Simple Yet Efficient, Transparent Airtime Allocation for TCP in Wireless Mesh Networks

Ki-Young Jang Konstantinos Psounis Ramesh Govindan
University of Southern California
{kjang,kpsounis,ramesh}@usc.edu
Paper #1569335179, 12 pages

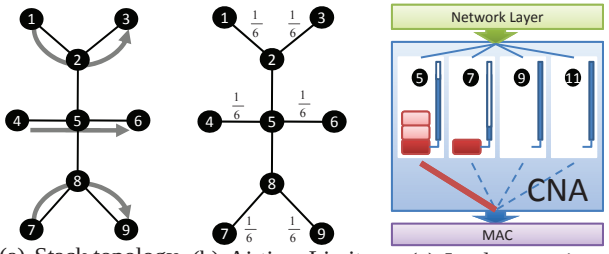
ABSTRACT

In this paper, we explore a simple yet effective technique for explicitly allocating airtime to each active pair of communicating neighbors in a wireless neighborhood so that TCP starvation in a wireless mesh network is avoided. Our explicit allocation is efficient, redistributing unused airtime and also accounting for airtime rendered unusable by external interference. Our technique requires no modifications to TCP/IP and the 802.11 MAC, and is responsive to short flows, MAC-layer auto rate adaptation, and other dynamics, as we demonstate in extensive experiments on two indoor testbeds. Despite its simplicity, the technique is on average within 12% of the max-min optimal allocation on several canonical topologies.

1. INTRODUCTION

It is well-known [10, 23, 29] that, in wireless mesh networks, TCP connections can starve because TCP fails to compete effectively. Figure 1(a) shows a canonical mesh network, called the *stack*, that has been used to illustrate this phenomenon. In this mesh network, which has three TCP flows, the flow in the middle always starves [29, 23, 10, 28]. Intuitively, this starvation occurs because TCP "hunts" for the available channel capacity, and in doing so, triggers channel capture (in which one backlogged transmitting station completely captures the channel at the expense of another) at the 802.11 MAC layer. In turn, channel capture results in multiple end-to-end losses, leading to repeated TCP timeouts and, consequently, low throughput.

To circumvent this failure mode, a thread of research has explored more equitable channel resource allocation techniques for wireless mesh networks. All of these schemes are based on the observation that, in a wireless mesh network, since neighboring nodes share the wireless channel, the available transmission capacity at a node depends on traffic traversing other neighbors. So, to avoid starvation, when a channel congestion is detected, all relevant nodes within the neighborhood are notified, either through explicit signaling [29, 23, 27] or implicitly through queue differential gradients [28]. However, schemes differ in the way congestion is propagated to the source. One class of schemes sends implicit or imprecise feedback by dropping or marking packets [29, 23], or regulating transmissions based on queue differentials [28]. The second class of schemes [23, 27] explicitly computes available channel capacity using a sophisticated model, and then sends precise rate feedback. All of these techniques avoid starvation, and most of them strive for fair allocation of channel resources (for some definition of fairness). Moreover, implicit in much of this work has been the assumption that either TCP or the MAC has to be modified for effective congestion control.



(a) Stack topology (b) Airtime Limits (c) Implementation

Figure 1: CNA Example and Implementation

In this paper, we discuss a novel technique called *Cooperative Neighborhood Airtime-limiting*, CNA. CNA is a hybrid approach, in that it explicitly allocates the channel resources, but provides only imprecise feedback to the source. CNA is qualitatively different from other approaches proposed in the literature (Section 2), and has the following noteworthy properties. (In Section 5, we describe an extensive validation of each of these properties on a mesh testbed with a complete implementation of CNA.)

Airtime Allocation. While existing explicit allocation schemes assign rates, CNA proactively and cooperatively allocates the *total airtime available* within a neighborhood of a given link (the set of links whose transmissions would interfere with the given link) (Section 3). Each link is assigned an *airtime limit*. In Figure 1(a), each active link gets an airtime limit of $1/6$th as shown in Figure 1(b) (this is a simplification for the purposes of this example, and we discuss the precise allocation scheme in Section 3). This limit means that over any time interval T like several milliseconds, each link can only use up $1/6 \cdot T$ for its transmissions. However, each packet on a link must still contend for the channel using the standard 802.11 MAC. If different links operate at different

rates, they will be able to transmit different amounts of traffic over the same time interval, so CNA does not attempt to ensure fair goodput. Retransmissions also get charged airtime usage. This choice of airtime as the resource enables CNA to work correctly in the presence of *auto-rate adaptation* mechanisms that attempt to tailor PHY-layer transmission rates to channel conditions. Specifically, with this choice, links that suffer from high PHY-layer losses do not constrain the goodput achievable on good links, as they would with *rate* allocation. Moreover, this choice enables CNA to be independent of the details of the PHY or MAC layers, so it can *work unmodified across 802.11 standards (a/b/g)*.

Efficiency. CNA's airtime limits change when a new flow begins to traverse a neighborhood, an existing flow departs, or when total airtime in the neighborhood is reduced by external interference (e.g., a microwave oven is turned on) (Section 3). For example, if the flow in the middle of Figure 1(a) were to depart, each link would now get an airtime limit of $1/2$ (since the links on the outer flows do not mutually interfere). When a flow uses less than its airtime-limit, CNA *redistributes the available airtime within the neighborhood*, ensuring efficient channel usage. CNA also *has novel mechanisms to detect external interference*, and adjusts available airtime accordingly. To our knowledge, prior work cannot claim similar properties.

Transparency. CNA's airtime allocation is transparent both to TCP and to the 802.11 MAC (Section 4), in the sense that *no modifications are required to either of these protocols*. CNA is implemented above the MAC layer, as shown in Figure 1(c). Packets to be forwarded on a link are queued by CNA, and transmitted only if the airtime limit has not been exhausted by previous transmissions on the link. If the queue overflows, packets are dropped, and TCP's congestion control adaptation is triggered at the sender. CNA tracks airtime usage by using information from the MAC layer about the number of retransmissions, and the rate at which each packet was transmitted. Because CNA requires no TCP modifications it can *support hybrid TCP connections* that traverse wired and wireless-mesh links. Most prior work cannot make this claim.

Responsiveness and Low Overhead. CNA is responsive to flow dynamics and achieves low signaling overhead (Section 4). A naive implementation of CNA can incur significant signaling overhead in attempting to ensure that airtime-limit adjustments are correctly made when flows enter or depart a wireless neighborhood, or when routing changes happen. Using a combination of promiscuous mode listening, and careful per-packet state encoding, CNA is able to *recompute airtime-limits at packet arrival timescales* and be responsive to flow arrivals and departures, and to short-lived flows. Specifically, airtime limits are dynamically calculated using summary information transmitted in unused fields of the IP header. Thus, CNA is *scalable*: its airtime limits are calculated based on information from within a neighborhood, and it incurs very little additional overhead.

Favorable Comparison with the State-of-the-Art. Using simulation, we show that CNA performs comparably with the state-of-the-art explicit rate allocation with precise feedback scheme, called WCPCap [23] (Section 5). That scheme has, however, not been demonstrated to be deployable, as its designers acknowledge [23]. More important, CNA is *within 5-20% of the max-min rate allocation* on several canonical topologies.

2. RELATED WORK

Extensive research has been done to understand the shortcomings of and improve the performance of TCP in wireless networks, for example, [26, 9, 11, 16, 19, 29]. We briefly discuss broad classes of research pertinent to our work while referring the interested reader to [20] for a more comprehensive survey of congestion control in wireless networks.

Early work on improving TCP performance in wireless networks focused on distinguishing between packet loss due to wireless corruption from loss due to congestion, in the context of last-hop wireless [7, 6] or wireless wide-area networks [25]. In contrast, we address congestion control for multi-hop wireless networks.

More recent work, however, has addressed congestion control in multi-hop wireless settings. One class of work, exemplified by TCP-ELFN [11], TCP-BuS [16] and ATCP [19] concentrates on improving TCP's *throughput* by freezing TCP's congestion control algorithm during link-failure induced losses, especially when route changes occur. Another class of work, exemplified by LRED [9] and ATP [26], discusses TCP performance issues even in the absence of link-failure induced losses. Unlike CNA, these proposals do not explicitly recognize and account for congestion within a neighborhood. (Some of these schemes use congestion metrics that *implicitly* take some degree of neighborhood congestion into account.) As a result, they would exhibit similar shortcomings as TCP – severe unfairness that may lead to starvation and poor overall performance. Moreover, unlike CNA these schemes are not compatible with the existing network stack, as they require TCP modifications and/or cross-layer implementation.

