assume node D finds that the link D-E is heavily loaded and should be moved to a lightly loaded channel 7. As D only has 2 NICs, it can only operate on two channels simultaneously. To satisfy this constraint, link D-F also needs to change its channel. The same argument goes for node E, which needs to change the channel assignment for link E-H. This ripple effect further propagates to link H-I. Similar ripple effects would ensue if link A-E were to change its channel. In this case, link E-G and G-K will need to change their channels as well. This channel dependency relationship among network nodes makes it difficult for an individual node to predict the effect of a local channel re-assignment decision.

To bound the impact of a change in channel assignment, we impose a restriction on the WMN nodes. Specifically, the set of NICs that a node uses to communicate with its

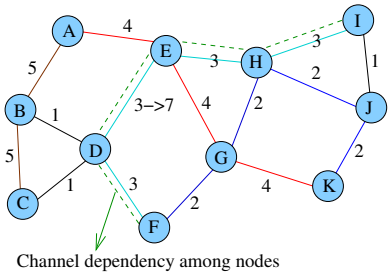


Fig. 3. *This example shows how a change in channel assignment could lead to a series of channel re-assignments across the network because of the channel dependency problem.*

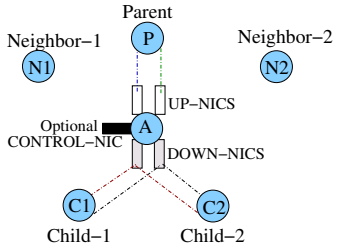


Fig. 4. *Eliminating the channel dependency problem by separating the set of NICs used in each WMN node into UP-NICs and DOWN-NICs so that any channel assignment change in a WMN node's DOWN-NICs does not affect its UP-NICs.*

parent node, termed *UP-NICs*, is disjoint from the set of NICs the node uses to communicate with its children nodes, called *DOWN-NICs*, as shown in Fig 4. Each WMN node is responsible for assigning channels to its DOWN-NICs. Each of the node's UP-NICs is associated with a unique DOWN-NIC of the parent node and is assigned the same channel as the parent's corresponding DOWN-NIC. This restriction effectively prevents channel dependencies from propagating from a node's parent to its children, and thus ensures that a node can assign/modify its DOWN-NICs' channel assignment without introducing ripple effects in the network. Because a gateway node does not have any parent, it uses all its NICs as DOWN-NICs. To increase the relay capability, each non-gateway node attempts to equally divide its NICs into UP-NICs and DOWN-NICs. However, a node farther from the gateway is assigned a lower priority in choosing channels, and thus may not get the required bandwidth on any single channel. In this case, the node can dedicate more NICs as DOWN-NICs to aggregate the leftover bandwidth from multiple channels.

2) Interface-Channel Assignment: Once the neighbor-to-interface mapping is determined, the final question is how to assign a channel to each of the NICs. The channel assignment of a WMN node's UP-NICs is the responsibility of its parent. To assign channels to a WMN node's DOWN-NICs, it needs to estimate the usage status of all the channels within its interference neighborhood. Each node therefore periodically exchanges its individual channel usage information as a CHNL_USAGE packet with all its $(k+1)$-hop neighbors, where k is the ratio of the interference range and the communication range. Because all the children and parent of a node, say A, can interfere with their own k hop neighbors, A's $(k+1)$-hop neighborhood includes all the nodes that can potentially interfere with A's communication. The aggregate traffic load of a particular channel is estimated by summing up the loads contributed by all the interfering neighbors that happen to use this channel. To account for the MAC-layer overhead such as contention, the *total load of a channel* is a weighted combination of the aggregated traffic load and the number of nodes using the channel.

Based on the per-channel total load information, a WMN node determines a set of channels that are least-used in its vicinity. As nodes higher up in the spanning trees need more relay bandwidth, they are given a higher priority in channel assignment. More specifically, the priority of a WMN node is equal to its hop distance from the gateway. When a WMN node performs channel assignment, it restricts its search to those channels that are not used by any of its interfering neighbors with a higher priority. The outcome of this priority mechanism is a *fat-tree* architecture where links higher up in the tree are given higher bandwidth.

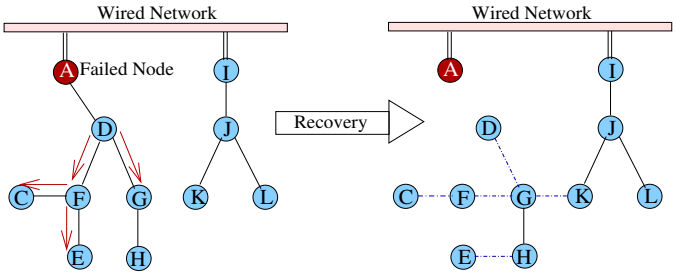
Because traffic patterns and thus channel loads can evolve over time, the interface-to-channel mapping is adjusted periodically, every T_c time units. Within a *channel load-balancing* phase, a WMN node evaluates its current channel assignment based on the channel usage information it receives from neighboring nodes. As soon as the node finds a relatively less loaded channel after accounting for priority and its own usage of current channel, it moves one of its DOWN-NICs operating on a heavily-loaded channel to use the less-loaded channel, and sends a CHNL_CHANGE message with the new channel information to the affected child nodes, which modify the channels of their UP-NICs accordingly. The node also sends an updated channel usage map to its $(k+1)$-hop neighbors. This quick channel usage update strategy ensures that other nodes in the neighborhood do not migrate to the new channel because they assume (incorrectly) that it is still less loaded. The probability of race condition is further reduced by skewing the load-balancing phases among neighboring nodes.

