# Channel and Interface Management in a Heterogeneous Multi-Channel Multi-Radio Wireless Network*

***Technical Report (March 2009)***

Vartika Bhandari
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
vbhandar@uiuc.edu

Nitin H. Vaidya
Dept. of Electrical and Computer Eng., and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
nhv@uiuc.edu

## Abstract

As new application scenarios for multi-hop wireless networks emerge, there has been an effort to improve performance in these networks by leveraging available physical layer diversity in the form of multiple channels, radio-interfaces, antennas, etc. However, designing cross-layer protocols that are capable of addressing a wide range of heterogeneous hardware characteristics can be very challenging. Theoretical results on scheduling provide a valuable set of tools to understand the structure of good network control algorithms for diverse scenarios; but these usually involve highly idealized assumptions that hinder their applicability. In this paper, we present a Layer 2.5 protocol solution for multi-channel multi-radio wireless networks with heterogeneous channel and radio-interface characteristics, whose design draws upon known theoretical results, but which takes into account practical concerns that arise in real-world networks. This design provides a proof-of-concept of the possibility of evolving practical cross-layer designs that are rooted in sound theoretical principles.

## 1 Introduction

Many applications for multi-hop wireless networks have emerged in recent years, ranging from mesh networks to sensor networks. At the same time, there has been significant research activity on trying to utilize available resources in the form of multiple channels, multiple interfaces, and even multiple antennas to improve network performance. These are non-trivial issues, even in a network where all nodes have identical hardware configurations. However, the situation becomes much more complex when different nodes may be equipped with different number and type of hardware resources. Such scenarios are by no means hard to envision. In particular, given the easy off-the-shelf availability of various kinds of 802.11 cards, multi-channel multi-radio networks constitute an important class of potential scenarios involving heterogeneous hardware that can serve as a candidate instance for the broader issue.

*This research was supported in part by US Army Research Office grant W911NF-05-1-0246, and a Vodafone Graduate Fellowship.

Consider a community mesh network in which each participant may choose to equip his/her device with 802.11 cards of varying number and type (802.11a/b/g) based on his/her willingness or enthusiasm. Of course, it is always possible to stipulate a minimum baseline requirement that all nodes must adhere to, and design protocols for that baseline. However, by doing so one foregoes the potential for improved performance if the additional hardware resources were better exploited. Additionally, heterogeneity in configuration may sometimes arise from accident: in a multi-radio network with initially identical nodes, one or more of the interfaces at one/multiple nodes may fail leading to an effectively heterogeneous network till they are replaced. Once again, it is possible to stipulate that nodes with one or more failed interfaces are not allowed to participate in the network till replacement occurs. However, by doing so one would deny connectivity to a participant who may otherwise still have sufficient functional hardware for network access. But if the network uses protocols designed to handle heterogeneous number and type of interfaces, then those protocols would be able to adapt to either scenario, and make effective use of the available resources.

*How then should one operate a multi-channel multi-radio network where nodes may be equipped with varying number and type of interfaces, and the channels available for use may have vastly different characteristics (e.g., consider an 802.11b channel in the 2.4 GHz band and an 802.11a channel in the 5GHz band)?* Evidently, awareness of the physical layer characteristics is needed to properly utilize the resources available. But it is also not feasible to modify every layer of the protocol stack each time the hardware configuration changes. The issue is thus of designing cross-layer protocols that respect modularity as much as possible, and encapsulate physical layer awareness within a limited part of the network stack while interacting with other layers through generic interfaces, and being capable of operating in a wide range of scenarios.

However, when faced with an array of diverse hardware configurations, it becomes very difficult to design protocols that are able to function effectively over a wide range of scenarios. In other words, generalized protocol

designs capable of handling a range of physical layer diversity pose a significant challenge.

Theoretical results developed over the past decade or so provide a valuable set of tools to develop an understanding of factors affecting network performance, as well as the structure of good control policies. The seminal work of Tassiulas and Ephremides [1] characterized the throughput-optimal scheduling policy for a wireless network. This scheduler, usually referred to as the Dynamic Backpressure Scheduler, makes control decisions based on information about current queue-occupancies, and channel state information, and using knowledge of interference relationships. It is to be noted that this scheduler is optimal over a wide class of traffic arrival processes, channel state processes, and hardware configurations. However, it is difficult to implement due to the need for global information, as well as potential intractability of schedule computation. This has led to the emergence of a body of work on *imperfect* schedulers ([2, 3, 4], etc.) that trade-off performance for ease of implementation. But even these sub-optimal schedulers are not immediately amenable to practical implementation. The reasons are various. Firstly, the entire body of work previously referenced views throughput-optimality in a time-asymptotic sense; thus, very large delays may be incurred. Secondly, all nodes are assumed to have unbounded buffers, and no packets are ever dropped. Finally, scheduling is often faciliated through synchronized time-slots, and information from the vicinity (usually interference neighborhood) is assumed available as and when required.

These assumptions severely hamper the potential for practical system design based on these scheduling algorithms. Also problematic is the interaction of these scheduling policies with congestion control. Usually, these schedulers operate well with congestion-controllers specifically designed for the purpose. However, given the widespread use of UDP and TCP in the real world, and the infeasibility of replacing them with alternatives (even substantially better ones) in the near term, it is imperative that a practical approach to scheduling perform well with these transport layer protocols.

*In light of this, is it possible to draw upon the valuable insights provided by these theoretical results to design a cross-layer design capable of addressing a wide range of hardware characteristics, while inter-operating with existent transport layer protocols, and do so with limited information exchange between nodes?* In this paper, we describe an effort in this direction by presenting Layer 2-3 protocols for a heterogeneous multi-channel multi-radio wireless network. Our algorithms are informed and inspired by some of the previously discussed theoretical work, as well as other work on multi-channel networks and load-balancing [5, 6], but they also take into account constraints and concerns expected in real systems. Key features of our approach include a two-timescale methodology for assigning channels to interfaces, and a backpressure-based local scheduling algorithm. Our work provides a proof-of-concept of the possibility of developing generalized and modular cross-layer designs rooted in sound theoretical principles.

## 2 Objectives

This work targets multi-channel multi-radio wireless mesh networks. In keeping with this, we focus on static topologies, and our primary metric of interest is throughput.[1] Our objective in this work has been to design a Layer 2.5 link layer protocol that handles channel assignment and packet scheduling decisions in the presence of variable number and type of interfaces per node, as well as heterogeneous channel characteristics; it also adapts to traffic in the network.

## 3 Related Work

Protocols and architectures for multi-channel networks can be broadly categorized into those intended for single-radio devices, and those intended for multi-radio devices. In the case of single-radio devices, the channel coordination problem can be quite complex whereas, with multi-radio devices, the coordination issues are made somewhat easier to address by the presence of many radios.

Many protocols have been proposed for channel-coordination amongst devices having a single radio each. A useful taxonomy for these has been described in [7]. These include common-hopping-sequence based protocols such as CHMA [8], split-phase protocols such as MMAC [9], and rendezvous-based protocols such as SSCH [10], McMAC [11], Dominion [12], etc.

Recently, there has been much interest in protocols/architectures for multi-channel multi-radio networks. Of these, the Net-X testbed [13] is relevant to our work, as we adopt the node configuration used in Net-X.

Many protocols have been proposed to incorporate traffic awareness in various queueing and scheduling decisions, both for single and multi-channel scenarios. Neighborhood RED [14] proposes a variant of the RED algorithm, whereby queues at nodes within two hops are also taken into account, and not just the local queue. Warrier et al. have proposed a cross-layer architecture that is based on recent theoretical work on cross-layer optimization [15] Traffic-aware channel assignment in LANs has been considered in [16]. For LANs with uncoordinated access points, it has been proposed in [17], that channel-hopping can help prevent worst-case scenarios, and provide good average case performance. A centralized traffic-oblivious joint routing and scheduling scheme for mesh networks has been proposed in [18].

Draves et al [19] have considered the issue of routing in a multi-channel multi-radio mesh network where nodes are equipped with one radio each of type 802.11a and 802.11g. However, they do not consider the problem of channel selection.

The use of heterogeneous interfaces to handle route breakages has been proposed in [20].

[1] Delay may be more important than throughput for certain traffic classes, but that is not the focus of this paper, so long as the delay is not so large as to lead to potential starvation (e.g., in the case of TCP).