	DiffQ	NRED	WCP	WCPCap	HOP	EZ-Flow	CNA
Complete TCP/IP transparency		✔				✔	✔
Complete MAC transparency		✔	✔	✔		✔	✔
Auto-rate adaptability					✔		✔
External interference adaptability							✔
Practicality (tested with implementation)	✔		✔		✔	✔	✔

Table 1: CNA and recent related works

A few recent pieces of work, however, have recognized the importance of explicitly detecting and signaling congestion over a neighborhood. Table 1 qualitatively compares CNA with these approaches along many dimensions: whether they require changes to TCP or not (TCP transparency, which

would enable TCP connections that traverse both wired and wireless links), require changes to the 802.11 MAC or not (MAC transparency), explicitly track and support dynamic switching of multiple PHY-layer transmission rates (multi-rate support), account for sources of external interference, and whether they have been shown to be practical using a realistic implementation or not. Where a particular feature has been explicitly considered or demonstrated by previous work, we place a check mark in the table. An absence of a check mark indicates either that the feature was not an explicit goal of the design (e.g., some schemes have not considered multi-rate support), or that it was not discussed and hence not known. None of these prior pieces of work achieves the generality and transparency of CNA, and several differ from CNA along more than one dimension.

We now discuss in more detail these recent approaches which explicitly or implicitly signal congestion within the neighborhood, and provide imprecise feedback to the source. NRED [29] identifies a subset of the flows that share channel capacity with flows passing through a congested node, and regulates their rates using a neighborhood queue size and a RED [8]-style marking on packets in this queue. WCP [23] explicitly exchanges congestion information within a neighborhood, and all nodes within the neighborhood mark packets with congestion indicators, triggering rate reductions at the source and resulting in fair and efficient rate allocation. More recently, backpressure congestion control techniques have been explored in wireless mesh networks. DiffQ [28], which proposes backlog-based transmission scheduling and backpressure congestion control, comes closest to one of the explicit goals of our work, transparency. Their mechanisms are implemented above the 802.11 MAC layer, but TCP's standard congestion control is disabled, and senders simply transmit when permitted by DiffQ's backpressure strategy. EZ-Flow [5] is another backpressure congestion control mechanism which does not require explicit signaling. Their design provides transparency, but it is not evaluated with TCP or under dynamics induced by auto-rate adaptation or by external interference. Finally, Hop [18] is a clean-slate design of hop-by-hop congestion control.

Two pieces of work propose explicit allocation of channel capacity by computing the achievable rate region of an 802.11 network. WCPCap [23] proposes a sophisticated stochastic model for estimating the achievable rate region, given packet loss rates, topology, and flow information. It then allocates the achievable capacity fairly across flows, sending precise feedback to sources. EWCCP [27] uses a simpler explicit capacity calculation based on the assumption that the achievable rate region of 802.11 is convex, an assumption later shown to be incorrect in [14].

Finally, several other pieces of work are tangentially related. Researchers [12, 17, 22] have explored theoretical methods under a perfect MAC scheduler or for jointly optimizing scheduling and rate assignment in wireless networks, and [4] describes a way to implement some of these ideas in practice.

3. EFFICIENT AIRTIME ALLOCATION

CNA achieves efficient airtime allocation by distributing available airtime within a wireless neighborhood, then monitoring the airtime utilization and dynamically redistributing unused airtime to improve overall airtime usage. In this section, we describe the airtime allocation algorithms in CNA. In the next section, we discuss how CNA achieves transparency, low overhead, and responsiveness.

Airtime Sharing Neighborhood. CNA enables TCP congestion control to function effectively in a wireless mesh network. The causes for congestion in a wireless network are qualitatively different from those in a wired network. In a wireless network, since neighboring nodes share the wireless channel, the available transmission airtime at a node depends on traffic traversing other neighbors.

More precisely, congestion in wireless networks is defined not with respect to a node, but with respect to transmissions from a node to its neighbor. We use the term *link* to denote a one-hop sender-receiver pair. Then, the set of links $N_{i\to j}$ that share airtime with a given link $i \to j$ is given by [23]:

> the set of all incoming and outgoing links of i, j, all neighbors of i, and all neighbors of j.

A transmission along any link in $N_{i\to j}$ would either cause carrier sense to be triggered at i or cause a collision at j if a packet were to be simultaneously transmitted on $i \to j$.

The neighborhood relationship is symmetric. If a link $i \to j$ belongs to the neighborhood of $k \to l$, $k \to l$ also belongs to the neighborhood of $i \to j$. The definition also applies when RTS-CTS is used in 802.11. However, it does not account for external interference from other wireless networks, nor for transmissions in which the receiver is outside the sender's communication range but within its interference range. We discuss these later in more detail.

CNA Overview. The central idea behind CNA is very simple. It *approximately* divides the wireless channel airtime among all active links within each neighborhood, then limits each link to the assigned airtime share, thereby explicitly allocating airtime and policing these allocations. Thus, a TCP flow in the mesh network encounters a sequence of airtime-limited links, and is bottlenecked by the most constraining link. TCP's congestion control mechanism adapts to this most constraining link in much the same manner as it would in a wired network. Thus, TCP itself need not be modified, since these changes can be transparent to TCP: ensuring this transparency is one of the challenges we address in the paper.

Conceptually, there are three distinct components in CNA's explicit airtime allocation: airtime distribution, re-distribution of unused airtime, and adaptation to external interference. Below, we discuss each of these components.

Airtime Allocation. Assume, for now, that there are no sources of external interference, so all of the airtime in a

neighborhood is available for allocation. In this section, we describe how CNA computes, for each link $i \rightarrow j$ an *airtime-limit* denoted by $A_{i\rightarrow j}$. $A_{i\rightarrow j}$ is a positive fraction. Within any time interval T, the total airtime occupied by all transmissions from i to j cannot exceed $A_{i\rightarrow j}T$.

To define a procedure to compute the airtime-limit, we first define a weight $W_{i\rightarrow j}$ on a link $i \rightarrow j$ as the number of flows traversing the link during a sliding window of time, and define a link $i \rightarrow j$ as *active* if $W_{i\rightarrow j} > 0$. We call the sum of weights in a neighborhood as the *neighborhood weight*, denoted by $NW_{i\rightarrow j}$ for $i \rightarrow j$. Thus,

$$NW_{i\rightarrow j} = \sum_{k\rightarrow l \in N_{i\rightarrow j}} W_{k\rightarrow l}.$$

As an example, consider Figure 2(a) with three TCP flows ($1 \rightarrow 3$, $4 \rightarrow 6$ and $7 \rightarrow 9$). Figure 2(b) shows the value of $NW_{i\rightarrow j}$ for each link. The neighborhood $N_{1\rightarrow 2}$ of link $1 \rightarrow 2$ consists of links $1 \rightarrow 2$, $2 \rightarrow 3$, $2 \rightarrow 5$, $4 \rightarrow 5$, $5 \rightarrow 6$, and $5 \rightarrow 8$, plus all the reverse links. The active links in $N_{1\rightarrow 2}$ are $1 \rightarrow 2$, $2 \rightarrow 3$, $4 \rightarrow 5$, $5 \rightarrow 6$, and the corresponding reverse links as well (for ACK traffic). Thus, $NW_{1\rightarrow 2}$ equals 8 as denoted in the label of link $1 \rightarrow 2$ in Figure 2(b). Similarly, $N_{4\rightarrow 5}$ includes all the links in the topology. The set of links in $N_{4\rightarrow 5}$ carrying data is $1 \rightarrow 2$, $2 \rightarrow 3$, $4 \rightarrow 5$, $5 \rightarrow 6$, $7 \rightarrow 8$, $8 \rightarrow 9$, and $NW_{4\rightarrow 5}$ equals 12 if we also account, as before, the reverse links traversed by ACKs.

Since contention within a neighborhood is roughly proportional to $NW_{i\rightarrow j}$, one could consider assigning to each link $i \rightarrow j$ an airtime-limit equal to $W_{i\rightarrow j}/NW_{i\rightarrow j}$. In practice, this simple approach may lead to instability because it does not take into consideration all the interdependencies between different neighborhoods. Specifically, if a link interferes with another link, which, in turn, interferes with another link, and so on, the dependence spans the entire network. Indeed, if we use these airtime-limits in the topology of Figure 2(a), it leads to instability as we have verified using simulation.

Instead, we adopt the following rule which guarantees stability without being too conservative: We assign to each link $i \rightarrow j$ an airtime-limit equal to

$$A_{i\rightarrow j} = \frac{W_{i\rightarrow j}}{max_{k\rightarrow l \in N_{i\rightarrow j}}(NW_{k\rightarrow l})} = \frac{W_{i\rightarrow j}}{D_{i\rightarrow j}},$$

where $D_{i\rightarrow j} = max_{k\rightarrow l \in N_{i\rightarrow j}}(NW_{k\rightarrow l})$ will henceforth be referred to as the airtime divider of link $i \rightarrow j$. Thus, each link is constrained not by its own neighborhood weight, but by the maximum neighborhood weight over its entire neighborhood. In Figure 2(a), this choice lets the most constrained neighborhoods, such as the neighborhood of link $4 \rightarrow 5$, constrain the airtime limit of all flows crossing this neighborhood, including the two outer flows. Indeed, this rule assigns as airtime-limit $\frac{1}{12}$ for all links, as shown in Figure 2(c).