In the case that a relatively less-loaded channel is unavailable for a NIC operating on a heavily-loaded channel, the WMN node performs *child-interface load balancing*. Here, it re-distributes its children among its DOWN-NICs such that the DOWN-NICs' channels get more uniformly loaded.

C. Virtual Control Network

Unlike in a single-channel mesh network, nodes in a multi-channel WMN may not share any common channel with some of their physical neighbors. One simple option is to add a *CONTROL-NIC* on each node, tune it to a common channel, and route all control traffic such as ADVERTISE messages over this control network. This additional hardware interface can be saved by forming a *virtual control network* over the same multi-channel mesh network to exchange control packets.

A WMN node needs to communicate each control message to its c-hop *physical* neighbors, where c depends on the message's type and can range from 1 to $(k+1)$, where k is again the ratio of interference and communication ranges. Since there is always a path between a WMN node and its physical neighbors, a control message can be delivered through one or multiple hops on the mesh network. For efficiency reasons, the broadcast control messages are delivered using *IP multicast*. Essentially the idea here is to implement layer-2



(a) Failure message after node A fails (b) New connectivity after recovery

Fig. 5. *This example network illustrates the distributed failure recovery protocol used in* Hyacinth.

communication using layer-3 routing so as to eliminate the dedicated control interface on each node. With the use of virtual control network, a *new* node needs to scan all channels for broadcasting HELLO messages during the neighbor discovery phase. Channel scanning process can be done in 5 to 10 sec. When two nodes scan the channels simultaneously, each node only uses its UP-NIC to perform the scanning, while keeping the DOWN-NIC at a fixed channel. This ensures that the neighboring nodes can eventually discover each other.

D. Failure Recovery

When a node fails, nodes in its subtree lose their connectivity to the wired network. *Hyacinth* reorganizes the network to bypass the failed node and restore the connectivity. To accommodate node failures, each WMN node remembers *alternative* advertisements it has received from all other potential parent nodes. Upon detection of a parent-node failure, each of its child nodes sends a JOIN message to a "backup" parent node, and re-establishes its connectivity with the wired network. This scheme allows fast recovery from a node failure without committing any additional physical radio resources.

Not all failures can be handled locally, as shown in Fig 5(a). Here, when node A fails, its child node D does not have a ready-backup parent that can re-connect it to the wired network. To recover from such failures, node D sends a FAILURE message to its immediate children F and G, asks them to perform their own failure recovery, and forces itself to go back to the channel-scanning mode. The FAILURE message contains the list of failed parent nodes, in this case, A and D. Each child node in turn attempts to associate with its respective backup parents. However, in this process the child nodes actively avoid known failed nodes, in this case A and D. If a child cannot find any usable backup parent, it recursively broadcasts a FAILURE message to its children after adding its own name to the failed-nodes-list included in the FAILURE message, and goes to channel-scanning mode. In Fig 5(a), node G performs local recovery, while node F relays the FAILURE message to its own children C and E. Fig 5(b) shows the final network connectivity after the recovery process is completed. A failure recovery is eventually followed by the usual periodic traffic profiling, and channel adjustments to rebalance the network load across different channels.

V. Performance Evaluation

We studied the performance gains of the proposed multi-channel WMN architecture and the effectiveness of the proposed channel assignment and routing algorithms through extensive ns-2 simulations. We modified ns-2 to support multiple wireless cards on each wireless node and to support dynamic channel assignment. The evaluation metric for most experiments is *cross-section goodput*, which is defined as the sum of all useful bandwidth between traffic aggregation nodes in a WMN and their corresponding gateway nodes (eqn 1). The following are the *default settings* for the simulations. Each node is equipped with 2 NICs, and the number of physical channels is 12. For effective multi-hopping, RTS/CTS mechanism is enabled. The ratio between the interference range and the communication range is set to 2. Channel load-balancing period T_c of a node is set to 1 minute, while the channel-usage and routes advertisement frequency, is set to once every 30 seconds (T_a).

A. Improvements due to Multi-channel Mesh Networking

We measured the throughput improvements achieved by *Hyacinth*'s multi-channel mesh networking architecture using different channel assignment algorithms. Ten different 60-node network topologies are generated, each randomly sampled from a 9x9 square grid network. Based on the topology and location, each node could communicate with up to 4 neighbors. Four of these 60 nodes were designated as the gateway nodes and connected to the wired network. For each topology, 30 nodes were chosen at random to generate traffic flows. Each traffic flow represents an aggregate of traffic streams from multiple users. The average bandwidth for each flow is chosen at random between 0 and 3 Mbps.

To drive the network to saturation, the bandwidth of all the flows is proportionally varied until the network can only route 80% of the aggregate input traffic. The *relative* performance of different algorithms does not change for other values of saturation threshold, e.g. 100% at which we ensure that each flow has to be assigned its full required bandwidth. The same is true when the saturation threshold is made per-flow to ensure fairness across different flows. For example, one can ensure that each flow has to be assigned at least a certain percentage of its traffic requirement. For brevity, we only show the overall cross-section goodput for all the graphs.