Joint channel assignment and routing in a heterogeneous multi-channel multi-radio wireless network has been considered in [21]. This work targets a situation very similar to what we have considered in this paper, and is closest in scope to our work. It allows for both heterogeneity in the operational abilities of interfaces, as well as in supported channel data-rates. It handles both single-radio, and multi-radio devices. A joint channel-assignment and routing scheme (JCAR) is proposed. However, this work treats the route for each flow as a sequence of interfaces, and therefore does not consider the possibility of link-layer data-striping. Moreover, it seeks a solution where interfaces switch channels only over substantially long periods of time.

The channel diversity in a multi-channel network provides opportunity for not merely load-balance but opportunistic selection of the channel with better channel quality. Opportunistic channel selection in a localized sense has been considered in MAC protocols such as DB-MCMAC [22], etc.

# 4 General Design Principles

We begin by briefly describing the general design principles on which we have based our work.

*A Route as a Sequence of Nodes*

A node-link is a pair of neighboring nodes. A radio link is a pair of radio-interfaces on neighboring nodes. Thus, a node-link comprises a set of radio-links, and with suitable link-layer strategies, one can exploit this diversity/multiplicity. ***We adopt an approach of single-path routing with link-layer data-striping***. Thus, a path from source to destination is a single sequence of nodes (and hence also a series of node-links).[2] When packets need to be transmitted over a node-link, the link layer determines which radio(s) and channel(s) to use. Thus, the link-layer can perform link-level data-striping if many radios are available at both transmitter and receiver. Moreover, this approach allows flexibility, as the link layer can make packet scheduling decisions at fine granularity.

*Channel Restriction*

Effectively exploiting available channel diversity requirees mechanisms to obtain information about channel quality, and can result in significant overhead, especially if the number of available channels is large. Moreover, in a distributed setting, when multiple entities act independently, opportunism can have an adverse effect on load-balance, e.g., consider a worst-case scenario where all nodes in a vicinity decide that channel $x$ has best quality and start using that channel simultaneously.

Typically, much of the benefit of opportunistic exploitation of channel diversity can be obtained by having the choice of a few channels. Thus, a reasonable solution lies in restricting the operation of a link to a subset of all possible channels available to it (a *channel pool*). One can then attempt to opportunistically exploit diversity amongst channels in this *channel pool*. Some prior work, e.g., [23], has studied this issue in a single-hop setting and concluded that a few channels indeed provide a good trade-off between diversity-gain and probing cost. The same conclusion is likely to hold even in multi-hop settings.

[2] As explained later, this implies that all interfaces of a node share the same IP address.

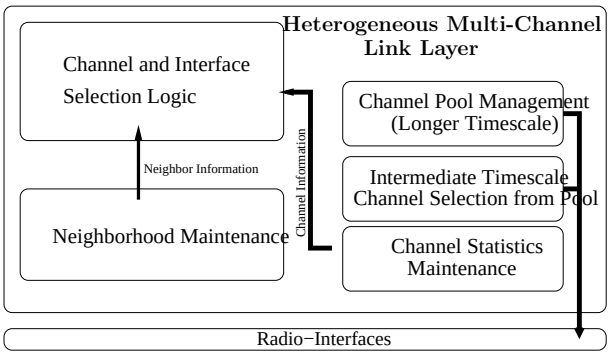


**Figure 1. General Architectural Template**

Moreover, channel-restriction has the potential to provide a degree of a priori load-balance (since different links will have different channel pools). This can help reduce the possibility of worst-case channel-selection scenarios like the one mentioned above, while still providing enough choices to each link for good load-balance. Some intuition for this can be derived from the capacity results proved in [5], as well as work on balls and bins with choices [6].

We propose the following simple channel restriction policy: each interface is assigned a small pool of $f$ channels for substantial periods of time. The channel pools are chosen and adjusted so that, within the two-hop neighborhood of any interface, each channel occurs in the pool of approximately the same number of interfaces (thus the criterion is *traffic-oblivious*). The current channel for each interface is selected more frequently using a traffic-aware criterion.

It is to be noted that ***the poolsize $f$ provides a control knob to tune the degree of dynamism of the protocol***. Setting $f = 1$ corresponds to a largely static channel assignment (where interfaces switch channels very infrequently), while setting $f = c$ corresponds to a fully dynamic assignment, in which the current channel may be chosen from the entire set of possible channels.

*Use of limited information from vicinity*

To limit overhead, explicit information exchange should not occur between nodes beyond 2 hops, even if conflicts extend beyond.

A high-level schematic of the envisioned framework incorporating the elements described above is depicted in Fig. 1.

# 5 Model

We assume a node configuration similar to the Net-X Project [13] where interfaces are classified as belonging to one of the following two categories:

1. *R-interface:* An R-interface is used for receiving packets, and whenever its channel is changed, the

change is advertised to neighbors. An R-interface is also used for transmitting packets that are to be sent on its current channel.

2. *T-interface:* A T-interface is used for transmitting packets. When a packet is to be transmitted to a next-hop node, a T-interface is switched to one of the R-channels of the next-hop node, and used to transmit the packet.

The interfaces can be of type: single-mode 802.11a, single-mode 802.11b and multi-mode 802.11ab. Dynamic rate adaptation is performed using a variant of AARF [24].

Each node is assumed to either have at least one R-interface and one T-interface of type $x$ or no interface of type $x$, where $x$ can be 802.11a or 802.11b. A multi-mode 802.11ab radio can be present as a T-interface, and can be counted towards each type, e.g., if a node has one R-interface each of type 802.11a and 802.11b, and a T-interface of type 802.11ab, then this is a valid configuration. Currently, we do not allow multi-mode R-interfaces.

Adopting this dual-radio framework helps avoid connectivity issues, and channel coordination problems such as multi-channel *deafness* [25], and enables us to focus on the scheduling aspects of the problem.

At each node, we have a single link-layer entity that manages all interfaces (which perform independent MAC procedures). Since we wish to perform single-path routing while allowing for the possibility of transparent link-layer striping, we require all interfaces of a node to have the same IP address. To avoid changing ARP, all interfaces of a node are also assigned the same MAC address.

Interfaces are assumed to be capable of fairly fast switching. More specifically, we consider that switching between channels in the same mode takes $250\mu s$ (this is consistent with channel switching times reported in recent work, e.g., [26]). If the channel-switch also requires a switch from 802.11a to 802.11b mode, the switching time is assumed to be $500\mu s$.

## 6 Interference and Interface Conflicts

An effective metric for channel assignment should be able to capture both interference and interface conflicts. We begin by illustrating these conflicts through examples in the context of the addressed model. In these examples, each node has one 802.11a R-interface and one 802.11a T-interface, and for ease of exposition, we assume that ideal scheduling is possible. A fixed 6 Mbps transmission rate is used.

**Example 1:** Consider the situation in Fig. 2. There are only two 802.11a channels available for use (let us denote them by 1 and 2). All links interfere with each other. Link $l_1$ has traffic-demand 6 Mbps, while links $l_2$ and $l_3$ have traffic demand 3 Mbps each. An ideal scheduler can meet these demands by having $l_1$ operate on channel 1 and $l_2$ and $l_3$ operate on channel 2. A traffic-unaware static distributed channel assignment strategy's best solution is to have two of these links on one channel, in a manner oblivious to actual load. Thus, it could potentially operate $l_1$ and $l_2$ on channel 1 and $l_3$ on channel 2, resulting in throughput degradation.

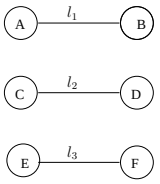


**Figure 2. Example 1: Interference Conflicts**

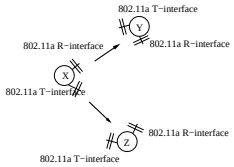


**Figure 3. Example 2: Interface Conflicts**

Now consider an example illustrating a potential interface conflict and how it can be resolved:

**Example 2:** Consider the situation in Fig. 3. There are 3 802.11a channels available for use. There are two flows: $X \rightarrow Y$ and $X \rightarrow Z$ with traffic demand 6 Mbps each. If the R-interfaces of all 3 nodes are on different channels, the maximum aggregate throughput possible is 6 Mbps. However, if the R-interface of either $Y$ or $Z$ is on the same channel as the R-interface of $X$, while the R-interface of the remaining node is on another channel, then both flows can get 6 Mbps, since $X$ can use its R-interface to transmit packets to one, and its T-interface to transmit packets to the other. A traffic-unaware strategy that only considers interference conflicts in a combinatorial sense (number of interfering interfaces on a channel) would not be adequate for this; in fact, such a strategy would typically try to assign different channels to all 3 R-interfaces.