Our choice for airtime allocation trades off some efficiency for stability. To see why this is so, consider again Figure 2(a). Under an ideal scheduling scheme, links $1 \rightarrow 2$ and $7 \rightarrow 8$ can be scheduled simultaneously, and the same holds true for links $2 \rightarrow 3$ and $8 \rightarrow 9$. In other words, the two outer flows may be scheduled to capture the channel simultaneously. Thus, each of the links in this topology can utilize $\frac{1}{8}$ of airtime, if we use, for example, the following time division multiplexing scheme: first $1 \rightarrow 2$ and $7 \rightarrow 8$ together, then $4 \rightarrow 5$, then $5 \rightarrow 6$, then $2 \rightarrow 3$ and $8 \rightarrow 9$ together, and so on.

With 802.11, the likelihood of sustained synchronized scheduling is very low and using $\frac{1}{8}$ as airtime-limit for all links is clearly not feasible. If the two outer flows are not synchronized at all, then it is easy to see that the maximum airtime-limit allocation for each link equals $\frac{1}{12}$, as, in the absence of synchronization, links $1 \rightarrow 2/2 \rightarrow 3$ and $7 \rightarrow 8/8 \rightarrow 9$ cannot be scheduled simultaneously. The rest of the links interfere with each other and have to be scheduled one by one.

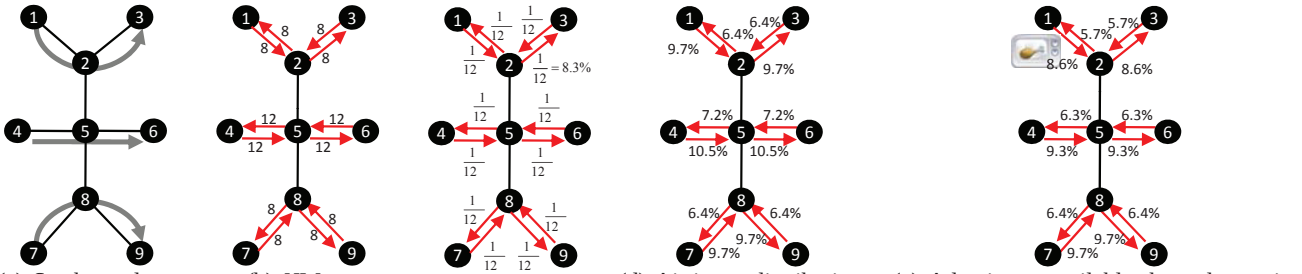
Is it possible to assign an airtime-limit between $\frac{1}{8}$ and $\frac{1}{12}$? After all, under 802.11, links $1 \rightarrow 2$ and $7 \rightarrow 8$ may transmit simultaneously every now and then as they are both carrier-sensing the same links, $4 \rightarrow 5$ and $5 \rightarrow 6$. Thus, the optimal airtime allocation under 802.11 should be somewhere between $\frac{1}{12}$ and $\frac{1}{8}$, but computing this optimal requires analyzing the complete topology (see [14] for more details) and it would be too expensive to implement (see [23] for an approximate, distributed implementation). Our choice is attractive because it permits a stable, decentralized implementation, with a small loss of efficiency (which we quantify later). As an aside, by "optimal" we mean the max-min optimal allocation under 802.11, and for this topology the max-min optimal yields equal allocation for all three flows.

Unused Airtime Re-distribution. In practice, the total traffic on a link $i \rightarrow j$ may never attain the airtime-limit $A_{i\rightarrow j}$. This may happen for several reasons: $i \rightarrow j$ may only be carrying ACK traffic, or the flows through $i \rightarrow j$ may be receiver-limited, or limited by a more constraining airtime-limit elsewhere along the path. To achieve greater efficiency, CNA distributes the unused portion of the airtime-limit to other links in the neighborhood. It first computes the unused airtime for each outgoing link of the node on which it runs, then informs neighboring links. Thereafter, each link computes the amount of unused airtime in its neighborhood by considering the unused airtime of all its neighbors, and increases its airtime-limit accordingly.

More specifically, let $U_{i\rightarrow j}$ be the utilization on link $i \rightarrow j$, defined as the proportion of the assigned airtime-limit used by the link. Then, the total unused portion of the airtime-limit on link $i \rightarrow j$ is defined by

$$UA_{i\rightarrow j} = A_{i\rightarrow j}(1 - U_{i\rightarrow j}).$$

We choose to re-distribute $UA_{i\rightarrow j}$ to each link $k \rightarrow l$ within $N_{i\rightarrow j}$ proportionally to $k \rightarrow l$'s weight, to achieve fairness at the flow level (other allocation policies are possible, of course, but we have deferred an exploration of these). Thus, each link $k \rightarrow l$ within $N_{i\rightarrow j}$ can increase its airtime-limit by



(a) Stack topology (b) $NW_{i\to j}$ (c) $A_{i\to j}$ (d) Airtime redistribution (e) Adapting to available channel capacity

Figure 2: Illustrating the design of CNA

taking on some of the unused airtime on link $i \to j$:

$$UA_{i\to j}\frac{W_{k\to l}}{NW_{i\to j}} = RA_{i\to j}W_{k\to l},$$

where $RA_{i\to j} = UA_{i\to j}\frac{1}{NW_{i\to j}}$ is the unit of reusable airtime on link $i \to j$. Similarly, link $i \to j$ may use a portion of the unused airtime from each of its neighbors, and can increase its airtime-limit thanks to the unused airtime on all of its neighbors by

$$\sum_{k\to l\in N_{i\to j}} UA_{k\to l}\frac{W_{i\to j}}{NW_{k\to l}} = NRA_{i\to j}W_{i\to j},$$

where $NRA_{i\to j} = \sum_{k\to l\in N_{i\to j}} RA_{k\to l}$ is the sum of the units of reusable airtime on link $i \to j$'s neighborhood.

Thus, the new airtime-limit $A'_{i\to j}$ of link $i \to j$ is given by

$$A'_{i\to j} = A_{i\to j}U_{i\to j} + \sum_{k\to l\in N_{i\to j}} UA_{k\to l}\frac{W_{i\to j}}{NW_{k\to l}}.$$

Figure 2(d) illustrates the re-distribution of unused air-time. Assume that all ACK flows in Figure 2(c) use only 60% of their allotted air-time limit. CNA redistributes the unused airtime from each reverse link across its neighborhood, resulting in the limits shown in the figure.

Adapting to Available Channel Airtime. So far, we have assumed that all the airtime in a neighborhood is usable. In reality, external interference from other networks (e.g., Bluetooth) or other devices emitting in the same band (e.g., microwaves), may occupy some of the channel airtime at a node. To account for this, we use information available in a popular 802.11 chipset (see Section 4) to measure ACA_i, the available channel airtime at node i after subtracting the airtime occupied by external interference.

Let $ACA_{i\to j}$ be the available channel airtime within the neighborhood $N_{i\to j}$ of link $i \to j$. We compute $ACA_{i\to j}$ as the lesser of the available airtime at the ends of the link:

$$ACA_{i\to j} = min(ACA_i, ACA_j).$$

For stability, we want

$$\sum_{k\to l\in N_{i\to j}} A'_{k\to l} = NA'_{i\to j} \le ACA_{i\to j}, \qquad (1)$$

i.e., $NA'_{i\to j}$, the total neighborhood airtime usage on a link, cannot exceed the available channel airtime. Suppose that the inequality in Equation (1) does not hold for link $i \to j$. We scale down each $A'_{k\to l}$ to a new $A''_{k\to l}$ where $k \to l \in N_{i\to j}$, such that the new airtime-limits satisfy the inequality. If link $i \to j$ is the only link for which the inequality is not satisfied, then using $S_{i\to j} = \frac{ACA_{i\to j}}{NA'_{i\to j}}$ to scale the airtime-limits of all links in the neighborhood of link $i \to j$ suffices to stabilize the system. However, we need to consider the case where there are multiple links in the neighborhood for which the inequality is not satisfied, in which case we need to use as a scaling factor the smallest $S_{k\to l}$ over all links $k \to l \in N_{i\to j}$. Letting $NS_{i\to j}$ denote this neighborhood scaling factor that produces the stable airtime-limits $A''_{k\to l}$, we have

$$NS_{i\to j} = min_{k\to l\in N_{i\to j}}(S_{k\to l}),$$

and we set $A''_{k\to l} = NS_{i\to j}A'_{k\to l}$ for all $k \to l \in N_{i\to j}$.