The results in Fig 6 show that even with identical channel assignment scheme [21], deploying 2 NICs on each node improves the network goodput by a factor of 2 compared with conventional single-channel network. With the proposed distributed channel assignment algorithm the network throughput becomes 6 to 7 times that of single-channel network. Intuitively, *Hyacinth*'s channel assignment algorithm breaks a collision domain in a single-channel network into multiple collision domains each operating in a different frequency range. This *division of collision domain across different frequency channels* is the key reason for the nonlinear goodput improvement (6-7 times) with respect to the increase in the number of NICs (from 1 to 2). Moreover, the interference among adjacent hops of an individual path and among neighboring paths is greatly reduced.

The first distributed channel assignment scheme [11], called *physical control network*, uses a dedicated control channel for communicating all control traffic. This requires an additional WLAN interface on each node specifically for control traffic. The second distributed channel assignment scheme, called

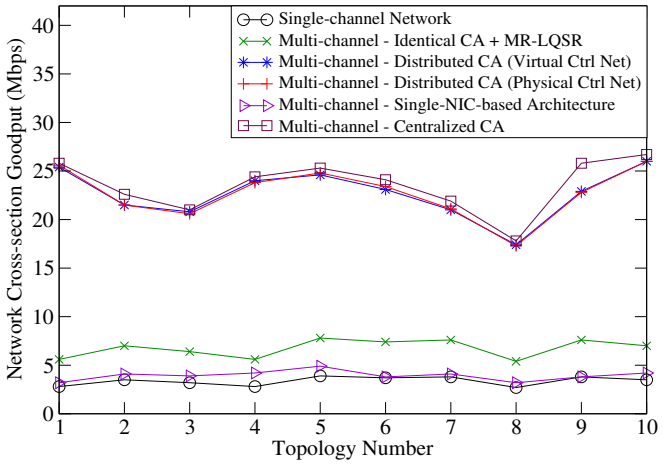


Fig. 6. *Network cross-section goodput of 10 different 60-node topologies randomly sampled from a 9x9 square grid. With just 2 interfaces on each node, the distributed channel assignment improves the network throughput 6 to 7 times as compared with a single-channel network of the same topology.*

virtual control network, multiplexes the control traffic over data NICs thereby reducing the per-node hardware cost. The fact that these two schemes have comparable performance suggests that control traffic introduces minimal overhead when multiplexed on the main data channels. Finally, the centralized channel assignment/routing algorithm does not perform much better than the distributed versions; this shows that the performance loss due to distribution of intelligence is very small.

An alternate design for a multi-channel mesh networking [30] is to equip each node with a single interface and operate the sub-network rooted at each gateway at a different channel [30]. Logically, this should reduce the contention among nodes and thus improve the network goodput. Surprisingly, this scheme does not give much throughput improvement over a single-channel mesh network as shown in Fig 6. The fact that only a single channel is used *within* a tree means that there is still heavy collision and interference on the wireless links around each gateway, which is most likely where the bottleneck is. In contrast, a true multi-channel mesh network architecture can split the wireless links around each gateway into as many collision domains as possible, thus delivering higher effective bandwidth.

We next simulated a 64-node network placed in an 8x8 grid with 4 uniformly distributed gateway nodes connected to the wired network. The remaining 60 nodes were (randomly) divided into 5 different popularity sets of equal sizes. A node's popularity determines the size of its user base and in turn the number of new HTTP connections generated every second from the node. The simulations were done using the PackMime extension of ns-2 that decides the request arrival pattern and response sizes using models discussed in [33]. Based on the node popularity, the average rate of new HTTP requests for the node was 0, X, 2X, 3X, or 4X, where X is the traffic intensity for the experiment. Fig 7 shows the response time observed by web users on this simulated network. The observed delay in the single-channel mesh network increases rapidly with the traffic intensity and eventually limits the number of users it can support. With just 2 NICs on each node, the multi-channel mesh network reduces the HTTP response time substantially. Additionally, at saturation the multi-channel

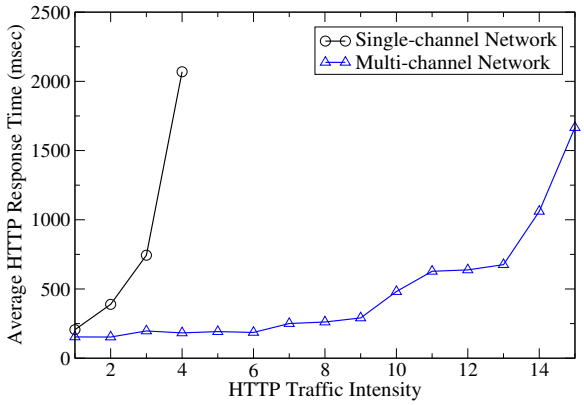


Fig. 7. *Web browsing (HTTP) response time as seen by users on a 64-node WMN. The multi-channel WMN architecture can drastically reduce the HTTP response time, as well as increase the number of users a network can support.*

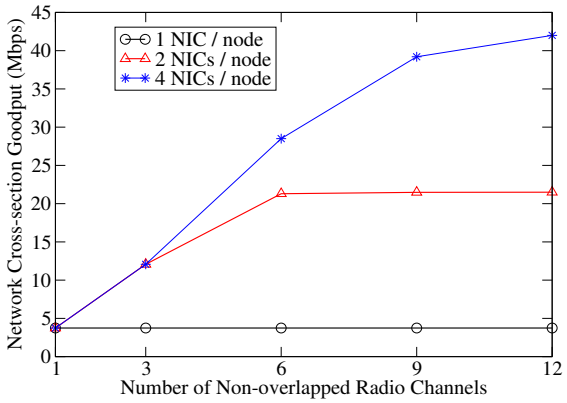


Fig. 8. *Effects of varying the number of WLAN interfaces per node and number of non-overlapped radio channels.*

WMN can support over 4 times as much web traffic as compared with the single-channel WMN, and consequently a much larger user base.

