# Minimising RTT across homogeneous 802.11 WLANs with CAIA Delay-Gradient TCP (v0.1)

Naeem Khademi*, Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 121113A
Swinburne University of Technology
Melbourne, Australia
naeemk@ifi.uio.no, garmitage@swin.edu.au

***Abstract*—Common loss-based TCP algorithms (such as NewReno) are known to induce significant additional latency over 802.11 wireless LANs (WLANs), negatively impacting multimedia traffic sharing the same network. Using live experimental trials (under FreeBSD 9.0 on the *Emulab* 802.11 testbed) we show that CAIA's delay-gradient TCP (CDG) version 0.1 induces two to seven times lower latency than NewReno and CUBIC in homogeneous 802.11 WLAN environments while achieving equivalent goodput. This is beneficial in 802.11 WLAN environments where TCP and multimedia traffic must coexist and the network administrators have control over each end-host's choice of TCP algorithm.**

## I. Introduction

Keeping end-to-end network latency low is a crucial factor for the reliable and convenient use of multimedia applications such as VoIP, audio/video streaming and conferencing and online interactive gaming. In recent years such applications have become increasingly popular, particularly on 802.11-assisted portable devices, such as laptops, smart-phones, tablets, etc. A major challenge is managing the coexistence of, and inevitable conflict between, multimedia traffic and applications using TCP (transmission control protocol) [1] over wired and wireless paths. Here we report on a new approach to TCP congestion control that noticeably improves coexistence between TCP and multimedia flows over 802.11 wireless LAN (WLAN) environments.

Interactive multimedia applications typically require network round trip times (RTT) well under 300 ms. For example, ITU-T G.114 recommends a maximum of 100∼150 ms one-way latency within which *most* applications would not be significantly affected [2]. Studies of online first person shooter (FPS) games suggest tighter RTT limits under 150-180 ms [3], [4]. Unfortunately, these latency limits are easily exceeded when traditional TCP and multimedia traffic share bottlenecks (such as 802.11 access points and/or consumer internet gateways).

Traditional *loss-based* TCP congestion control (as used by NewReno [5] and CUBIC [6]) is known to introduce significant fluctuations in RTT for all flows sharing a common bottleneck queue (congestion point) in the network. The bottleneck may be caused by simple bandwidth mismatch (such as a home LAN feeding into a lower speed consumer internet connection), wireless access contention (as experienced by hosts utilizing 802.11 WLAN services in homes, enterprises or public hotspot) or a blend of both. RTTs have been observed jumping by 100s of milliseconds [7], [8] to multiple seconds [9], [10], depending on the amount of buffering in the bottleneck device.

Coexistence in 802.11 WLAN environments is a challenge. Packets can be delayed (buffered) due to the distributed control function (DCF), channel access contention levels with other clients, noise and interference levels, choice of modulation and coding schemes (MCS) and choice of bit-rate selection (a.k.a "rate adaptation") mechanisms. RTT will rise significantly due to contention in either direction, but contention and rate adaptation has been seen to increase RTT more significantly for uplink TCP flows (towards the access point) [11], [12], [13].

An alternative class of *delay-based* TCP algorithms exist that utilize trends in observed RTT (rather than packet losses) to infer the onset of congestion along an end-to-end path. Beginning with Jain's CARD [14] in 1989, such algorithms can optimize their transmission rates *without* forcing the filling of buffers to induce packet losses. By not filling buffers, delay-based TCPs induce significantly lower queuing delays than loss-based TCPs, regardless of how much actual buffer space is available at the bottleneck. However, many delay-based

*Networks and Distributed Systems Group, Department of Informatics, University of Oslo, Norway

TCP techniques do not fare well in the presence of packet loss or when sharing a bottleneck with (i.e. competing against) loss-based TCP in 'lightly multiplexed' cases (where only a handful of TCP flows coexist at the same time).