Figure 2(e) illustrates the adaptation to available channel capacity when a microwave oven is placed near node 1 at a time when airtime-limits are as shown in Figure 2(d). Assume that the microwave source uses up 40% of the airtime around node 1. Each link in the neighborhood of links $1 \to 2$ and $2 \to 1$ have reduced airtime limits, such that the total airtime around node 1 does not exceed 60%. The links traversed by the bottom flow are unaffected.

4. ACHIEVING TRANSPARENCY, LOW OVERHEAD, AND RESPONSIVENESS

The design of CNA requires node i to know, for each of its links, several quantities: $i \to j$, $NW_{i\to j}$, $D_{i\to j}$, $NRA_{i\to j}$ $NA'_{i\to j}$ and $NS_{i\to j}$. Once it has this information, the node can precisely compute the airtime limit $A''_{i\to j}$ and enforce the airtime-limit on each link. We describe how these steps are implemented in CNA in a manner transparent to TCP and the 802.11 MAC layer.

Before we do this, we discuss two requirements that we place on CNA. The first is *low overhead*; clearly, wireless bandwidth is not abundant and the scheme must be careful in limiting the amount of control overhead. The second is *responsiveness*; TCP flows can be short (on the order of a few packets), so the scheme must support changes to airtime-limits on these timescales. These two requirements make the design of CNA non-trivial, as we discuss below.

To ensure responsiveness, one can beacon the above values in a separate control message as often as the status of a

link changes, but this can cause high overhead. Our CNA implementation achieves responsiveness without increased overhead by encoding most of these values in unused fields of the IP header. This per-packet signaling, together with packet overhearing, is used to efficiently exchange various values within the neighborhood. Finally, CNA uses low-rate periodic beacons to exchange slowly-varying information.

The following paragraphs describe, in detail, the implementation of CNA.

Optimizing Information Exchange. Computing the quantities $NW_{i\to j}$, $D_{i\to j}$, $NRA_{i\to j}$, $NA'_{i\to j}$ and $NS_{i\to j}$ may seem to require a prohibitive amount of signaling over the neighborhood. However, CNA leverages the special structure of its computations and of the neighborhood definition to significantly reduce the amount of information exchanged. Specifically, it aggregates information at each hop before re-distributing it to neighboring nodes whose links are part of the neighborhood.

Table 2 shows that the computation of a neighborhood quantity can be decomposed by computing two kinds of intermediate values: a "one-hop quantity" and a "two-hop quantity". Intuitively, a one-hop quantity is a summary of information at a node i and all its attached links, and a two-hop quantity uses information learned by overhearing neighbors' transmissions to other nodes. Notice from Table 2 that, with this decomposition, the structure of the neighborhood quantity computations fall into two classes: one exemplified by $NW_{i\to j}$, and the other by $D_{i\to j}$. In the paragraphs below, we describe how these are decomposed in order to optimize signaling overhead. For brevity, we omit the details of computing the 3 other neighborhood quantities, since their structure is similar to the two we describe.

Calculating $NW_{i\to j}$. To calculate this, suppose we define the following one-hop quantity $W_{i\leftrightarrow j}$:

$$W_{i\leftrightarrow j} = W_{i\to j} + W_{j\to i}$$

$W_{i\leftrightarrow j}$ can be obtained at node i (or node j) without exchanging any infomation by counting the number of outgoing flows from node i and incoming flows from node j.

Let K_i be the set of active links within two hops of node i. Then, observe that node i can easily obtain the set of $W_{k\leftrightarrow l}$ where link $k\leftrightarrow l$ is in K_i. By itself, node i can determine the weights of directly-attached active links. By overhearing packets sent from each neighbor k to another node l, node i obtains the weights of active links two hops away. Using these two pieces of information, node i can compute a two-hop quantity W'_i as described in Table 2.

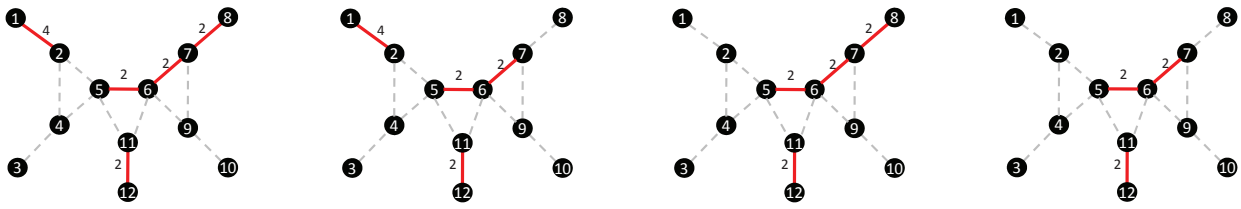
Finally, node i and node j can calculate the neighborhood quantity, $NW_{i\leftrightarrow j}$, after exchanging the two-hop quantities W'_i and W'_j with each other, as shown in Table 2. To do so, they need to compute $K_i \cap K_j$, for which each node needs some topology information. In our current implementation, CNA sends a topology message that contains only two hop connectivity information every T_b (20 seconds). More generally, this information can (and should) be integrated into whichever mesh routing protocol is used to track topology changes and compute paths. We have left this integration to future work. Regardless, this message suffices to track neighborhood changes at the same timescale as routing changes, and the overhead of doing this is comparable to the overhead of dynamic routing.

To understand how CNA uses the topology information, consider Figure 3(a) in which the solid lines indicate active links, and each label on each active link $i\to j$ is $W_{i\leftrightarrow j}$. Then, from Figures 3(b), 3(c) and 3(d) it is easy to infer that $W'_5 = 10$, $W'_6 = 8$ and $\sum_{k\leftrightarrow l\in K_5\cap K_6} W_{k\leftrightarrow l} = 6$. Thus, $NW_{5\leftrightarrow 6} = 12$. For W'_5, node 5 calculates $W_{5\leftrightarrow 6}$ by itself, and obtains $W_{1\leftrightarrow 2}$, $W_{11\leftrightarrow 12}$, $W_{6\leftrightarrow 7}$ by overhearing transmissions from nodes 2, 6 and 11. For W'_6, node 6 calculates $W_{5\leftrightarrow 6}$ and $W_{6\leftrightarrow 7}$ by itself, and obtains $W_{11\leftrightarrow 12}$ and $W_{7\leftrightarrow 8}$ by overhearing transmissions from nodes 7 and 11.

There is one subtlety we have omitted in the discussion so far. Suppose that in our working topology shown in Figure 3(a), link $8\to 7$ carries some packets (destined, say, to node 6) while the reverse link $7\to 8$ is not used at all (i.e., routing is asymmetric and ACK packets return via some other path, say via the $6\to 9\to 10$ path which leads to node 8 via more links not shown in the figure). Then, node 6 cannot detect if $8\to 7$ is active since it relies on link $7\to 8$ to detect this; it can certainly overhear MAC-layer acknowledgements sent by 7, but because the MAC layer header has no information about the identity of the sender, it cannot infer that $8\to 7$ is active. In these cases, node 7 explicitly sends a control packet which indicates the existence of the unidirectional active link and includes one-hop and two-hop quantities, to node 8. Node 6 can then adjust its count appropriately. This control packet, and the topology beacon, are the only two additional control messages used by CNA.

Calculating $D_{i\to j}$. As we described in the previous section, $D_{i\to j}$ is the largest $NW_{k\to l}$ over all $k\to l$ in the neighborhood of $i\to j$. To compute $D_{i\to j}$, we compute M_i and M'_i, a one-hop quantity and a two-hop quantity respectively as shown in Table 2. First, each node computes M_i as the largest neighborhood weight of active links on all its outgoing links. It then encodes M_i in the IP header transmitted from node i. After a few packet exchanges, i will have the M_j values for all its neighbors j. Using this, it can compute M'_i, the maximum among M_i and all the M_j's: essentially, in this step, CNA aggregates the maximum values and reduces information. Finally, by obtaining M'_j from neighbor j (again, by encoding it in an IP header of a packet), i can compute $D_{i\to j}$.