B. Impact of System Parameters

1) Number of WLAN Interfaces and Radio Channels: The number of non-overlapped radio channels is 3 for 802.11b/g and 12 for 802.11a. Fig 8 shows the effects of varying the number of radio channels on the network goodput. These experiments were conducted on a 64-node grid network with 4 gateway nodes uniformly placed across the network. The traffic was generated by 30 randomly chosen wireless mesh nodes, at an average rate chosen randomly between 0 and 3 Mbps for each node. We also vary the number of NICs per node to evaluate the impact of number of NICs on the utilization efficiency of the channels. The network goodput increases monotonically with the number of non-overlapped channels when the the number of NICs per node is kept constant, because a collision domain can be broken into more non-interfering collision domains. When each node has 2 NICs, the network goodput saturates at about 6 channels. When the number of NICs on each node is increased to 4, the network can use up to 12 channels before its performance starts to saturate.

2) Number of Gateway Nodes: A major cost component of a WMN is its gateway nodes to the wired Internet [17]. The proposed algorithms attempt to make the best of the

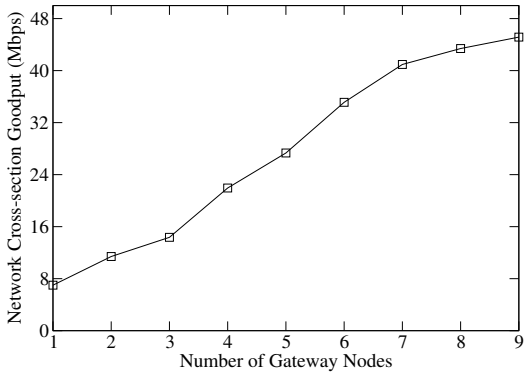


Fig. 9. *The network goodput increases with the number of gateway nodes in the network. This means that the proposed channel assignment and routing algorithm can effectively reconfigure the WMN to leverage additional bandwidth made available by additional gateways.*

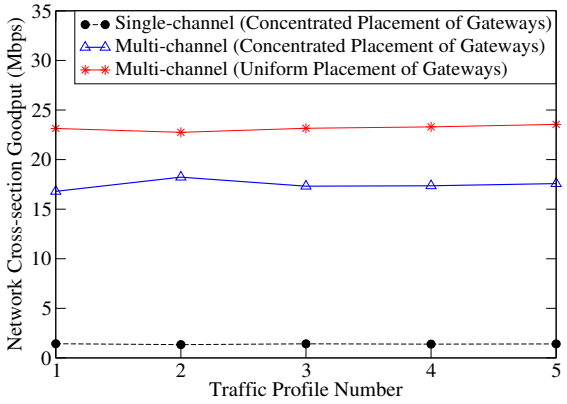


Fig. 10. *Effect of placement of gateway nodes on the network goodput. Even when gateway nodes are concentrated, the proposed algorithms can still adapt the channel assignment and routes to maximize the network goodput.*

available gateways to increase the network goodput, as shown in Fig 9. In these experiments, the number of gateway nodes was increased from one to nine in a 9x9 grid network. In the final configuration, the 9 gateway nodes were uniformly distributed across the network. Each additional gateway node improves the network goodput because it adds more capacity to relay traffic from the user nodes to the wired network, and the proposed channel assignment and routing algorithm can effectively reconfigure the WMN to leverage additional bandwidth made available by additional gateways.

3) Placement of Gateway Nodes: Fig 10 shows the effect of placement of gateway nodes on the network goodput. As expected, the best performing configuration is the uniform placement of gateways as it makes it easier to spread the traffic load among the gateways. However, even when gateway nodes are concentrated in one place, the proposed channel assignment and routing algorithm can still reap most of the performance benefits. Concentration of gateway nodes is desirable because it simplifies installation (wiring) and management of gateway nodes, reducing the overall cost of WMN.

C. Impact of Algorithm Parameters

1) Channel Selection Criterion: To select a channel for an interface, a node needs to estimate the load on each of the available physical channels. We compare three different criterion used in selecting a channel. The first criterion uses the number of interfering nodes sharing a channel. The second criterion estimates a channel's load by adding up the channel usage of each of the interfering nodes. The third criterion takes into account a channel's load as well as whether it is being used by nodes closer to some gateways. By reducing the extent of congestion and collision for channels used by nodes closer to a gateway, this metric provides higher capacity to links closer to the root of a tree, thus enabling a fat tree architecture.

As shown in Fig 11, using the measured channel usage as the channel selection criterion gives substantial performance improvement over the criterion based only on the number of nodes sharing a channel, because measured channel usage is a more accurate way to reflect the load of a channel. Surprisingly, adding channel prioritization does not help at all. The reason is that channels used by links closer to the gateway nodes are typically more loaded, and as a result tend to be avoided by nearby descendant nodes that select channels

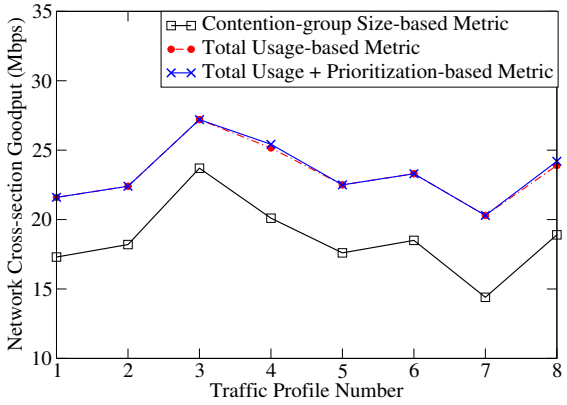


Fig. 11. *Performance comparison among three channel selection criteria. Measured channel usage is a more accurate way to estimate the load of a channel.*

based only on the measured channel usage criterion. This automatically assigns more bandwidth to links closer to the gateway nodes, thus forming a fat-tree.