CAIA's recently published and implemented *delay-gradient* TCP called CDG[1] shows signs of improved performance in the presence of non-congestion packet losses (a useful trait for wireless networks), and improved coexistence with loss-based TCPs in wired networks (a useful trait in realistic scenarios) while still keeping induced RTTs low [15], [16]. CDG is a sender-side modification that works with all conventional TCP receivers.

In this paper we significantly extend [15] by experimentally comparing the behaviour of CDG, NewReno and CUBIC flows in Emulab's 802.11 WLAN environment [17], [18], using [15]'s FreeBSD implementation of CDG version 0.1 [16] under different WLAN contention and rate-adaptation conditions. To our knowledge CDG has not previously been evaluated in any 802.11 scenarios.

We see positive results – while achieving similar goodput, CDG induces additional path latency roughly twofold to fourfold smaller than NewReno and CUBIC in homogeneous WLAN to WLAN scenarios, and reduces the latency of uplink TCP traffic up to sevenfold. CDG appears worthwhile trialling in environments where that mix TCP bulk data transfers and multimedia traffic, and the end-hosts can be constrained to use a particular TCP stack. Our results also provide incentive for future work to explore the more challenging case where CDG flows coexist with loss-based TCPs in WLANs.

The rest of our paper is structured as follows. Section II provides more background on how TCP induces significant swings in additional RTT, summarizes some past work on delay-based TCP algorithms, and highlights the various influences on TCP traffic over 802.11 WLAN links. Section III presents the key technical design elements of CDG from [15]. Our experimental testbed is described in Section IV, with Section V covering our performance evaluation and results. The paper concludes in Section VI

## II. Background

Before describing our evaluation of CDG, we review the reasons why conventional TCP induces significant swings in RTT, summarize past work on delay-based TCP algorithms, and highlight the various influences on TCP traffic over 802.11 WLAN links.

[1]A working name derived from "CAIA Delay Gradient"

### A. Challenges for TCP congestion control

A key goal of TCP is to expedite the reliable transfer of byte-streams across the IP layer's packet-based service while minimizing congestion inside both end hosts and the underlying IP network(s) [19]. TCP minimizes congestion by continually adjusting the number of packets 'in flight' (sent but not yet acknowledged) – allowing more whenever the path appears capable of handling more traffic, and sending less when it senses the network path is congested (overloaded).

More specifically, TCP receivers regularly advertise a receive window (`rwnd`) back to the TCP sender to indicate how much data it can buffer, and TCP senders continually re-calculate a congestion window (`cwnd`) as an estimate of how much data the network path can absorb. The number of packets allowed by the sender to be in flight at any given time is the smaller of `cwnd` and `rwnd`. Congestion control (CC) algorithms typically focus on how to adjust `cwnd` when congestion is sensed in the network.

*1) Loss-based TCP and bufferbloat:* Common TCP algorithms, such as NewReno and CUBIC, utilize *loss-based* CC – they treat IP packet loss as an indicator of network or end-host congestion. The underlying assumption is that `cwnd` can grow while ever packets are getting through, and should shrink if packets are lost when a bottleneck along the path runs out of space to temporarily buffer packets.

This creates two key problems. Loss-based TCP can be misled by (and perform poorly across) paths or links that exhibit intrinsic (non-congestion related) packet losses, such as wireless LANs. Loss-based TCP also contributes to significant additional per-hop queuing delays (and hence overall RTT) by cyclically growing `cwnd`, filling the bottleneck buffer(s), triggering packet drop(s), shrinking `cwnd` and starting all over again. These cyclical RTT spikes cause collateral damage to latency-sensitive applications (such as UDP-based Voice over IP, or online games) sharing the same bottlenecks [7], [8].