Measuring External Interference. To measure external interference, we rely on registers on a popular 802.11 chipset (Atheros) which are accessible using an open-source driver (Madwifi). Essentially, the wireless card performs carrier sense at a high frequency and updates four registers which count: the number of times carrier sense is performed, the number of times the channel was found to be *busy*, or a *transmission* or *reception* was in progress. Using the difference



(a) Example (b) K_5 (c) K_6 (d) $K_5 \cap K_6$

Figure 3: Basic rule to find neighborhood quantities

One-hop quantity	Two-hop quantity	Neighborhood quantity
$W_{i\leftrightarrow j}$	$W'_i = \sum_{k\leftrightarrow l \in K_i} W_{k\leftrightarrow l}$	$NW_{i\rightarrow j} = W'_i + W'_j - \left(\sum_{k\leftrightarrow l \in K_i \cap K_j} W_{k\leftrightarrow l}\right)$
$M_i = max_j\left(NW_{i\rightarrow j}\right)$	$M'_i = max_{k=i}$, one-hop neighbor of $_i\left(M_k\right)$	$D_{i\rightarrow j} = max\left(M'_i, M'_j\right)$
$U_{i\rightarrow j}$, $RA_{i\leftrightarrow j}$	$RA'_i = \sum_{k\leftrightarrow l \in K_i} RA_{k\leftrightarrow l}$	$NRA_{i\rightarrow j} = RA'_i + RA'_j - \left(\sum_{k\leftrightarrow l \in K_i \cap K_j} RA_{k\leftrightarrow l}\right)$
$A'_{i\leftrightarrow j}$	$V_i = \sum_{k\leftrightarrow l \in K_i} A'_{k\leftrightarrow l}$	$NA'_{i\rightarrow j} = V_i + V_j - \left(\sum_{k\leftrightarrow l \in K_i \cap K_j} A'_{k\leftrightarrow l}\right)$
ACA_i,$S_i = min_j\left(S_{i\leftrightarrow j}\right)$	$S'_i = min_{k=i}$, one-hop neighbor of $_i\left(S_k\right)$	$NS_{i\rightarrow j} = min\left(S'_i, S'_j\right)$

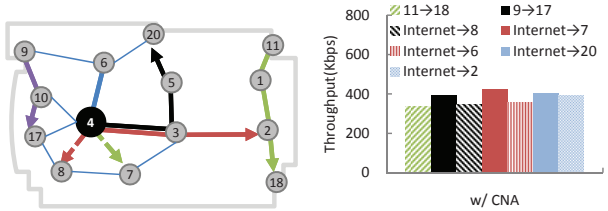
Table 2: Relations among one-hop, two-hop and neighborhood quantities

between the busy counter and the sum of the tranmission and reception counters, we can measure the fraction of airtime occupied by sources of external interference. A more detailed discussion of this, together with a validation of our measurement approach, can be found in [13].

Encoding quantities in the IP header. Twelve quantities need to be transmitted between nodes, seven one-hop quantities, and 5 two-hop quantities as shown in Table 2. We use the following fields in the IP header to carry this information: Type-of-Service, Identification, Flags and Fragmentation Offset. This strategy is based on the observation that the amount of fragmented IP traffic is negligible [24]. When an IP packet is not fragmented, CNA sets the *Reserved bit* in Flags field to mark that it (and the Id and offset fields) carries CNA information; otherwise, information is only encoded in the Type-of-Service field. Each value is encoded into 5 bits. Of the available 37 bits (we do not use DF and MF bits), two bits are used to specify which values are encoded. Thus, seven values can be encoded to one IP header, and two packets exchanges suffice to convey the current values of all quantities at a node. As soon as an updated set of quantities is received, CNA recomputes the neighborhood quantities. All IP header modifications are performed in the CNA "layer", requiring no modifications to IP.

Enforcing airtime limits. CNA enforces airtime limits by a) carefully accounting for the overhead of transmitting all 802.11-layer headers, the preamble, and the initial backoff; b) using a token-bucket per neighbor to enforce airtime limits, and servicing each flow within a token bucket in round-robin fashion; c) estimating the airtime consumed by a packet by correctly accounting for the PHY rate at which the packet was transmitted; and d) reserving a small amount of airtime for broadcast packets, and appropriately charging broadcast airtime to all active links. A more detailed discussion of these mechanisms can be found in [13].

5. EXPERIMENTAL EVALUATION



(a) Testbed **A** (b) Throughput

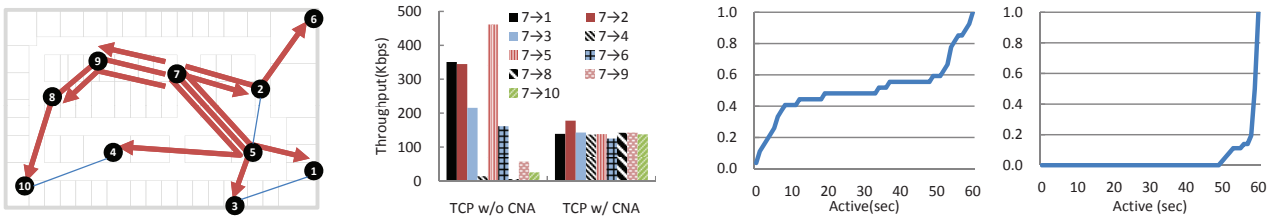
Figure 5: TCP connections traversing wired links

We have conducted extensive experiments to validate CNA's functionality and assess its performance. In this section, we present results from this experimental evaluation.

Goal and Methodology. The goal of our experiments is to validate, using a real implementation, the following properties of CNA: starvation-avoidance; equitable airtime allocation under different topologies, numbers of flows, and radio technologies; responsiveness to dynamics; efficient redistribution of underutilized airtime; seamless operation over MAC layer auto-rate adaptation; transparency so when mesh network flows can traverse wired links; and adaptation of airtime limits in response to external interference. In Section 6, we examine how far off CNA is from the optimal.

We have implemented CNA as a user-level element in Click [21]. We have not modified the 802.11 MAC or TCP. We use Iperf [1] to generate TCP traffic and a microwave oven to generate external interference.

In all our experiments, each mesh node is an eBox-3854, a mini-PC equipped with a NMP-8602 wireless card which supports 802.11a/b/g. The box runs Ubuntu (Linux kernel 2.6.22) and uses the Madwifi driver [2]. Wireless cards are operated in monitor (promiscuous) mode, since CNA needs to overhear transmissions. In all the experiments, RTS/CTS is disabled by default (CNA does not require RTS/CTS but will work if it is enabled). For our 802.11b experiments at 11Mbps, we use channel 14 (which, in some countries in-



(a) Testbed **B** (b) Throughput (c) w/o CNA (d) w/ CNA

Figure 4: Experiment result from **B** testbed

cluding ours, is unused by commercial cards and does not overlap with any other permitted channel, and therefore ensures that we do not observe uncontrolled external interference). We enabled auto-rate adaptation on some nodes: the Madwifi driver uses SampleRate as a default rate control algorithm. For 802.11b, the wireless card uses a short preamble, and MAC layer ACKs are transmitted at 2Mbps.

We present results from experiments conducted on two different indoor testbeds. The first testbed, called **A**, consists of 14 nodes deployed in a building on a university campus. The second testbed, called **B**, consists of 10 nodes deployed in a single floor of an office building. Both of these are fairly harsh wireless environments, and we have conducted several experiments on these testbeds, only some of which we are able to present due to lack of space.

Starvation-Avoidance and Transparency. Our first experiment demonstrates that CNA *avoids starvation* and works on *802.11a*, while providing *complete TCP and MAC transparency*. In this experiment, conducted on testbed **B**, we run a traffic pattern that mimics a community access mesh network with nodes downloading content through a gateway (Figure 4(a)). All nodes download a file from a single node 7, and start almost simultaneously. We ran the same experiment five times with and without CNA. In each run, every flow lasts for 60 seconds.

Figures 4(b) shows the average throughputs that each flow achieves. In each run, with CNA *no starvation occurs and the throughput for each flow is relatively consistent*. Without CNA, *three or more TCP flows starve in every experimental run.* Figures 4(c) and 4(d) help us understand the reason behind this dramatic performance difference. We divide the time into 1-second bins, and a flow is said to be active if at least one data packet of this flow was received by a node in that second (i.e., some useful work was done in that second). We then define the *activity* of each flow as the total number of bins for which it was active. These figures plot the distribution of flow activity across all flows in all experiments. As the figures shows, with CNA, no flow has an activity less than 50 seconds, and the average activity is 58.5 seconds. By contrast, without CNA, almost 40% of the flows are active for less than 10 seconds and the average activity is 31.5 seconds. Clearly, for nearly half the duration of a flow, on average, no useful work is done, because the connection incurs repeated timeouts.

Our second experiment, conducted on testbed **A** with 802.11b, demonstrates that CNA works over multiple 802.11 standards, and works seamlessly when some TCP flows on the mesh network also traverse a wired network, a compelling demonstration of its *TCP transparency*. In this experiment, node 4 is set to be an Internet gateway (Figure 5(a)), and five TCP flows download a file from a website 2 wired hops from 4. There are also two other TCP flows which traverse only the mesh network. All flows use a 1500 byte MTU. This experiment runs for five minutes, and the average throughput achieved by each flow shown in Figure 5(b). All flows are in the neighborhood of the most congested link $3 \rightarrow 4$, and, as it is evident from the figure, they share equal airtime and no flow starves. The slight differences in throughput can be explained by slight differences in packet loss-rates observed on links: recall that in CNA retransmissions are charged airtime, so links of different quality will achieve different goodputs.