## 7 Link Layer Protocol

The link layer protocol, which we term the Heterogeneous Multi-Channel Link Layer (HMCLL) Protocol, can be said to lie in Layer 2.5, i.e., between layers 2 and 3 in the protocol stack. The HMCLL is IP-aware. Despite extending above Layer 2 in its scope, we consider it appropropriate to term it a link layer (LL) protocol, since, in a wireless setting, an adaptive LL protocol must take into consideration the entire local network region whose events significantly affect or are affected by the operation of a "link", and its scope may need to extend above Layer 2 to do so effectively, which is the case here.

In this section, we summarize major protocol aspects. Some details have been omitted due to paucity of space.

### 7.1 Neighborhood and Channel/Traffic Statistics Maintenance

We begin by introducing some terminology.

The one-hop neighborhood of a node $u$ is denoted by $nbd(u)$, and its two-hop neighborhood is denoted by $nbd_2(u)$ ($u \notin nbd(u)$ and $u \notin nbd_2(u)$).

Each node $u$ has a set of active interfaces $\mathcal{M}(u) = M_R(u) \cup M_T(u)$, where $M_R(u)$ and $M_T(u)$ are the R-interfaces and T-interfaces respectively of node $u$. Let $\mathcal{C}(x)$ denote the set of channels on which interface $x$ is capable of operating. Each interface has a type denoted by $type(x)$ which uniquely determines the set of channels $\mathcal{C}(x)$ on which $x$ can operate[3]. Each R-interface $x$ has

[3]For instance, we currently consider three types: 802.11a, 802.11b, and 802.11ab. Of these, only 802.11a and 802.11b are

an associated subset of channels called the channel-pool $\mathcal{P}(x) \subseteq \mathcal{C}(x)$ such that $|\mathcal{P}(x)| = f$. The current channel of interface $x$ is denoted by $c(x)$. We use the notation $c(\mathcal{S})$ where $\mathcal{S}$ is a set to denote $\bigcup_{x \in \mathcal{S}} \{c(x)\}$.[4]

The link layer maintains lists of one-hop and two-hop neighbors. One-hop neighbors are qualified as being symmetric or asymmetric.

A node $u$ maintains a number of statistics. These include: the average contention time experiences when transmitting on channel $c$ (denoted by $\kappa(u,c)$ and maintained as an EWMA), the average length of packets sent to a neighbor $v$ (denoted by $l(u,v)$), succes-rate when transmitting to a neighbor $u$ on channel $c$ (denoted by $x(u,v,c)$), effective transmission rate when sending to $v$ on channel $c$ ($r(u,v,c)$), estimate of interface TX-utilization for each local interface $x$, denoted by $\rho(x)$ computed over intervals of duration $T_{rassign}$, and an average $\overline{\rho}(x)$ maintained as an EWMA. .

All rate estimates above are in units of bits per second.

Neighborhood management, as well as channel and traffic statistics maintenance are facilitated by exchange of link layer control packets.

For each $v \in nbd(u)$, $u$ maintains a set $\mathcal{T}(u,v) \subseteq M_R(v)$, which is the set of R-interfaces of $v$ that $u$ would be willing to send packets to. The choice of $\mathcal{T}(u,v)$ can be used to allow/disallow link-layer data-striping (e.g., if $|M_R(v)| > 1$ but $|\mathcal{T}(u,v)| = 1$, then this corresponds to no data striping). Currently, we use $\mathcal{T}(u,v) = M_R(v)$. However, in the rest of the description, we will continue to use the term $\mathcal{T}(u,v)$ to highlight that the link layer algorithms can work for other choices of $\mathcal{T}(u,v)$ (of course, in that case, an additional algorithm will be needed to select $\mathcal{T}(u,v)$).

The link layer also maintains a system of queues (described later). These include a queue of outgoing packets to each next-hop neighbor. The length of the queue (in bits) for neighbor $v$ at node $u$ is denoted by $q_{nbr}(u,v)$. There is also a queue for each channel. The length of the queue (in packets) for channel $c$ is denoted by $q^p_{ch}(u,c)$.

***Since our goal is to address channels/interfaces with heterogeneous characteristics, we rely on computing various numerical quantities associated with each channel/interface that essentially provide an abstract representation of its characteristics, and help evaluate the desirability of using it.***

We present the definitions of these quantities and discuss how they capture important characteristics/properties:

The minimum-rate constant $\theta$ is a constant used to avoid division-by-zero anomalies. It is chosen to be much smaller than the typical rate values (currently $\theta = 1$).

The net datarate to a neighbor $v$ on channel $c$ is computed when needed as:

$$\mu(u,v,c) = \frac{l(u,v)}{\frac{l(u,v)}{\max\{\theta, r(u,v,c)\}} + \kappa(u,c)\left(\frac{1}{\max\{10^{-3}, x(u,v,c)\}}\right)}.$$

The ratesum for a link $(u,v)$ is denoted by $\sigma(u,v)$ and defined as $\sigma(u,v) = \sum_{y \in \mathcal{T}(u,v)} r(u,v,c(y))$. Intuitively, the significance of the ratesum is that the LL needs to estimate the load on each channel in the near future. To do so, it pretends that each neighbor $v$ splits traffic it sends to $u$ across channels in $\mathcal{T}(v,u)$ in proportion to the channel-rates, and therefore, the ratesum plays a role in computing various estimates, as will be evident ($v$ may not necessarily split traffic in this manner, but it serves as a reasonable hint for LL decisions).

The link-layer at $u$ tracks the number of bits sent to $v$ over intervals of duration $T_{rassign}$, denoted by $s(u,v)$. Average sent bits for link $(u,v)$ are denoted by $\overline{s}(u,v)$, and maintained as an EWMA.

Interface-conflict cost for channel $c$ over link $(u,v)$ is defined as follows (in the following text $K$ is a suitably chosen threshold constant):

If $q_{nbr}(u,v) < K$ then $\chi(u,v,c) = 0$

If $q_{nbr}(u,v) >= K$ : If $c$ is an R-channel of $u$, i.e., there is $x \in M_R(u)$ such that $c(x) = c$, then it is defined as:

$$\chi(u,v,c) = \sum_{w \in nbd(u)} \left( \frac{q_{nbr}(u,w)}{\max\{\sigma(u,w), \theta\}} + T_{rassign}(\rho(x) - 0.8)_+ \right) Y$$

where $Y = I_{(\exists\, y \in \mathcal{T}(u,w) : c(y) = c)}$

If $c$ is not an R-channel, let $\mathcal{S}(b) \subseteq M_T(u)$, be the set of T-interfaces of $u$ that can operate on a channel $b$. Then:

$$\chi(u,v,c) = h(u,v,c) + U(u,c) I_{h(u,v,c) > H}$$

where

$$h(u,v,c) = \frac{1}{|\mathcal{S}(c)|} \sum_{x \in \mathcal{S}(c)} \left( \sum_{w \in nbd(u)} \sum_{\substack{y \in \mathcal{T}(u,w) \\ c(y) \notin c(M_R(u)) \\ \wedge c(y) \in \mathcal{C}(x)}} \frac{q_{nbr}(u,w)}{\max\{\sigma(u,w), \theta\} |\mathcal{S}(c(y))|} \right)$$

$$U(u,c) = \frac{T_{rassign}}{|\mathcal{S}(c)|} \sum_{x \in \mathcal{S}} (\rho(x) - 0.8)_+$$

and $H$ is a suitably chosen threshold

Some intuition for the relevance of this quantity is that it provides an estimated measure of the amount of traffic (normalized by rate) that contends for interface time at sending neighbor $v$ on the interface(s) that are used to send packets on channel $c$. The utilization-based component is included primarily because when we have TCP traffic, the queues may never become large enough to trigger a change in channel assignment.