Fig 12 shows the effectiveness of using measured channel usage as the channel selection criterion to allocate capacities to links in proportion of their bandwidth requirements. Before channel load balancing, a large fraction of network links have a high packet drop rate in either the interface queue or over the air. Channel load balancing puts heavily loaded links onto different channels, substantially reducing these packet-drops.

The current *Hyacinth* prototype uses both measured channel usage and contention group size in channel selection: If the channel usages of two channels differ by more than 10%, the lightly loaded one is chosen; otherwise, the channel with a smaller contention group size is chosen.

2) Routing Metric: We compare the impact of various routing metrics on the overall network performance for skewed traffic profiles. These experiments were conducted over a 64-node grid network with 4 uniformly placed gateway nodes. For each traffic profile, 20 different traffic aggregation devices were chosen in a *skewed* manner, specifically closer to two of the gateway nodes. As expected, *shortest path routing* does not utilize the gateways' bandwidth effectively. *Gateway load balancing* considers the available bandwidth on each of the gateway nodes while choosing paths, and *path load balancing* estimates the end-to-end available bandwidth between a WMN node and a gateway node when choosing the routing path. Per-

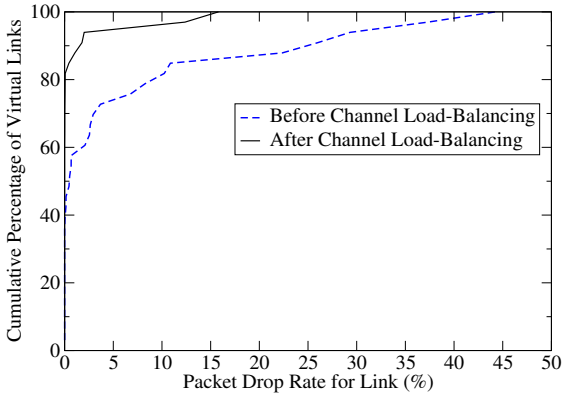


Fig. 12. *Effectiveness of usage-based channel selection in dividing the interference zone across different frequencies. Packet drop rates are decreased because of reduced contention among links as well as allocation of higher capacity to more loaded links.*

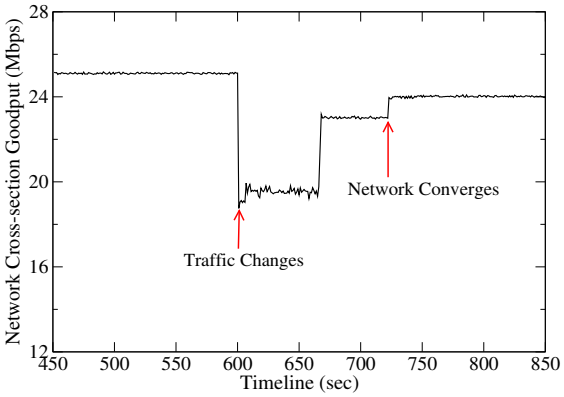


Fig. 14. *Responsiveness of a* Hyacinth *network's adaptation to changes in traffic load distribution. The traffic profile switches from one set of 30 aggregated flows to another at time 600 seconds. The network converges to a new configuration in 2 load-balancing periods, each of which is 60 seconds.*

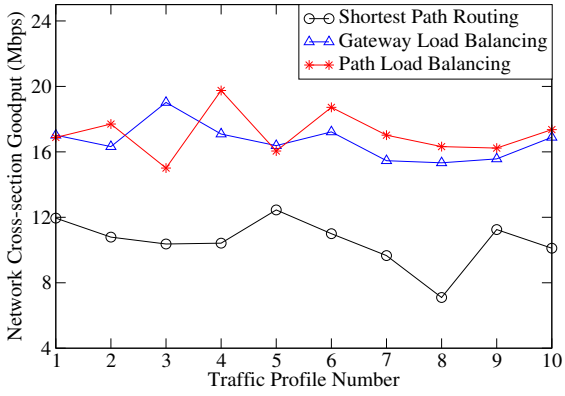


Fig. 13. *Performance comparison among load balancing routing metrics in the presence of skewed traffic. Gateway load balancing only looks at the available bandwidth on the gateway nodes, while path load balancing considers end-to-end available bandwidth between a WMN node and a gateway node.*