CNA Behavior Under Dynamics. To validate CNA behavior under various kinds of dynamics, we run a single experiment on testbed **A** where we script a scenario in which 12 flows (long and short) between nodes on the mesh network arrive and depart over 10 minutes and use 10 different routes. During this scenario, we induce route changes and external interference, and enable auto-rate adaptation. Figure 6 depicts the testbed and the TCP flows used in this experiment; all radios use 802.11b. Figure 7 shows the evolution of TCP throughput for these flows. It also shows different segments of the scenario designed to illustrate different capabilities of CNA. We now describe each segment from one of our runs: we have conducted several runs of this scenario, and have experimented with other scenarios as well. During this scenario, a total of nearly 105 MB of data were transmitted over the network.

Airtime-limit allocation. Our first segment focuses on the first 90 seconds of Figure 7, and illustrates the efficacy of CNA's basic airtime-limit allocation mechanism. We first describe the interval between 0 and 90 seconds in Figure 7. At time 0, two flows $f_{1\rightarrow 18}$, which uses the route $1 \rightarrow 2 \rightarrow 18$, and $f_{20\rightarrow 9}$, which uses the route $20 \rightarrow 6 \rightarrow 9$, start. Auto-rate adaptation is enabled on node 20 for the entire duration of the experiment. The links that $f_{1\rightarrow 18}$ traverses do not share airtime with the links that $f_{20\rightarrow 9}$ traverses, so they do not contend for bandwidth. This can be seen in Fig-

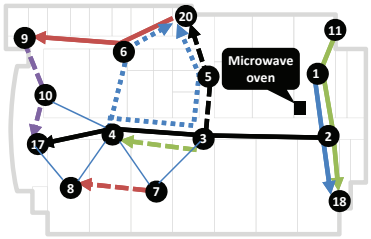


Figure 6: Flow configuration on testbed **A**

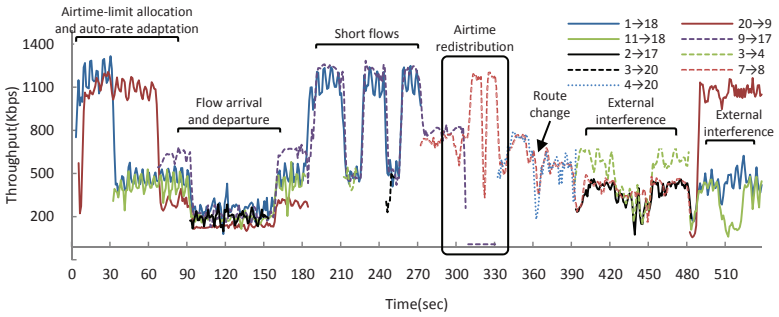
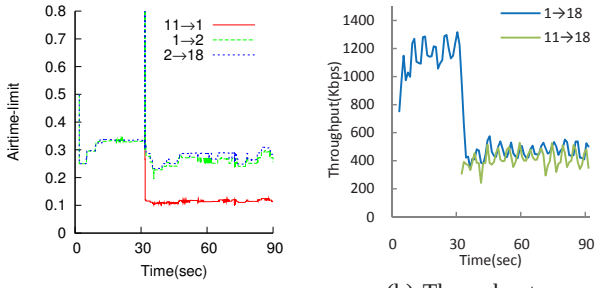


Figure 7: Throughput from an experiment on **A** testbed



(a) Airtime-limit (b) Throughput

Figure 8: links $f_{1\to18}$ and $f_{11\to18}$ traverse

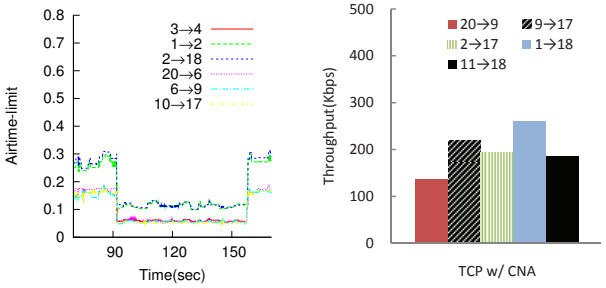
ure 8(a): the links 20 → 6 and 6 → 9 get an airtime limit of 60%, and the reverse links get (not shown) an airtime limit of 20% each, totaling 100%.

We start another flow $f_{11\to18}$ at 30 seconds, and its links are in the same neighborhood as the links of $f_{1\to18}$. Figure 8(a) shows the airtime-limit allocation that CNA assigns. Before $f_{11\to18}$ starts, links 1 → 2 and 2 → 18 have the same airtime limit. Once $f_{11\to18}$ starts, however, the airtime-limits on both its links are reduced, and but they each get an airtime-limit twice that of link 11 → 1. That is because these links carry two flows ($f_{1\to18}$ and $f_{11\to18}$) while link 11 → 1 carries only one flow ($f_{11\to18}$). This illustrates that CNA assigns airtime-limits proportional to the number of flows traversing a link, resulting in fair throughput allocation to all the flows (Figure 8(b)). A closer look at the figure reveals that $f_{11\to18}$ gets a slightly lower throughput because it has a slightly higher RTT.

Auto-rate adaptation. Now consider $f_{20\to6}$ and observe that auto-rate adaptation results in about 11 PHY-layer rate changes before 60 seconds (Figure 9(c)). Interestingly, this does not affect the throughput of this flow. In 802.11, the preamble is transmitted at 2Mbps. Even if each packet waits for the average initial backoff period, it turns out that, with our 512-byte payloads and a TCP ACK, the airtime difference between a packet sent at 11Mbps and at 5.5 Mbps is a little over 20%. This difference is comparable to the throughput fluctuations shown in Figure 9(b).

We introduce, a little after 60 seconds, $f_{9\to17}$. This flow passes through neighborhoods that overlap with $f_{20\to9}$, but not with the other flows. Although one would expect these two flows to get equal throughput because CNA allocates approximately the same airtime-limits to all four links (Figure 9(a)), they don't (Figure 9(b)). Flow $f_{20\to9}$ gets much lower goodput. Some of this throughput loss can be explained by the brief choice of a 2Mbps rate by the auto-rate mechanism (Figure 9(c)).

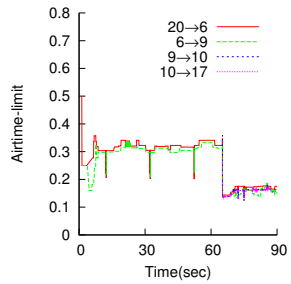
More interesting, however is the observation that the introduction of the new flow increases channel losses and the average number of retries on 20 → 6 (Figure 9(d)), causing it to expend airtime on retransmissions with a consequent loss of throughput. We believe there is a very subtle reason for this. In Section 4, we explained how, using device-level registers, CNA is able to correctly *detect* internal interference: i.e., interference from nodes whose packets it cannot decode. When all such interferers are within two hops, CNA correctly allocates airtime to them. However, in the relatively infrequent event that an interferer is three hops away, CNA's signaling is unable to appropriately assign airtime. In this example, we conjecture that transmissions on 10 → 17, which is three hops away from 20 → 6, causes interference at node 6, resulting in reduced goodput on $f_{20\to9}$. Regardless, $f_{9\to17}$ gets its fair share of airtime and goodput, and is unaffected.