The local interface conflict seen by channel $c$ at node $u$ is denoted by $\chi_{local}(u,c)$ and defined as:

1. If $c$ is the current channel of a local R-interface, $\chi_{local}(u,c) = 0$.

valid types for R-interfaces.

[4] An interface is said to be active if it has not been deactivated by the LL (this might happen if number of interfaces exceeds number of channels).

2. If $c$ is not an R-channel:

$$\chi_{local}(u,c) = \frac{1}{|\mathcal{S}(c)|} \sum_{x \in \mathcal{S}(c)} \sum_{\substack{d \in \mathcal{C}(x) \\ d \neq c \wedge d \notin c(M_R(u))}} \left\lceil \frac{q^p_{ch}(u,d)}{|\mathcal{S}(d)|} \right\rceil \quad (1)$$

where $\mathcal{S}(b)$ denotes the set of T-interfaces at the local node $u$ that can operate on channel $b$.

The intuition behind $\chi_{local}(u,c)$ is that it provides a quantification of the conflict faced by packets bound to channel $c$ from packets bound to channels that compete with $c$ for local interfaces.

Total incoming data score for interface $x \in M_R(u)$ with respect to channel $b$ is denoted by $Incoming(x,b)$ and defined as:

$$\sum_{v \in nbd(u)} \left( \frac{s(v,u) + q_{nbr}(v,u)}{\max\{[\sigma(v,u) - r(v,u,c(x)) + r(v,u,b)], \theta\}} \right)$$

Incoming queue score for an R-interface $x$ at node $u$ is defined as:

$$\eta(x) = \sum_{v \in nbd(u)} \frac{q_{nbr}(v,u)}{\max\{\sigma(v,u), \theta\}}$$

$\eta(x)$ provides an estimated measure of the amount of traffic queued at neighbors of $u$ that is expected be sent to interface $x$.

### 7.1.1 Link Layer Control Packets

The link layer sends/receives the following control packets:

1. **LLINFO:** This packet is broadcast by each node $u$. The contents of an $LLINFO(u)$ packet are as follows: Sequence number, number of active R-interfaces, for each active R-interface $x \in M_R(u)$: $ID(x), type(x), |\mathcal{P}(x)|, c(x), \{b | b \in \mathcal{P}(x)\}, \eta(x)$, for each $v \in nbd(u)$: $seqno, \forall y \in M_R(v) : \{ID(y), type(y), |\mathcal{P}(y)|, c(y), \{b | b \in \mathcal{P}(y)\}, \eta(y)\}$.
2. **QINFO:** A QINFO($u \rightarrow v$) packet is unicast by each node $u$ to some or all neighbors in situations where the number of channels is greater than 1 and the poolsize is also greater than 1. The QINFO sending routine is invoked after intervals of duration $T_{QINFO} + X$, where $X$ is a random variable uniformly distributed in $[0, J_{QINFO}]$. This packet contains the following information:

   Length of outgoing queue to neighbor: $q_{nbr}(u,v)$ and recently sent data $s(u,v)$, number of active R-interfaces at $v$ known to $u$ (this will be $|M_R(v)|$ unless $u$ has wrong information about $v$), for each R-interface $y \in M_R(v)$: $|\mathcal{P}(y)|$, $c(y)$, $\forall\ c \in \mathcal{P}(y)$: $r(u,v,c), \kappa(u,c), \chi(u,v,c)$.
3. **CINFO:** A CINFO($u \rightarrow v$) is sent by $u$ to $v \in nbd(u)$ if $u$ receives a QINFO from neighbor $v$ containing incorrect information about $u$'s interfaces. The contents of a $CINFO(u)$ packet are as follows: Sequence number, number of R-interfaces of $u$, for each R-interface $x \in M_R(u)$: $ID(x), type(x), |\mathcal{P}(x)|, c(x), \{b | b \in \mathcal{P}(x)\}, \eta(x)$.
4. **PROBE:** A probe packet is a broadcast packet which is periodically sent with the sole purpose of estimating contention on each channel. This packet does not contain any information.

The sequence numbers for the LLINFO and CINFO packets are drawn from the same 32-bit sequence number space, and the sequence number is incremented after each packet is sent. QINFO and PROBE packets have no sequence number.

The link layer (LL) at node $u$ updates its locally maintained neighborhood information on receipt of control packets, which are also used to assess reachability on different bands. Details are omitted due to space constraints.

## 7.2 Interface Management

As has been described earlier, interfaces are classified as being either R-interfaces, or T-interfaces.

Except for LL control packets, packets received on a T-interface are discarded by the LL, to avoid the possibility of receiving duplicate packets. However, link layer control packets are processed in the same way as packets received on an R-interface. This helps provide resilience to loss of control packets sent on the R-interface's channel. It does not affect correctness as the operations performed on receipt of a control packet are idempotent. The sequence numbers make negligible chances of stale information overwriting fresher one.

### 7.2.1 R-Interface Management

Following the channel restriction approach we described in Section 4, we associate with each interface a pool of channels, from which the current channel is dynamically selected. Thus, the R-interface management has two aspects, viz., channel pool management, and R-channel selection. We now describe each of these.

#### 7.2.1.1 Channel Pool Management

Recall that $\mathcal{C}(x)$ denotes the set of channels on which interface $x$ is capable of operating, each R-interface $x$ has an associated channel-pool $\mathcal{P}(x) \subseteq \mathcal{C}(x)$ such that $|\mathcal{P}(x)| = f$, and the current channel of an interface $x$ is denoted by $c(x)$.

In keeping with the objective of a priori load-balance, it is desirable that the channels be equitably distributed across pools, such that in any vicinity all channels occur in roughly the same number of pools.

We use a probabilistic mechanism for pool management.

At the time of starting up, each interface is assigned a set of $f$ channels chosen uniformly at random from all such possible $f$-subsets. Progressively, as LLINFO packets are received from neighboring nodes, the Neighbor Table gets populated with information about the channel-pools of the R-interfaces of these nodes. The Channel Pool Manager uses a timer that is scheduled at start-up after an interval uniformly distributed between 0 and $T_{pool}$ seconds, and thereafter rescheduled every $T_{pool}$ seconds.

On each timer expiration, a pool-management algorithm is run by interface $x$. A succint description of the algorithm is as follows: for each valid channel, compute $n(c)$ as the number of R-interfaces within 2-hops

(including $x$) in whose pool $c$ occurs, and $\overline{n}$ as the average over all valid channels. $c_{max}$ is the pool-member with largest value of $n(c)$, while $c_{min}$ is the channel in $\mathcal{C}(x) \setminus \mathcal{P}(x)$ with smallest value of $n(c)$. If $n(c_{min}) < \overline{n}$ and $n(c_{max}) > \overline{n}$ and $n(c_{max}) > n(c_{min} - 1$, then compute $p = \frac{\overline{n} - n(c_{min})}{m}$ where $m$ is the number of interfaces within 2-hops for which $c_{min}$ is a valid channel. If $c_{max}$ is the current channel, divide $p$ by 2. With probability $\min\{1, p\}$ replace pool member $c_{max}$ with $c_{min}$.

We remark that our algorithm for pool-management bears similarity to the algorithm for minimum conflict coloring in [27]), and the algorithm for channel assignment in Net-X [13]. Also related is the probabilistic distributed learning algorithm for channel assignment described in [28].

Ideally, we would like the pool membership to stabilize after a brief period of churn, with further changes occurring rarely. However, due to the distributed and probabilistic nature of the algorithm, the channel pool membership can exhibit quasi-stable behavior; this does not pose any serious concern.

#### *7.2.1.2 R-Channel Selection*

The R-channel selection algorithm is designed on the premise that all selection decisions are sequential and staggered at different nodes. To reduce the chance of inadvertant synchronization, the protocol incorporate an element of random jitter in the assignment-interval. Thus, each interface has a R-channel re-assignment timer that is rescheduled over duration $T_{rassign} + X$, where $X$ is a random variable uniformly distibuted over $[0, J_{rassign}]$.