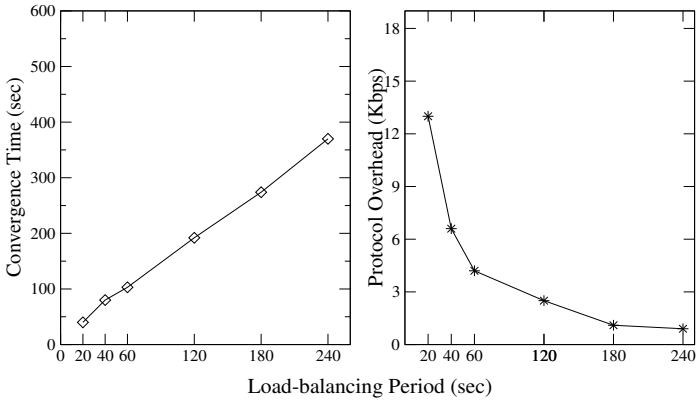


Fig. 15. *Impact of load-balancing period on network convergence time against change in traffic load, and the corresponding bandwidth overhead. The network typically converges in less than 2 load-balancing periods, because the load-balancing periods for different nodes are de-synchronized. The bandwidth overhead due to load balancing actions is relatively low even for a small load-balancing period of 20 seconds, and reduces logarithmically with increasing load-balancing period.*

formance of path load balancing is only slightly better than that of gateway load balancing, suggesting that gateways are the main bottlenecks. In a real-world network, even intermediate wireless links could form bottlenecks due to interference from other radio sources, and path load balancing should find more optimal paths than gateway load-balancing.

D. Traffic Adaptation and Protocol Complexity

In *Hyacinth*, the network nodes continuously monitor the loads on their interfaces, exchange channel usage information every 30 sec, and make load-balancing decisions every 60 sec. The network goodput should improve as nodes modify their channel assignments and routing. Fig 14 shows how a 64-node *Hyacinth* network adapts to a change in traffic loads. In this case, one traffic profile of 30 aggregated flows changes to another at time 600 sec, and the network converges to a new configuration within 2 load-balancing periods at time 723 sec.

Fig 15 shows the impact on the convergence time of the load-balancing period. For lower load-balancing periods, the network converges within 2 load-balancing periods. For higher load-balancing periods, it takes 1.5 rounds on the average to converge. Different load-balancing periods also incur different bandwidth overheads. This overhead is less than 15 Kbps even with very frequent load-balancing, At the default 60-second load-balancing period for our experiments, the protocol overhead is 5 Kbps, which is about 0.025 % of the maximum network cross-section goodput.

VI. Prototype Implementation and Evaluation

To demonstrate the feasibility of the proposed multi-channel WMN architecture, we built a 9-node *Hyacinth* prototype. In this section, we summarize the key components of this prototype, and report the results obtained from a brief performance study conducted on it.

A. Hardware Components

Each node in the current *Hyacinth* prototype is a standard desktop PC running Windows XP equipped with two different network interfaces – an Orinoco 802.11a/b/g PCI card and a Netgear 802.11a/b/g PCI card. Both cards operate in the 802.11a ad-hoc mode. Although the 802.11 interfaces mounted on the same machine operate on non-overlapped channels, they still interfere with one another [21]. We believe this interference arises from radiation leakage because of imperfect channel-filter hardware in commodity cards. We reduce this interference by using PCI cards equipped with external antennas (separated by 2 feet distance) and no internal antenna.

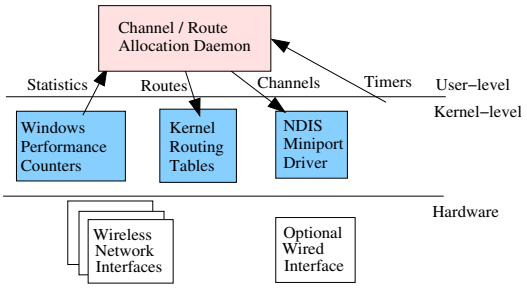


Fig. 16. *The software architecture of an individual multi-channel wireless mesh network node in the* Hyacinth *prototype*

Additionally, the channels assigned to the two cards mounted on the same machine are at least one channel apart from each other. Two out of these nine nodes serve as gateway nodes that connect the prototype to our department's wired network.

B. Software Architecture

Fig 16 shows the overall software architecture of a WMN node. To enable fast packet forwarding, *Hyacinth* directly uses the kernel's packet forwarding routine. The routing/channel assignment daemon running at the user-level, modifies the NICs' IP addresses and the kernel routing table so as to utilize multiple NICs while forwarding packets over multiple hops. ICMP-redirect is disabled to avoid the need to allocate addresses from different subnets. IP address assignment to mesh nodes is done through a multi-hop DHCP protocol. A node first assigns a temporary IP address from a small address space 192.168.254.1 to 192.168.254.254, and uses this address to connect to a global DHCP server sitting on the wired network. Upon contacting the global DHCP server, the node receives a set of new IP addresses (one for each NIC) that are unique within the entire mesh network.

Each DOWN-NIC and the associated child nodes' UP-NICs are assigned to a common channel and a common essid. This assignment is done using Network Driver Interface Specification (NDIS) API calls. These calls set the corresponding parameters and then reset the card. Channel assignment/Routing protocol also requires collecting detailed traffic statistics from individual interfaces. The daemon extracts this information from the kernel using Windows performance counters.

Each WMN node connects to one or more 802.11b access points, and is responsible for assigning IP addresses to mobile stations by acting as a local DHCP server. The IP addresses are allocated from a range that is obtained from the global DHCP server. This ensures that each mobile station gets a mesh-wide unique IP address, which is required for supporting mobility. To support user mobility, each WMN node also act as a home/foreign agent and runs a Mobile-IP like protocol. Details of mobility support are beyond the scope of this paper.