Excessive queuing delays have also recently gained broader notoriety under the banner of *Bufferbloat* [9], [20] – an observation that over recent years, vendors have dramatically increased the level of buffering almost everywhere a queue might form (such as routers, switches, network device drivers, etc). In conjunction with loss-based TCPs, queue sizes in modern equipment have been observed adding multiple seconds of additional queuing delay on typical consumer internet paths [9], [10]. Worse still, bufferbloat degrades the timeliness of congestion

notifications required by loss-based TCP's control loop.

*2) Delay-based TCP and reduced queuing delays:* Proposals for using *delay-based* congestion indications for TCP have been around since Jain's CARD (Congestion Avoidance using Round-trip Delay) in 1989 [14]. Broadly speaking, delay-based TCP algorithms utilize trends in observed RTT to infer the onset of congestion along an end-to-end path. By inferring the onset of queue build-up at a bottleneck, delay-based algorithms can regulate their transmission rates well before the bottleneck queue fills and starts dropping packets. Since queues need not fill up, additional queuing delays are also kept low regardless of the amount of buffering built-in at each bottleneck point.

Many variations have emerged since [14], differing in the way they measure delay (RTT, one-way delay, per-packet measurements, etc), how they infer congestion (set thresholds, etc), and how they adjust the sender's congestion window (`cwnd`) in response to congestion.

*3) Delay-based TCP: An exceedingly brief survey:* We will provide a brief summary of previous delay-based TCPs from [15] using the following terminology: $\beta$ is the multiplicative decrease factor for Additive Increase Multiplicative Decrease (AIMD), $\theta$ represents a delay threshold, $\tau_i$ is the $i^{\text{th}}$ RTT measurement, $\tau_{\min}$ is the smallest RTT, $\tau_{\max}$ is the largest RTT, and $d_i$ is $i^{\text{th}}$ one-way delay.

CARD [14] utilizes the normalized delay gradient of RTT, $\left(\frac{\tau_i-\tau_{i-1}}{\tau_i+\tau_{i-1}}\right) > 0$, to infer congestion, and AIMD ($\beta = \frac{7}{8}$) to adjust `cwnd`. DUAL [21] uses $\tau_i > \frac{(\tau_{\min}+\tau_{\max})}{2}$ with delay measured every 2$^{\text{nd}}$ RTT, and AIMD ($\beta = \frac{7}{8}$) to adjust `cwnd`.

Vegas [22] uses $\tau_i > \tau_{\min} + \theta$ (normalized by the data sent), delay measured every RTT and Additive Increase Additive Decrease (AIAD) to adjust `cwnd`. Fast TCP [23], TCP-Africa [24] and Compound TCP (CTCP) [25] all infer congestion similarly to Vegas, but use smoothed RTT measurements and differ in how they adjust `cwnd` upon detecting congestion.

Probabilistic Early Response TCP (PERT) [26] uses dynamic thresholds based on inferred queuing delay ($q_j = \tau_j - \tau_{\min}$) using smoothed RTT measurements and probabilistic reaction to queuing delay inspired by Random Early Discard (RED), with loss probability matching when $q_j \geq 0.5 q_{\max}$. TCP-LP [27] uses $\overline{d}_i > d_{\min} + \delta(d_{\max} - d_{\min})$, based on smoothed one-way delay ($d$) using TCP time stamps, and AIMD with a minimum time between successive window decreases to adjust `cwnd`.

An unnamed algorithm from the Hamilton Institute [28], [29] and a more loss-tolerant variant from CAIA [30], implement probabilistic adjustment of `cwnd` based on queuing delay, RTT thresholds and a back-off function.

*4) Delay-gradient TCP:* A number of weaknesses in previous delay-based TCPs motivated the recent development of a *delay-gradient* TCP with the working name of *CDG* ("CAIA Delay Gradient") [15].

Some delay-based TCP techniques do not fare well in the presence of packet loss, or do not fare well when competing against aggressive loss-based TCPs in *lightly multiplexed* scenarios (where only a handful of TCP flows are competing, such as in last-mile consumer links).