The channel cost metric for channel $b$ computed for interface $x$ of node $u$ has four components:

1. Explicitly known interference conflict cost:

$$C_{einc}(x,b) = \frac{1}{T_{rassign}} \sum_{v \in nbd_2(u)} \sum_{\substack{y \in M_R(v) \\ c(y)=b}} \eta(y) \qquad (2)$$

2. Interface conflict cost

$$C_{ifc}(x,b) = \frac{1}{T_{rassign}} \sum_{\substack{v \in nbd(u) \\ q_{nbr}(v,u)>0}} (\chi(v,u,b) - D(v,u,b,x))_+ \qquad (3)$$

where $D(v,u,b,x) = \frac{q_{nbr}(v,u)}{\max\{\sigma(v,u) - r(v,u,c(x)) + r(v,u,b), \theta\}}$ if $c(x) = b$ or if $[(c(x) \notin c(M_R(v))) \wedge (b \notin c(M_R(v)))]$, and is 0 else.

The intuition behind subtracting $D(v,u,b,x)$ from $\chi(v,u,b)$ is that the latter may sometimes include traffic intended for interface $x$. This should not be counted as is as a cost as is, as even after a channel switch, one might typically expect the same amount of traffic (in bits) to be re-directed to whatever new channel $x$ may switch to (although rate difference between the channels should be considered).

3. Contention cost, which helps capture interference beyond the two hop neighborhood which is not captured by the explicit interference cost, and also captures interference conflicts not reflected in queue-lengths (note that $\alpha$ below is a suitably chosen constant):

Let $w_v = q_{nbr}(v,u) + s(v,u)$

$$C_{iinc}(x,b) = \begin{cases} \frac{\alpha}{T_{rassign}} \left( \frac{1}{\sum_{v \in nbd(u)} w_v} \sum_{v \in nbd(u)} w_v \kappa(v,b) \right) & \text{if } \sum_{v \in nbd(u)} w_v > 0 \\ 0 & \text{else} \end{cases} \qquad (4)$$

4. Expected cost of traffic incoming to itself:

$$C_{self}(x,b) = \frac{Incoming(x,b)}{T_{rassign}} \qquad (5)$$

The cost of a channel $b$, as computed by R-interface $x$ of node $u$ is given by:

$$Cost(x,b) = C_{einc}(x,b) + C_{ifc}(x,b) + C_{iinc}(x,b) + C_{self}(x,b) \qquad (6)$$

The R-channel is selected using a procedure whereby the channel with least cost is determined, and if it yields a substantial improvement over the current R-channel, a switch is initiated. We remark that the hysteresis is important to avoid unstable behavior.

## 7.3 Packet Scheduling: Channel and Interface Binding

The channel and interface selection decisions are decomposed into two separate decisions, viz., channel selection, and interface selection, which are coupled through the channel queue occupancies, and the local interface conflict score $\chi_{local}$ defined earlier.

The structure of the packet scheduling component is depicted in Fig. 4. The channel binding decision is performed by a channel scheduler (denoted by CH-scheduler), and the interface binding decision is performed by an interface scheduler (denoted by IF-scheduler).

The link-layer at each node $u$ maintains the following system of queues:

1. **Neighbor Queues:** Each outgoing unicast packet has a next-hop $v \in nbd(u)$, and is enqueued in the queue corresponding to the appropriate neighbor $v$. The queue at node $u$ for neighbor $v$ is denoted by $Q_{nbr}(u,v)$, while the length of this queue in bits is denoted by $q_{nbr}(u,v)$, and the length in packets is denoted by $q^p_{nbr}(u,v)$.

2. **Channel Queues:** There is a pair of queues for each channel $c$ such that some interface of $u$ can tune to $c$. These contain packets that have already been bound to channel $c$ (i.e., these packets will be sent on channel $c$). The first of these is meant to temporarily hold high-priority packets (LL control packets, ARP packets and routing packets). We shall refer to this as the *high priority holding buffer* for the channel. All other packets are enqueued in the second queue. We shall refer to this as the *channel queue*, and denote this queue for channel $c$ at node $u$ by $Q_{ch}(u,c)$,

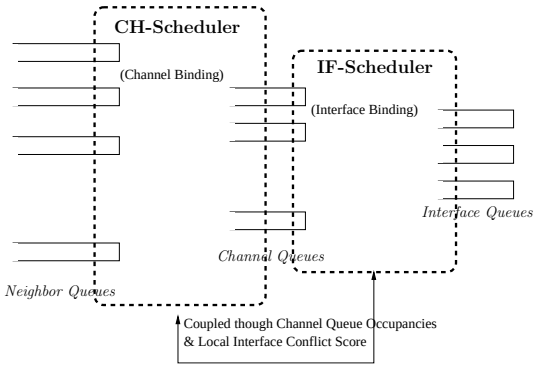


**Figure 4. Structure of Scheduling Module**

with the length in bits denoted by $q_{ch}(u,c)$. The length in packets is denoted by $q^p_{ch}(u,c)$.

3. **Interface Queues:** There is a queue for each interface $x$, containing packets that have already been bound to the interface $x$, and are awaiting their turn for transmission by interface $x$. The queue for an interface $x$ is denoted by $Q_{if}(x)$ and the queue-length is denoted by $q_{if}(x)$.

### 7.3.1 Handling Multi-Channel Broadcast

Currently, we adopt a very simple approach to broadcast. The node $v$ sends a copy of each broadcast packet on all channels that can be operated on by at least one of its interfaces.

### 7.3.2 High Priority Packets

Broadcast packets have higher priority than unicast packets. Whenever the link layer receives a broadcast packet for sending, it creates a copy of this packet for each channel and enqueues it in the high-priority holding buffer of that channel.

High-priority unicast packets are handled as follows: if the next-hop node for the packet is $v$, the packet is enqueued in the high priority holding buffer of the channel with highest effective rate that can be used to reach that neighbor. Link-layer and routing packets also have high priority.

The CH-scheduler determines how regular packets will be transferred from the Neighbor Queues to the Channel Queues, while the IF-scheduler determines how packets will be transferred from the Channel Queues to the Interface Queues.

### 7.3.3 Channel Binding

Consider the set of all eligible neighbor-queues at node $u$. Each has a certain next-hop node (MAC destination) $v$ for which there is a set of valid interfaces $\mathcal{T}(u,v) \subseteq M_R(v)$, and correspondingly a set of possible channels $\mathcal{T}_c(u,v) = \{c(y) | y \in \mathcal{T}(u,v)\}$.

Since the channel-assignment has already attempted to factor in the traffic-awareness, it is now reasonable to treat the link-layer packet scheduling problem as an independent local decision. From the perspective of the link-layer at node $u$, each packet enqueued in the set of neighbor-queues has a next-hop node from amongst $u$'s neighbors to which it has to send the packet. Thus, the link-layer treats the local packet scheduling problem as if it were a problem involving single-hop flows.

**We draw intuition from the Dynamic Backpressure Scheduler of Tassiulas and Ephremides [1]**. In a scenario where all flows traverse only a single-hop, a scheduler which activates links in a manner than maximizes $\sum q_l r_l$ is throughput-optimal (assuming the traffic load falls within the network's stability region). In our scheduling scenario, we can treat each valid (neighbor, channel) pair as a link, and define a conflict between two pairs if they have the same channel. Trying to map the algorithm of [1] directly, one might consider trying to assign packets from various eligible queues to channels, such that the assignment maximizes $\sum q_p \mu_p$, where $q_p$ is the length of the neighbor-queue from which the packet $p$ is taken, and $\mu_p$ is the net datarate of the link-channel pair over which $p$ is scheduled.

However, in practice, this can lead to long delays and possible starvation for some flows (especially if other flows are aggressive and inelastic). Additionally, some overhead amortization is desirable.

At each invocation of CH-scheduler at node $u$, do the following: Initially consider the set of valid (neighbor, channel) pairs $\mathcal{S} = \bigcup_{v \in nbd(u)} (\{v\} \times \mathcal{T}_c(u,v))$. Eliminate from $\mathcal{S}$ the pairs $(v,c)$ for which $q^p_{ch}(u,c) > CQ\_THRESH$ or $\chi_{local}(u,c) > CQ\_THRESH$ or $\mu(u,v,c) = 0$.

Let $Age(v)$ denote the age of the HOL packet in $q_{nbr}(u,v)$. For each remaining (neighbor, channel) pair $(v,c) \in \mathcal{S}$, compute a weight $w(v,c) = Age(v)\mu(u,v,c)$ and the rate-value $r'(v,c) = \mu(u,v,c)$. Select the pair with largest weight (with $r'(v,c)$ used to break ties), and transfer up to QUANTUM number of packets from $q_{nbr}(u,v)$ to $q_{ch}(u,c)$. Eliminate all other pairs with channel $c$. Since $\chi_{local}$ for some channels may have changed after packet-transfer, re-check to eliminate all pairs with $\chi_{local}(u,c) > CQ\_THRESH$. Repeat till no pairs remain.