C. Hyacinth Prototype Performance

Fig 17 shows the topology of the 9-node *Hyacinth* prototype we built. The nine nodes are placed in an area of size approximately 20m x 10m spanning two lab rooms with Node-1 and Node-9 connected to the department wired network. The transmit power on each node is reduced to 1 mW to limit interference zones of individual nodes. For evaluation purposes, the prototype can be run in two different modes

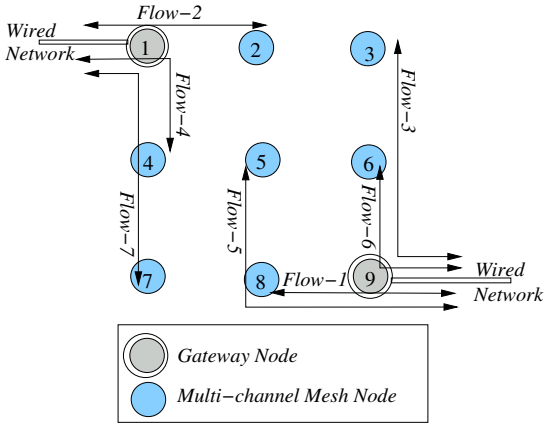


Fig. 17. *Physical topology of the 9-node* Hyacinth *prototype. Each node is equipped with 2 802.11a PCI NICs whose channels are tuned dynamically by the channel/route daemon. Node 1 and Node 9 are connected to the wired network providing access to departmental servers and the Internet.*

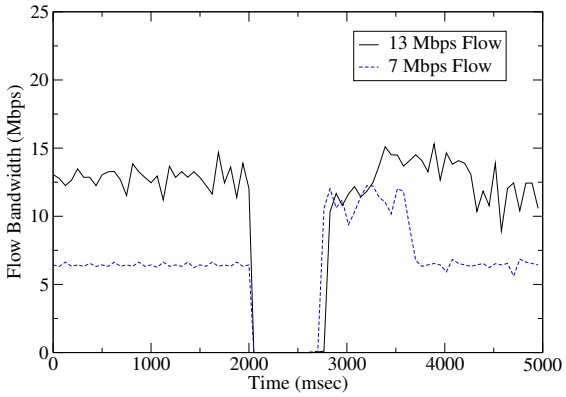


Fig. 18. *Failure recovery time for Node 3 in Fig 17, when node 6 fails. Failure occurs at time 2000 msec, and network recovers within 700 msec.*

– single-channel mode, and multi-channel mode. The single-channel mode only uses one of the cards to form the mesh network.

1) FTP bandwidth: We measured the performance of FTP flows (shown in Fig. 17) that download data from the department server simultaneously. The aggregate performance of the flows in the multi-channel operation mode is 55.58 Mbps, which is about 5 times the aggregate throughput in the single-channel operation mode (11.32 Mbps). Measurements of upload FTP traffic showed similar performance gains. These results match closely with those obtained from an ns-2 simulation of the prototype, and thus validate our previous simulation results. The throughput improvement over single-channel case is limited to 5 times as opposed to 6 or 7 times, because of the small size of the prototype. A larger network should provide higher throughput gains with multi-channel mesh networking.

2) Failover Latency: We evaluated the failover aspect of the proposed architecture using the prototype. Fig 18 shows how the bandwidth of network flows evolves over time when a node in the *Hyacinth* prototype fails. In both the experiments, Node 6 was made to fail, and Node 3 failed over to Node 2 as its new parent. The period of time during which the network flows' bandwidth drops to zero represents the failure recovery time. The failure recovery time is between 600 to 700 msec.

Out of these 150 msec is the failure detection time, which was done by exchanging HELLO packets between Node 3 and Node 6. The propagation time for the route-change request is about 1 msec. Most of the remaining time goes into changing the routing tables.

For failover, Node 3 has to establish a new link-layer connection with Node 2. Due to driver implementation problem, the channel switching latency on the Windows platform is extremely high. To remove this overhead from the measurement, we kept an additional interface of Node 3 tuned on the same channel as Node 2. With an appropriate driver implementation, channel switching should only add about 50-100 msec to overall failover latency [34].

VII. Conclusion

Despite many technological advances at the physical layer, limited bandwidth remains a pressing issue for wireless networks, especially when compared with their wired counterpart. The bandwidth problem is even more serious for multi-hop wireless mesh networks (WMNs) due to interference between successive hops on the same path as well as that between neighboring paths. As a result, conventional single-channel WMNs cannot adequately support the bandwidth requirements of last-mile wireless broadband access networks, let alone a campus backbone that completely replaces the wired Ethernet. In this paper, we describe a novel multi-channel WMN architecture that effectively addresses this bandwidth problem by fully exploiting non-overlapped radio channels that the IEEE 802.11 standards make available.

In particular, this paper addresses two fundamental design issues in the proposed multi-channel WMN architecture. First, which of the available non-overlapped radio channels should be assigned to each 802.11 interface in the WMN? For two nodes to communicate with each other, their interfaces need to be assigned to a common channel. However, as more interfaces within an interference range are assigned to the same radio channel, the effective bandwidth available to each interface decreases. Therefore, a channel assignment algorithm needs to balance between maintaining network connectivity and increasing aggregate bandwidth. Second, how packets should be routed through a multi-channel wireless mesh network? The routing strategy in the network determines the load on each 802.11 interface, and in turn affects the bandwidth requirement and thus the channel assignment decision for each interface.