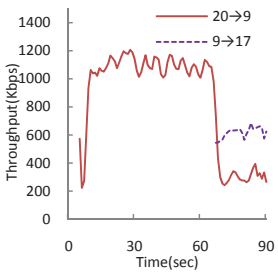
(a) Airtime-limit (b) Throughput

Figure 10: Adding and removing flows

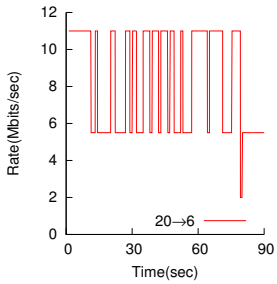
Flow departure. We have illustrated CNA's adaptation to flow arrival above. We now explore flow departure, from 90 to 180 seconds, in Figure 7. Around 90 seconds, we introduce one more flow $f_{2\to17}$, and it runs for 60 seconds. This flow traverses the middle of topology, causing all 5 active flows to traverse at least one link in the neighborhood of link 3 → 4. Figure 10(a) shows how the airtime-limits change on some links in $N_{3\to4}$. Adding $f_{2\to17}$ results in a reduction of airtime-limits for all links, but before $f_{2\to17}$ starts and after it



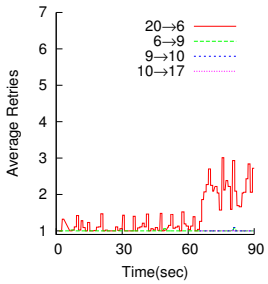
(a) Airtime-limit



(b) Throughput



(c) Auto-rate adaptation



(d) Average retries in MAC

Figure 9: links $f_{20\to9}$ and $f_{9\to17}$ traverse

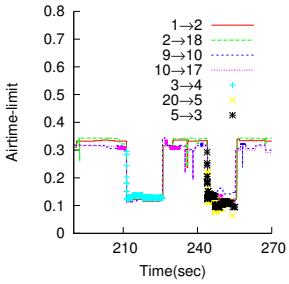


Figure 11: Responsiveness

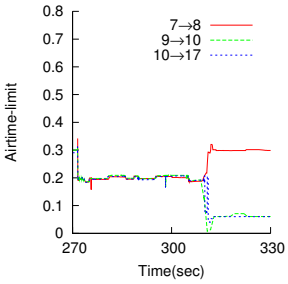


Figure 12: Airtime redistribution

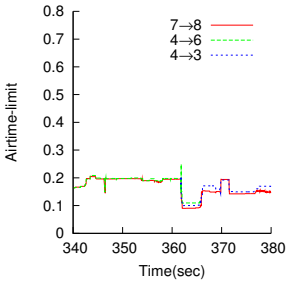


Figure 14: Route change

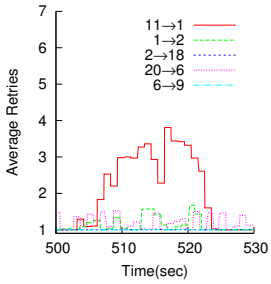


Figure 15: Average retries in MAC

departs, the airtime-limit allocations are roughly same. The factor of two difference in the airtime-limit for links $1 \to 2$ and $2 \to 18$ exists because these links carry two flows. Figure 10(b) shows the throughput each flow achieves in this interval. As explained above, $f_{20\to9}$ gets lower throughput because of increased retransmissions. The throughput differences between the other flows are due to differences in RTT (the 3 hop flows $f_{2\to17}$ and $f_{11\to18}$ get lower throughput) and in packet loss rates.

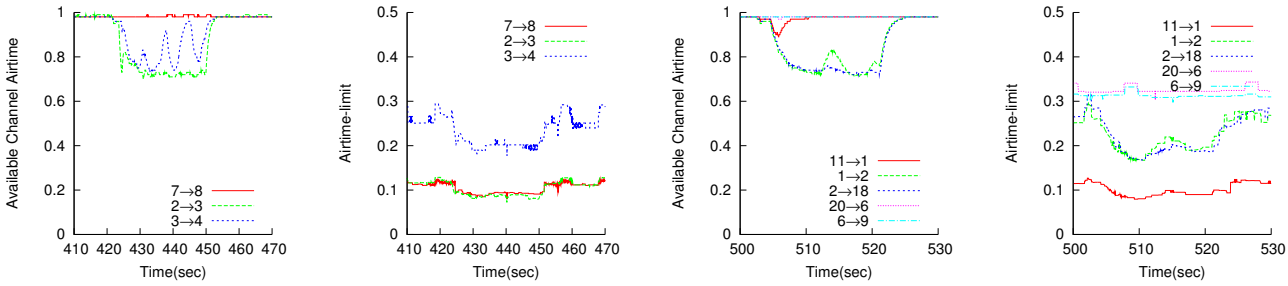
Short Flows. CNA uses per-packet signaling to achieve responsiveness. The interval from 180 and 270 seconds in Figure 7 demonstrates this. In this interval, two long flows $f_{1\to18}$ and $f_{9\to17}$ are present, and two short flows $f_{3\to4}$ and $f_{3\to20}$ start around at 210 and 240 respectively. Each runs for 10 seconds, and they transmit approximately 611 KB and 370 KB respectively. As Figures 11 and 7 show, CNA quickly responds to each new flow, reducing the airtime of the contending links and, as a result, the throughput of the long flows. When the short flows leave, the absence of their packets is quickly noticed, and the long flows regain their airtime-limits and throughputs.

Unused airtime redistribution. We use the interval between 300 and 350 seconds in Figure 7 to demonstrate the efficacy of CNA's unused airtime re-distribution mechanism. Around at 310 seconds, the existing flow $f_{9\to17}$ ends, and another low-rate flow $f_{9\to17}$, which sends only at 8 Kbps starts. Links $9 \to 10$ and $10 \to 17$ have unused airtime, and this is redistributed to other links in their neighborhood. Thus, $7 \to 8$'s airtime-limit is now increased to 30% (Figure 12), and flow $f_{7\to8}$ achieves greater throughput while $f_{9\to17}$ runs. Notice that $f_{7\to8}$ incurs a TCP timeout in the middle, which causes a 1-second long downward spike in its instantaneous throughput; CNA cannot, of course, completely eliminate TCP timeouts, but greatly reduces their occurrence.

Responsiveness to route change. The time interval between 340 and 380 seconds in Figure 7 demonstrates CNA's responsiveness to route changes. During this interval, two flows, $f_{7\to8}$ and $f_{4\to20}$, are present. Initially, $f_{4\to20}$ uses the route $4 \to 6 \to 20$. Around 360 seconds, we change it to $4 \to 3 \to 5 \to 20$. As the Figure 14 shows, CNA re-assigns airtime-limits, and both flows achieve relatively fair throughput. There is a short transient where air-time limits are lower than the steady-state value: during this interval, CNA has not detected the departure of the flow along the old route, but has detected the arrival of the flow on the new route.

Adapting to external interference. CNA explicitly measures external interference, and sets airtime-limits based on the available channel airtime. To demonstrate this, consider the time interval between 410 and 470 seconds in Figure 7, where there are three flows running: $f_{7\to8}$, $f_{3\to4}$ and $f_{2\to17}$. At about 420 seconds, we turned on a microwave oven (placed near nodes 1 and 2, Figure 6) for 30 seconds.

Figure 13(a) and 13(b) shows the computed available channel airtime and airtime-limits. Link $7 \to 8$ does not detect external interference, but its airtime-limit is reduced, because it is in the neighborhood of $2 \to 3$ and $3 \to 4$. Therefore, throughputs for all three flows are decreased. Note that in this period one three-hop flow $f_{2\to17}$ is competing with two one-hop flows $f_{7\to8}$ and $f_{3\to4}$. Link $3 \to 4$ gets proportionally higher airtime because it carries two flows.



(a) (b) (c) (d)

Figure 13: Adapting to external interference

For the time interval between 500 and 530 seconds, we turned on the microwave oven for 20 seconds. During this time interval, flows $f_{1\to18}$, $f_{11\to18}$ and $f_{20\to9}$ are present. As Figure 13(c) and 13(d) show, links 20 → 6 and 6 → 9 do not get reduced airtime-limits, since they are outside the neighborhood affected by the external interference. Notice though, in Figure 7, that there is throughput gap between $f_{1\to18}$ and $f_{11\to18}$. The external interference causes increased retransmissions on $f_{11\to1}$ shown in Figure 15, so the throughput of $f_{11\to18}$ is reduced. This results in unused airtime on 1 → 18, which is taken up by $f_{1\to18}$.

6. OPTIMALITY

How far off from the optimal is CNA? The most comprehensive answer to this question can be given if one computes the achievable rate region of representative, real-world mesh topologies and identifies the point corresponding to CNA on this regions. [1] Prior work [14] has developed a theoretical framework which, given information about the interference graph and the link loss rates in a topology, can compute the achievable rate region for an 802.11 mesh network. Unfortunately, this framework cannot be directly applied to real-world wireless networks, such as the ones we have used in Section 5, because in the real world, both the interference graph and the loss rate of each link change over time (as they did during our experiments). We could have attempted to use the "most likely interference graph" and the "average loss rate of a link" to compute the optimal rates. However, it is unclear whether, and to what extent, this calculation would under-estimate or over-estimate the actual achievable-rate region. So, we leave such efforts for future work, and instead use simulation to study the optimality of CNA in a number of canonical topologies for which the achievable rate region can be computed using [14]. To keep the exposition simple, we find the max-min rate allocation on the boundary of the achievable rate regions and compare the max-min rates, referred to as optimal from now on, to those of CNA. We emphasize that the max-min rates under an optimal scheduler are not the same with the max-min rates under 802.11, and that our focus is on the latter. For a comparison of these max-min rates the interested reader is referred to [15].

[1] The achievable rate region is the set of all flow-rates which do not blow up any queues. Its boundary corresponds to optimal rate allocations, e.g. the max-min rate allocation is one of these optimal points.