***The key observation is that instead of queue-length, we use the age of the HOL packet. This gives priority to packets that have been waiting longer, and thus improves fairness characteristics. At the same time, it does not completely deviate from the intuition behind the throughput-optimal dynamic backpressure scheduler described in [1], since queue-length and HOL packet's age are usually strongly correlated in a FIFO queue.***

### 7.3.4 Interface Binding

Interface-binding for regular packets involves a greedy approximation to maximum weight matching on a bipartite graph of valid (channel, interface) queue pairs. The weight of a pair is the age of the HOL packet in the channel-queue, with a tie-breaking criterion based on switching-time considerations. Details are omitted due to space considerations.

### 7.3.5 Interface Queues

Once a packet has been transferred to an interface queue, the link layer relinquishes control over it (except

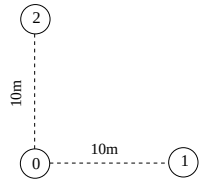


**Figure 5. Topology 6**

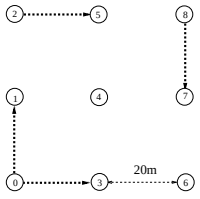


**Figure 6. Topology 5**

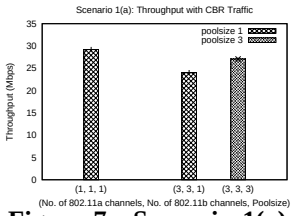


**Figure 7. Scenario 1(a): CBR Traffic**

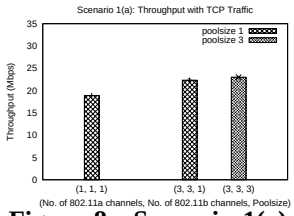


**Figure 8. Scenario 1(a): TCP Traffic**

for possibly triggering a flushing of packets from the interface-queue in case of a channel-switch).

## 8 Evaluation

The ns-2 simulator (version 2.31) [29] has been used as the codebase, with substantial modifications to the physical layer and node models. The transmission power is set at 65mW. A SINR threshold based model is used, and cumulative interference has been modeled. Propagation is modeled via the shadowing model with a path-loss exponent of 3.5 and a shadowing deviation of 2 dB. RTS?CTS is disabled. Physical carrier-sense is performed. As stated earlier, dynamic rate adaptation is performed using a variant of adaptive ARF. However, the 802.11 ACKs are sent at the respective base-rates. A data-payload of 1024 bytes is used for all reported simulations. To aid comparison of presented results with the best-possible, Table 1 lists the throughput achieved by UDP/TCP in a single-channel/single-link situation with sender-receiver separation *d* using 802.11a and 802.11b respectively, for our choice of payload-size, and rate-selection algorithm (averaged over 30 runs).

All plotted points are the average over 30 runs, and 95% confidence intervals are plotted. The *next-substream* feature of the ns-2 random number generator was used to assure independence of runs. TCP simulations use the ns-2 TCP-Sack1 agent, and comprise FTP traffic.

Often wireless simulation results are obtained via simulating over long time intervals. However, in many wireless scenarios, traffic patterns may not persist very long. ***To determine how effective an adaptive protocol might be, it is important to gauge its ability to adapt swiftly to traffic over short time intervals***. In keeping with this view, the maximum data-session length in our simulations is chosen to be only 10s.

*Scenario 1:*

This scenario (Fig. 5) helps illustrate *how the link-layer schedules and stripes packets in the presence of heterogeneous radios/channels*. Nodes 0 and 1 have one 802.11a R-interface, one 802.11b R-interface, and 1 802.11ab T-interface. 2 has one 802.11b R-interface and 1 802.11b T-interface. Two traffic scenarios are considered, viz., 1(a): single-flow $0 \rightarrow 1$ (CBR $\sim$ 32.77 Mbps, and FTP), and 1 (b): two flows $0 \rightarrow 1$ (CBR $\sim$ 23.4 Mbps, and FTP) and $0 \rightarrow 2$ (CBR $\sim$ 5.46 Mbps, and FTP). Three (802.11a channels, 802.11b channels, poolsize) combinations are evaluated: $(1,1,1), (3,3,1), (3,3,3)$.

With $(1,1,1)$, the T-interfaces are deactivated by the LL, and nodes 0 and 1 have 1 R-interface each on each of the channels, while node 2 has one R-interface on the single 802.11b channel. Thus, $(1,1,1)$ provides excellent opportunity for data-striping between 0 and 1.

When node 0 has a single multi-mode T-interface, and there is a single flow $0 \rightarrow 1$, $(3,3,1)$ yields lower throughput than the other two combinations with CBR traffic (Fig. 7). This is because with $(3,3,1)$, the R-interfaces are more likely to be on different channels, and thus node 0 can only use its multi-mode T-interface to send data (note from throughput value that data gets primarily sent on the 802.11a channel, as is desirable). In case of $(1,1,1)$, we get near-best performance due to data-striping across 2 channels. With $(3,3,3)$, the network is likely to initially have all R-interfaces on different channels, but is able to quickly adapt based on the interface conflict cost, and get the benefit of data-striping across two interfaces/channels.

TCP throughput (Fig. 8) is lower than CBR traffic, and the difference between $(3,3,1)$ and $(3,3,3)$ is not very marked, probably because it is more difficult for the latter to adapt to interface-conflicts with TCP (as queues never become too large). $(1,1,1)$ has lowest throughput. This can be explained by the fact that DATA/ACK always contend for the same two interfaces/channels in this case and the resultant delay-increase has a detrimental effect; TCP is also likely to get somewhat lesser benefit from data-striping due to out-of-order delivery issues.

When there are two flows $0 \rightarrow 1$ and $0 \rightarrow 2$, $(3,3,1)$ again exhibits much lower performance (this time for both CBR (Fig. 9) and TCP (Fig. 10)), as there is a smaller chance of R-channel overlap, and thus, node 0 must typically time-share its T-interface to send to node 1 and node 2. The other multi-channel combinations benefit from the R-interfaces, as already explained above. Note that despite having to contend for the same interface in $(3,3,1)$, the two flows each get reasonable throughput. Of course, the throughput for destination 2 is lower, since the packet-scheduler tries to achieve a balance between providing fairness and getting the best rate (though the two flows do get a reasonably fair share of *interface time*). If greater "throughput fairness" is desired, the scheduling rules can be suitably modified to achieve that.

*Scenario 2:*

The scenario is depicted in Fig. 6. 9 nodes are arranged in a 3 by 3 grid (of side 20m). Each node has one R-Interface and one T-interface of type 802.11a. There are 3 CBR flows: $0 \rightarrow 1$ at rate $\sim$ 13.65 Mbps starts at $t = 40.0s$, $0 \rightarrow 3$ at rate $\sim$ 13.65 Mbps starts at $t = 40.5s$, $2 \rightarrow 5$ at rate $\sim$ 6.83 Mbps starts at $t = 40.6s$, $8 \rightarrow 7$ at

| PHY-Traffic Type | 10m | 20m | 30m | 40m | 50m | 60m | 70m | 90m | 110m | 130m |
|---|---|---|---|---|---|---|---|---|---|---|
| 802.11a-CBR | 24.28 | 11.75 | 9.5 | 7.03 | 0.51 | 0.002 | 0.0 | 0.0 | 0.0 | 0.0 |
| 802.11b-CBR | 4.95 | 4.95 | 4.95 | 4.95 | 4.95 | 4.93 | 4.63 | 2.0 | 0.32 | 0.046 |
| 802.11a-TCP | 17.1 | 8.16 | 6.89 | 5.62 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 802.11b-TCP | 3.52 | 3.52 | 3.52 | 3.52 | 3.52 | 3.51 | 3.31 | 1.58 | 0.28 | 0.005 |

**Table 1. Single-channel single-link throughput values (Mbps)**

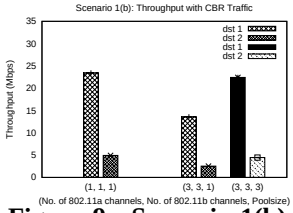


**Figure 9. Scenario 1(b): CBR Traffic**

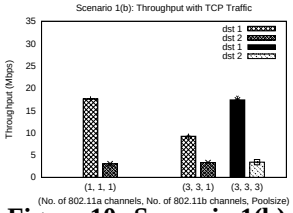


**Figure 10. Scenario 1(b): TCP Traffic**

rate $\sim$ 6.83 Mbps starts at $t = 40.9s$. All flows run till end of simulation at $t = 50.0s$. *The topology is of interest as it involves both interface and interference conflicts (recall examples discussed in Section 6).* An ideal scheduler can meet almost all the traffic demand with just 3 channels, by assigning one channel to the R-interface of 0 and either of 1 or 3, assigning the second channel to the remaining node from amongst $1,3$, and assigning the third channel to 5 and 7. The following (number of channels, poolsize) combinations are evaluated: $(1,1),(3,1),(12,1),(3,3),(12,3),(12,12)$.