Through a detailed simulation study, we show that by deploying just 2 NICs per node, the distributed channel assignment/routing algorithm we developed for the proposed multi-channel WMN architecture can achieve a factor of 6 to 7 throughput improvement compared to the conventional single-channel WMN architecture. In addition, to demonstrate the feasibility of the proposed multi-channel WMN architecture, we successfully build a 9-node *Hyacinth* prototype that consists of PCs each equipped with two commodity 802.11a interfaces, and empirically show that it can indeed improve the aggregate throughput of multiple FTP sessions by a factor of 5. These results also convincingly prove that with proper channel assignment and routing algorithms the proposed multi-channel wireless mesh network architecture can become a serious contender for campus-wide wireless backbones.

Acknowledgement

This research is supported by NSF awards ACI-0234281, CCF-0342556, SCI-0401777, CNS-0410694 and CNS-0435373 as well as fundings from Computer Associates Inc., New York State Center of Advanced Technology in Sensors, National Institute of Standards and Technologies, Siemens, and Rether Networks Inc. We would like to thank Piyush Kumar (FSU) and Pradipta De for insightful discussions.

References

[1] K. Jain, J. Padhye, V. N. Padmanabhan, L. Qiu; "Impact of interference on multi-hop wireless network performance"; Proc. ACM MobiCom '03
[2] Y. Liu, E. Knightly; "Opportunistic Fair Scheduling over Multiple Wireless Channels"; Proc. of IEEE INFOCOM '03.
[3] J. So, N. Vaidya; "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver"; Proc. of MobiHOC 2004.
[4] A. Tzamaloukas, J. J. Garcia-Luna-Aceves; "A Receiver-Initiated Collision-Avoidance Protocol for Multi-channel Networks"; Infocom '01
[5] A. Nasipuri, S. Das; "A Multichannel CSMA MAC Protocol for Mobile Multihop Networks"; Proc. of IEEE WCNC '99.
[6] "IEEE 802.1D - MAC Bridges"; http://standards.ieee.org/getieee802/download/802.1D-1998.pdf
[7] N. Migas, W. J. Buchanan, K. A. Mc Aartney; "Mobile Agents for Routing, Topology Discovery, and Automatic Network Reconfiguration in Ad-Hoc Networks"; Proc. of the 10th IEEE ECBS '03.
[8] R. Chandra, C. Fetzer, and K. Hogstedt; "Adaptive Topology Discovery in Hybrid Wireless Networks"; Informatics '02.
[9] K. Claffy, H. Braun, G. Polyzos; "A parameterizable methodology for Internet traffic flow profiling"; IEEE JSAC '95.
[10] Y. J. Lin, and M. C. Chan; "A Scalable monitoring approach based on aggregation and refinement"; IEEE JSAC '02.
[11] A. Raniwala, T. Chiueh; "Evaluation of A Wireless Enterprise Backbone Network Architecture"; Proc. of the 12th Hot-Interconnects '04.
[12] A. Iwata, C. Chiang, G. Pei, M. Gerla, T. Chen. "Scalable Routing Strategies for Ad-hoc Wireless Networks."; IEEE JSAC '99
[13] E. Royer, C. Toh; "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks"; IEEE Personal Communications, April '99
[14] Mesh Networks Inc; http://www.meshnetworks.com
[15] Cowave Inc; http://www.cowave.com
[16] Radiant Networks; http://www.radiantnetworks.com
[17] R. Karrer, A. Sabharwal, E. Knightly; "Enabling Large-scale Wireless Broadband: The Case for TAPs"; Proc. of the HotNets '03
[18] P. Bhagwaty, B. Ramanz, D. Sanghi; "Turning 802.11 Inside-Out"; Proc. of HotNets '03.
[19] FireTide Inc; http://www.firetide.com
[20] Strix Networks; http://www.strixsystems.com
[21] R. Draves, J. Padhye, and B. Zill; "Routing in Multi-radio, Multi-hop Wireless Mesh Networks"; ACM MobiCom '04.
[22] P. Hsiao, A. Hwang, H. Kung, D. Vlah; "Load-Balancing Routing for Wireless Access Networks"; Proc. of IEEE INFOCOM '01.
[23] I. Katzela, M. Naghshineh; "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey"; IEEE Personal Communications (June '96)
[24] T. R. Jensen, B. Toft; "Graph Coloring Problems"; Wiley Interscience, New York, '95.
[25] Jie Gao, Li Zhang; "Load Balanced Short Path Routing in Wireless Networks"; Proc. of INFOCOM '04 .
[26] H. Hassanein, A. Zhou; "Routing with load balancing in wireless Ad hoc networks"; Proc. of ACM MSWiM '01.
[27] S.J. Lee and M. Gerla; "Dynamic Load-Aware Routing in Ad hoc Networks"; Proc. of ICC '01.
[28] A. Raniwala, K. Gopalan, T. Chiueh; "Centralized Channel Assignment and Routing Algorithms for Multi-channel Wireless Mesh Networks"; ACM Mobile Computing & Comm Review (MC2R), April '04.
[29] C. Villamizar, R. Chandra, R. Govindan; "Internet RFC 2439 - BGP Route Flap Damping"
[30] Tropos Networks; http://www.tropos.com
[31] Mesh Dynamics; http://www.meshdynamics.com
[32] BelAir Networks; http://www.belairnetworks.com
[33] J. Cao, et. al.; "Stochastic Models for Generating Synthetic HTTP Source Traffic" Proc. of Infocom '04.
[34] S. Sharma, N. Zhu, T. Chiueh; "Low-Latency Handoffs for Infrastructure mode Wireless LANs"; IEEE JSAC '04.