Many previous delay-based TCPs are better characterised as *delay-threshold* algorithms – they infer congestion when path delays hit or exceed certain thresholds. Unfortunately, meaningful thresholds are hard to set if little is known about the network path that packets will take. In addition, competing delay-threshold flows can suffer relative unfairness if their inability to accurately estimate a path's base RTT (the smallest possible RTT for the path) leads to thresholds established on the basis of different estimates of base RTT [31].

In contrast, CDG makes use of relative variations (gradients) in RTT over time to infer congestion, adapts to conditions along the paths each flow takes and needs no a priori knowledge of a path's minimum or typical RTT levels. The preliminary analysis in [15] also reveals improved tolerance to intrinsic (non-congestion related) packet losses, and improved sharing of capacity with traditional NewReno [5] in lightly multiplexed environments. CDG is a sender-side modification that works with all conventional TCP receivers.

### B. Considerations for 802.11 WLANs

Relative to wired networks, the behaviour of 802.11 networks is further complicated by their medium access methods. Potential frame collisions and random back-off times due to contention, and their consequent frame re-transmissions, increases network access times. Contention levels go up as the number of concurrent hosts goes up, increasing the latencies experienced by all parties.

*1) Rate adaptation:* 802.11 networks use a variety of modulation and coding schemes (MCSs), such as 802.11g's range from 6 Mbps $\sim$ 54 Mbps. Rate adaptation (RA) mechanisms for bit-rate selection ("rate control") are vendor-specific, with SampleRate [32] and

Minstrel [33] being the most widely used RA mechanisms in today's 802.11 devices [11]. The RA triggers the MCSs yielding lower bit-rates in presence of noise and interference and may also react to contention. This in turn may lead to a sub-optimal performance and further increase in delay depending on the traffic scenario and RA mechanism being used. The study of different RA mechanisms, however is out of this paper's scope and has been carried out in [11]. A delay-based congestion control mechanism should also be able to react to the MAC-level induced delay.

*2) Uplink versus downlink:* When traffic flows predominantly through the access point (AP) towards wireless clients, the AP controls and coordinates downlink transmission of all (usually large) TCP Data packets. Consequently collisions are either ACK-Data or ACK-ACK making it less wasteful of the channel compared to Data-Data collisions. With $n$ stations an AP's channel access success is of order $O(1/n)$, indicating the equal chance as any other competing station. Generation of an ACK packet is always a consequence of a successful data packet transmission at the AP. Therefore there will be fewer TCP receivers having their upstream queues backlogged with ACK packets at any instance of time. The number of active stations contending to access the channel normally does not grow beyond 2∼3 stations at any instance of time when $n < 100$ [34], [35].

However, TCP's uplink performance suffers significant latency increases in scenarios with long-lived uploading flows [11], [13], [10]. Channel-wasting Data-Data collisions are more common (of order $O(\frac{n-1}{n})$) as all uploading stations are actively participating in contention. The upstream queues of each station are filled by TCP flows growing their individual congestion windows, resulting in increased collision probability and in turn much higher latency levels compared to download scenarios.

## III. Technical summary of CDG version 0.1

CDG is a sender-side modification to TCP. To assist the reader's understanding this section summarizes key design elements originally specified in [15]. Throughout this section the congestion window `cwnd` will be represented by $w$. CDG utilises two multiplicative decrease factors – one for delay-gradient back-off and the other for loss-based back-off (conventionally labelled $\beta$). Although not labelled as such in [15] we will refer to them here as $\beta_d$ and $\beta_l$ respectively.