Per-flow throughput is depicted in Fig. 11. A poolsize of 3 typically yields better performance than a poolsize of 1 for same number of channels. With 12 channels and poolsize 3, the throughput is lower than the throughput with 3 channels and poolsize 3. The reason for this is that there is an interface-conflict that arises at node 0, as it has only one T-interface but is generating data for both 1 and 3 at $\approx$ 13.65 Mbps each. Hence it is desirable to have the R-interface of 0 and one of 1 and 3 on the same channel (so that 0 can use its R-interface for transmission), while the T-interface is used to transmit packets to the remaining node on another channel. The interface-conflict component of the channel cost metric does try to capture this; however, the receiver's R-interface cannot change its assignment to address interface conflicts if the transmitter's R-channel is not in the pool of the receiver's R-channel. This leads to the observed inversion scenario. It can potentially be addressed by additional signaling leading to pool-adjustment, but the extra complexity may not be justified if such scenarios are not very common. Our justification that the inversion phenomenon is being caused by channel-restriction is borne out by the fact that with $(12,12)$, the throughput is better than with $(3,3)$.

The key observation is that $(12,12)$ provides close-to-best-possible performance, and $(3,3)$ also comes fairly close despite having no surplus channels.

*Scenario 3:*

9 nodes are arranged in a 3 by 3 grid (Fig. 12). Node 4 has 4 R-interfaces of type 802.11a, and one T-

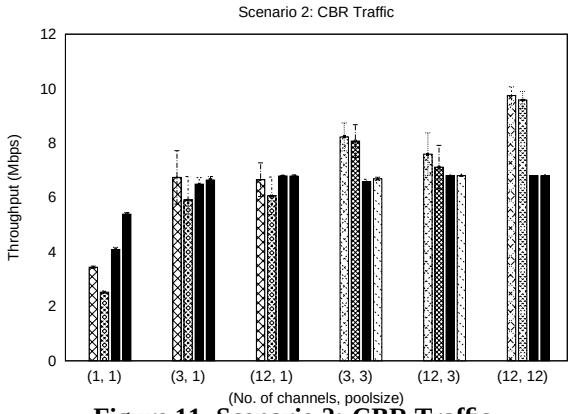


**Figure 11. Scenario 2: CBR Traffic**

interface of type 802.11a (this could be a server or gateway node). All other nodes have one R-interface and one T-interface of type 802.11a. There are 4 one-hop flows $1 \rightarrow 4, 3 \rightarrow 4, 5 \rightarrow 4, 7 \rightarrow 4$ which start at times $t = 40.0s, 40.5s, 41.0s, 41.5s$ respectively, and continue till $t = 50.0s$. In the CBR case, each flow has traffic rate $\sim$ 11.7 Mbps. *An ideal scheduler would assign each sender one of node 4's R-interfaces to transmit to, avoiding any inter-flow contention. In our protocol, the CH-scheduler at each node makes its own decision; we evaluate how it compares with the ideal.* Fig. 13 shows the aggregate throughput for CBR. From Table 1 and throughput for $(1,1)$, one would expect best-possible value to be around 40 Mbps. All multi-channel combinations perform well; $(4,1),(4,3)$ and $(12,12)$ provide close-to-best possible performance, but even $(12,3)$ and $(12,1)$ are only moderately inferior. The very good performance with $(4,1)$ and $(4,3)$ is due to R-channel overlap between senders and receiver, and resultant better chance of all 4 channels being fully utilized.

Fig. 14 shows the aggregate throughput with TCP flows. The throughput is lower than CBR as expected. Furthermore, with 12 channels, larger poolsize yields marginal degradation in throughput. This is partly explainable by the fact that with 12 channels, TCP ACKs sent by node 4 to all sources mostly compete for the single T-interface with each other (with 12 channels, the potential for R-channel overlap is limited) and with LL control packets (which pose greater overhead at larger poolsize). Though the bandwidth consumed is not very significant, the increased delay seen by ACKs affects throughput. To validate this hypothesis, we equipped node 4 with an additional T-interface. Fig. 15 depicts the results, and vindicates the explanation (performance with 4 channels

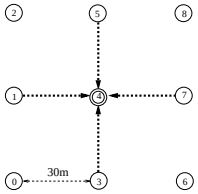


**Figure 12. Scenario 3**

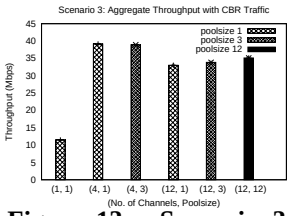


**Figure 13. Scenario 3: CBR Traffic**

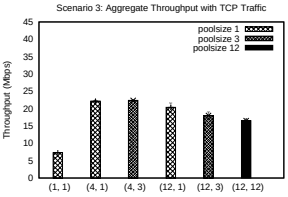


**Figure 14. Scenario 3: TCP Traffic**

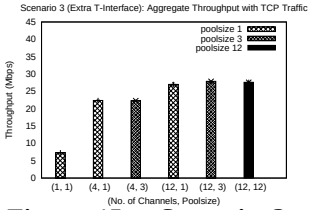


**Figure 15. Scenario 3: TCP Traffic (extra IF)**

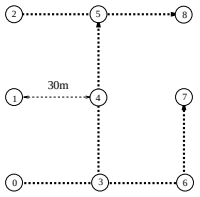


**Figure 16. Scenario 4**

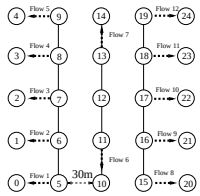


**Figure 17. Scenario 5**

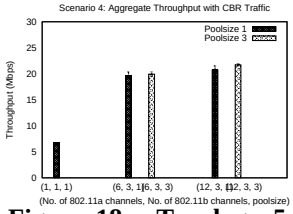


**Figure 18. Topology 5: CBR Traffic**

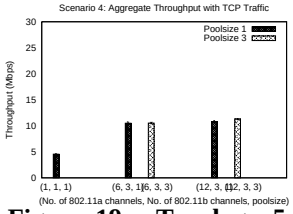


**Figure 19. Topology 5: TCP Traffic**

is unchanged since only the 4 R-interfaces at node 4 can be active, and any T-interfaces are deactivated by LL).

*Scenario 4:*

This topology (Fig. 16) helps evaluate how effectively the LL can schedule packets over different channels/interfaces, given multi-hop flows with routes specified as sequences of nodes. 9 nodes are arranged in a 3 by 3 grid spatial layout. Each node has one R-Interface and one T-interface of each type 802.11a and 802.11b. There are 3 flows: $0 \rightarrow 7$ with manually specified route $0 \rightarrow 3 \rightarrow 6 \rightarrow 7$, $3 \rightarrow 5$ with manually specified route $3 \rightarrow 4 \rightarrow 5$, and $2 \rightarrow 8$ with manually specified route $2 \rightarrow 5 \rightarrow 8$ starting at $t = 40s, 40.5s, 40.6s$ respectively and continuing till $t = 50$.