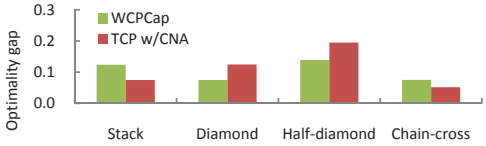


Figure 16: The Optimality Gap

We implemented CNA on Qualnet 3.9.5 [3]. In addition to comparing CNA and optimal, we also compare the rate allocation obtained by WCPCap [23], the state-of-the-art rate-control design for mesh network transport which closely approximates a max-min rate controller while being fully distributed (We have obtained the code in [23] from the authors to compute WCPCap rates.) In particular, under WCPCap, nodes estimate the available capacity in a congested region (using [14]), compute the max-min rate allocation in the region, and send these rates explicitly to the sources.

We use the four canonical topologies depicted in [23] to compare the rates achieved by TCP without CNA, WCPCap, TCP with CNA, and the optimal allocation. These topologies have been used in a number of prior works, see, for example, [23, 29, 28], and are representative of most interference scenarios occurring in mesh network topologies.

Figure 16 shows how far off CNA and WCPCap are from their respective optimal max-min rates by comparing aggregate rates for each topology. Note that although CNA allocates airtime, not rates, we compare rates achieved because in our simulations all radios run at a fixed rate, and we set the channel to be perfect.

For all the four topologies, CNA is *no further away from the optimal than WCPCap* . More surprisingly, CNA enables TCP to achieve throughputs between 5-20% of the optimal rate allocation across the topologies, and is about 12% off the optimal on average across the topologies. Although we made several conservative design choices to preserve stability (Section 3), these have not impacted performance significantly.

While these results are very promising, they do not preclude the existence of topologies where CNA is further away from the optimal. With this in mind, we briefly comment on CNA's worst case performance. Consider the stack topol-

ogy again (figure 2(a)). As already discussed, an optimal scheduler would support an airtime limit of $\frac{1}{8}$ for all links, 802.11 would support an airtime limit between $\frac{1}{12}$ and $\frac{1}{8}$, and CNA assigns an airtime limit of $\frac{1}{12}$. The difference comes from the ability of an optimal scheduler to always schedule the two outer flows concurrently, the inability of 802.11 to always do this, and our CNA design choice which conservatively assumes that 802.11 will never be able to do this. If there are more than two outer flows, the difference among the airtime limits will increase. In general, the worst case scenario that maximizes the difference consists of many edges which interfere with a common edge (the edge in the "middle") but do not interfere with each other, and the optimal scheduler is the only one which can always schedule concurrently the flows that traverse them. (See [15] for a complete discussion.) It is worth pointing out that a large imbalance in the number of flows per edge exacerbates this problem. If, for example, there are 10 flows in the upper outer branch of the stack and one flow in the lower outer branch, the lower flow suffers because CNA conservatively assumes that 802.11 will never schedule it concurrently with any of the 10 upper flows. Despite the suboptimal performance of CNA in such corner cases, we stand by our design choice on how to allocate airtime limits because it is very simple to use and implement, and it yields near-optimal performance in many real-world scenarios.

7. CONCLUSIONS

We have discussed the design and evaluation of cooperative neighborhood airtime-limiting (CNA), a mechanism that achieves efficient explicit airtime allocation in a manner transparent to TCP/IP and the 802.11 MAC. CNA's design is robust, responsive, and handles external interference, MAC-layer rate adaptation, and permits mesh TCP connections that also traverse wired links. Its performance is encouraging, being on average within 12% of the optimal on the canonical topologies we have explored.

8. REFERENCES

[1] Iperf. http://sourceforge.net/projects/iperf/.

[2] MadWifi. http://madwifi-project.org/.

[3] Qualnet. http://www.scalable-networks.com/.

[4] AKYOL, U., ANDREWS, M., GUPTA, P., HOBBY, J., SANIEE, I., AND STOLYAR, A. Joint scheduling and congestion control in mobile ad hoc networks. In *Proc. of IEEE INFOCOM* (2008).

[5] AZIZ, A., STAROBINSKI, D., THIRAN, P., AND EL FAWAL, A. Ez-flow: removing turbulence in ieee 802.11 wireless mesh networks without message passing. In *Proc. ACM CoNEXT* (2009).

[6] BAKRE, A., AND BADRINATH, B. I-TCP: indirect TCP for mobile hosts. In *Proc. of IEEE ICDCS* (1995).

[7] BALAKRISHNAN, H., SESHAN, S., AND KATZ, R. H. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks* (1995).

[8] FLOYD, S., AND JACOBSON, V. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking* (1993).

[9] FU, Z., LUO, H., ZERFOS, P., LU, S., ZHANG, L., AND GERLA, M. The impact of multihop wireless channel on tcp performance. In *IEEE Transactions on Mobile Computing* (2005).

[10] GARETTO, M., SALONIDIS, T., AND KNIGHTLY, E. Modeling Per-flow Throughput and Capturing Starvation in CSMA Multi-hop Wireless Networks. In *Proc. of IEEE INFOCOM* (2006).

[11] HOLLAND, G., AND VAIDYA, N. Analysis of TCP performance over mobile ad hoc networks. In *Proc. of ACM MobiCom* (1999).

[12] JAIN, K., PADHYE, J., PADMANABHAN, V. N., AND QIU, L. Impact of interference on multi-hop wireless network performance. In *Proc. of ACM MobiCom* (2003).

[13] JANG, K., GOVINDAN, R., AND PSOUNIS, K. Simple yet efficient transparent airtime allocation in wireless mesh networks. Tech. Rep. 915, University of Southern California, July 2010.

[14] JINDAL, A., AND PSOUNIS, K. Characterizing the Achievable Rate Region of Wireless Multi-hop Networks with 802.11 Scheduling. *IEEE/ACM Transactions on Networking* (2009).

[15] JINDAL, A., AND PSOUNIS, K. Making the Case for Random Access Scheduling in Wireless Multi-hop Networks. In *Proc. of IEEE Infocom (Mini-Conference)* (San Diego, CA, March 2010).

[16] KIM, D., TOH, C.-K., AND CHOI, Y. TCP-BuS: improving TCP performance in wireless ad hoc networks. *IEEE International Conference on Communications* (2000).

[17] KUMAR, V., M.V, M., PARTHASARATHY, S., AND SRINIVASAN, A. Algorithmic Aspects of Capacity in Wireless Networks. In *Proc. of ACM SIGMETRICS* (2005).

[18] LI, M., AGRAWAL, D., GANESAN, D., AND VENKATARAMANI, A. Block-switched Networks: A New Paradigm for Wireless Transport. In *Proc. of NSDI* (2009).

[19] LIU, J., AND SINGH, S. Atcp: Tcp for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications* (2001).

[20] LOCHERT, C., SCHEUERMANN, B., AND MAUVE, M. A survey on congestion control for mobile ad hoc networks: Research Articles. *Wirel. Commun. Mob. Comput.* (2007).

[21] MORRIS, R., KOHLER, E., JANNOTTI, J., AND KAASHOEK, M. F. The Click modular router. *SIGOPS Oper. Syst. Rev.* (1999).

[22] NEELY, M., AND MODIANO, E. Capacity and Delay Tradeoffs for Ad-Hoc Mobile Networks. *IEEE Transactions on Information Theory* (2005).

[23] RANGWALA, S., JINDAL, A., JANG, K.-Y., PSOUNIS, K., AND GOVINDAN, R. Understanding congestion control in multi-hop wireless mesh networks. In *Proc. of ACM MobiCom* (2008).

[24] SHANNON, C., MOORE, D., AND CLAFFY, K. C. Beyond folklore: observations on fragmented traffic. *IEEE/ACM Transactions on Networking 10*, 6 (2002).

[25] SINHA, P., NANDAGOPAL, T., VENKITARAMAN, N., SIVAKUMAR, R., AND BHARGHAVAN, V. WTCP: a reliable transport protocol for wireless wide-area networks. *Wireless Networks* (2002).

[26] SUNDARESAN, K., ANANTHARAMAN, V., HSIEH, H.-Y., AND SIVAKUMAR, R. ATP: A Reliable Transport Protocol for Ad Hoc Networks. *IEEE Transactions on Mobile Computing* (2005).

[27] TAN, K., JIANG, F., ZHANG, Q., AND SHEN, X. Congestion Control in Multihop Wireless Networks. *IEEE Transactions on Vehicular Technology* (2006).

[28] WARRIER, A., JANAKIRAMAN, S., HA, S., AND RHEE, I. DiffQ: Differential Backlog Congestion Control for Multi-hop Wireless Networks. In *Proc. of IEEE INFOCOM* (2009).

[29] XU, K., GERLA, M., QI, L., AND SHU, Y. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proc. of ACM MobiCom* (2003).