In the CBR traffic case, the traffic generation rates are : $0 \rightarrow 7$ at $\sim$ 10.9 Mbps , $3 \rightarrow 5$ at $\sim$ 5.46 Mbps, and $2 \rightarrow 8$ at $\sim$ 10.9 Mbps. *Note than an ideal scheduler can meet almost all the traffic demand with just 5 802.11a channels, and 2 802.11b channels.*

The following combinations of (no. of 802.11a channels, no.of 802.11b channels, poolsize) are evaluated: $(1,1,1),(6,3,1),(6,3,3),(12,3,1),(12,3,3)$. With CBR traffic (Fig. 18), even with poolsize 1, the performance is quite good indicating that the LL is able to make good packet scheduling decisions. The performance with poolsize 3 is marginally better than poolsize 1 for a given number of channels. Note $(6,3,3)$ is within $\sim$80% of best-possible performance for CBR traffic (estimated based on Table 1); $(12,3,3)$ is even better.

In the case of TCP (Fig. 19), delays due to multiple hops combined with DATA/ACK contention (akin to previous scenario) result in poorer performance, and little variation across multi-channel combinations.

*Scenario 5:*

25 nodes are arranged in a 5 by 5 grid spatial layout (Fig. 17). Each node is equipped with one pair of 802.11a interfaces (one R-interface and one T-interface). We predesignate 12 (disjoint) one-hop *SD* pairs, as depicted in Fig. 17. We vary the number of channels $c$. If $c$ channels are in use, the first $c$ sources start sending data at $t = 40.0s$ and continue till the end of simulation at $t = 50.0s$. An ideal omniscient scheduler can assign each flow to a separate channel.

Fig. 20 depicts aggregate throughput for CBR traffic at rate $\sim$ 10.9 Mbps. Given $c$ channels, a useful benchmark is to compare with $c$ times the single-channel throughput. The difference between best-possible and our LL increases as $c$ increases, one can see that even with $c = 12$, the LL is able to get within $\sim$ 70% of it. Poolsize 3 shows a small but consistent performance gain over poolsize 1 (statistically significant only for $c = 4$). TCP trends are not much different, and are omitted to save space.

## 9 Discussion

The presented simulation results indicate that in scenarios with diverse radio-configurations, the proposed link layer is able to adapt appropriately, especially in face of interface-conflicts and/or multiple R-interfaces at receiving nodes. Also, a poolsize of 3 generally yields better performance than a poolsize of 1, and there seems limited additional benefit in increasing the poolsize further. It must be noted that when the number of flows is large (e..g, in Scenario 5), the difference is not marked unless the number of channels is small; this is because by a sheer averaging effect, a combinatorially load-balanced assignment ends up being quite good when number of flows or channels is large. An aspect that needs improvement is channel estimation; in particular, we observed that in topologies where some link-separations are very large, a channel-switch within the pool can lead to loss of link-connectivity (since even within the same band, there is some difference in propagation characteristics).

We also remark that this effort is a step towards a broader objective of developing a comprehensive cross-layer scheduling and routing framework, whereby the link-layer encapsulates knowledge of low-level details of channels/interfaces, and computes numerical link-costs that are made available to Layer 3, so that multi-hop flows

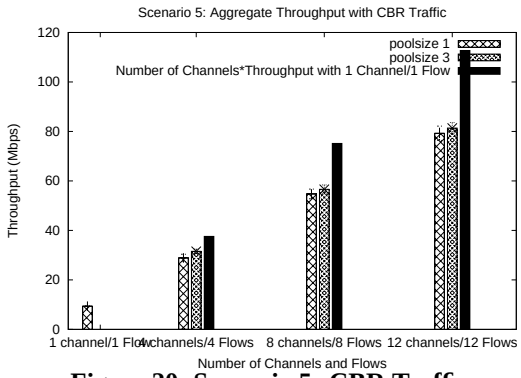


**Figure 20. Scenario 5: CBR Traffic**

can be routed effectively without Layer 3 being exposed to these details. Some preliminary work by us indicates that this approach has potential, but further research is required to develop a metric formulation that works well in a wide range of network scenarios.

## 10 References


[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[2] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," in *Proceedings of IEEE INFOCOM*, 2005, pp. 1804–1814.

[3] X. Wu and R. Srikant, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," in *Proceedings of IEEE INFOCOM*, 2006.

[4] X. Wu, R. Srikant, and J. R. Perkins, "Queue-length stability of maximal greedy schedules in wireless networks," in *Workshop on Information Theory and Applications*, 2006.

[5] V. Bhandari and N. H. Vaidya, "Capacity of multi-channel wireless networks with random (c, f) assignment," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM Press, 2007, pp. 229–238.

[6] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, "Balanced Allocations," *SIAM J. Comput.*, vol. 29, no. 1, pp. 180–200, 2000.

[7] J. Mo, H.-S. W. So, and J. Walrand, "Comparison of multichannel mac protocols," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 50–65, 2008.

[8] A. Tzamaloukas and J. J. Garcia-Luna-Aceves, "Channel-hopping multiple access," in *ICC (1)*, 2000, pp. 415–419.

[9] J. So and N. H. Vaidya, "Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. ACM Press, 2004, pp. 222–233.

[10] P. Bahl, R. Chandra, and J. Dunagan, "Ssch: slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM Press, 2004, pp. 216–230.

[11] H.-S. W. So, G. Nguyen, and J. Walrand, "Practical synchronization techniques for multi-channel mac," in *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2006, pp. 134–145.

[12] J. A. Patel, H. Luo, and I. Gupta, "A cross-layer architecture to exploit multi-channel diversity with a single transceiver," in *Proceedings of IEEE INFOCOM Minisymposium*, Anchorage, AK, USA, May 2007.

[13] P. Kyasanur, C. Chereddi, and N. H. Vaidya, "Net-x: System extensions for supporting multiple channels, multiple interfaces, and other interface capabilities," Technical Report, CSL, UIUC, Aug. 2006.

[14] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Tcp unfairness in ad hoc wireless networks and a neighborhood red solution," *Wirel. Netw.*, vol. 11, no. 4, pp. 383–399, 2005.

[15] A. Warrier, L. Le, and I. Rhee, "Cross-layer optimization made practical," in *Proc. of Broadnets '07*, 2007.

[16] E. Rozner, Y. Mehta, A. Akella, and L. Qiu, "Traffic-aware channel assignment in wireless lans," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 2, pp. 43–44, 2007.

[17] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly, "Distributed channel management in uncoordinated wireless environments." in *MOBICOM*, 2006, pp. 170–181.

[18] W. Wang, X. Liu, and D. Krishnaswamy, "Robust routing and scheduling in wireless mesh networks," in *Proc. of IEEE SECON*, June 2007.

[19] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM Press, 2004, pp. 114–128.

[20] W. Yoon, J. So, and N. H. Vaidya, "Routing exploiting multiple heterogeneous wireless interfaces: A tcp performance study," in *Proceedings of Military Communications Conference (MILCOM)*, 2006.

[21] H. Wu, F. Yang, K. Tan, J. Chen, Q. Zhang, and

Z. Zhang, "Distributed Channel Assignment and Routing in Multi-radio Multi-channel Multi-hop Wireless Networks," *Journal on Selected Areas in Communications(JSAC)*, vol. 24, pp. 1972–1983, Nov. 2006.

[22] M. Cao, V. Raghunathan, and P. Kumar, "Cross layer exploitation of MAC layer diversity in wireless networks," in *Proc. of ICNP*, 2006.

[23] E. Vergetis, R. Guérin, and S. Sarkar, "Realizing the benefits of user-level channel diversity," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 15–28, 2005.

[24] M. Lacage, M. H. Manshaei, and T. Turletti, "Ieee 802.11 rate adaptation: a practical approach," in *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM, 2004, pp. 126–134.

[25] R. Maheshwari, H. Gupta, and S. R. Das, "Multichannel mac protocols for wireless networks," in *Proceedings of IEEE SECON*, 2006.

[26] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, "Understanding and mitigating the impact of rf interference on 802.11 networks," in *Proc. of ACM SIGCOMM '07*, 2007.

[27] S. Fitzpatrick and L. Meertens, "Experiments on dense graphs with a stochastic, peer-to-peer colorer," in *Workshop on Probabilistic Approaches in Search at AAAI'02*, 2002.

[28] D.J.Leith and P.Clifford, "Convergence of distributed learning algorithms for optimal wireless channel allocation," in *Proc. of CDC '06*, 2006.

[29] Information Sciences Institute, "NS-2 network simulator," Version 2.31.