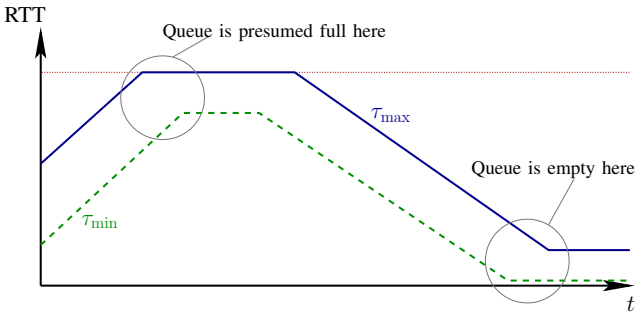



Fig. 1. Tracking the RTT measurement envelope to detect queue growing, full, draining and empty states

### A. Constructing a delay gradient signal and characterising packet losses

CDG tracks the envelope ($\tau_{\max}$ and $\tau_{\min}$) of estimated RTTs seen within a measured RTT interval, and constructs two gradient signals from the changes in $\tau_{\max}$ and $\tau_{\min}$ seen across successive RTT intervals. Given $n$ is the $n$th RTT interval, the gradients are calculated as:

$$g_{\min,n} = \tau_{\min,n} - \tau_{\min,n-1} \quad (1)$$

$$g_{\max,n} = \tau_{\max,n} - \tau_{\max,n-1} \quad (2)$$

Figure 1 illustrates CDG's assumption that when a queue fills to capacity, $\tau_{\max}$ stops increasing before $\tau_{\min}$ stops increasing. A queue draining to empty is indicated when $\tau_{\min}$ stops decreasing before $\tau_{\max}$.

CDG further smooths these two noisy gradient signals with an iterative moving average based on Equation 3.

$$\bar{g}_n = \bar{g}_{n-1} + \frac{g_n - g_{n-a}}{a} \quad (3)$$

In Equation 3, $a$ represents the number of samples in the moving average window, $g_i = g_{\min,i}$ for calculating $\bar{g}_{\min,n}$ and $g_i = g_{\max,i}$ when calculating $\bar{g}_{\max,n}$. (Note that when operating in slow start mode $g_{\min,n}$ and $g_{\max,n}$ are used without smoothing for a more timely response to rapid $w$ increases.)

Both $\bar{g}_{\min}$ and $\bar{g}_{\max}$ are used to infer the path's bottleneck queue state[2]. CDG infers that $Q$ = full, and lost packets may be interpreted as a congestion signal, when $\bar{g}_{\max}$ drops to zero before $\bar{g}_{\min}$ ($\tau_{\max}$ stops growing before $\tau_{\min}$ in Figure 1). Otherwise, aside from requiring re-transmission, a lost packet is assumed to not indicate congestion.

CDG relies on FreeBSD's enhanced RTT measurement module [36] to obtain live RTT measurements without the noise caused by duplicate acknowledgements, segmentation offload and sack.

[2] $Q \in$ {full, empty, rising, falling, unknown}

### B. Delay gradient window progression

In congestion avoidance mode, CDG increments $w$ by one MSS once every RTT unless the RTT gradient is positive, in which case there is a finite probability CDG will back off. This is expressed succinctly in Equation 4

$$w_{n+1} = \begin{cases} w_n \beta_d & X < P[\text{back-off}] \wedge \bar{g}_n > 0 \\ w_n + 1 & \text{otherwise} \end{cases} \quad (4)$$

where $w$ is the congestion window in packets[3], $n$ is the $n$th RTT interval, $X = [0, 1]$ is a uniformly distributed random number, $P[\text{back-off}]$ is a back-off threshold (section III-B1), and $\beta_d$ is the multiplicative decrease factor for delay-gradient back-off ([15] used $\beta_d = 0.7$).

Since the effect of an Equation 4 back-off will not be measured in the next RTT interval, the next calculation of $g_{\min}$ or $g_{\max}$ is ignored. Thus a delay-gradient congestion indication will cause CDG to back off at most once every two RTT intervals[4].

Growth of $w$ in slow start mode mimics that of NewReno. The decision to reduce $w$ and enter congestion avoidance mode is made per RTT as per Equation 4, or on packet loss as in NewReno, whichever occurs first.

*1) RTT-independent back-off:* The $X < P[\text{back-off}]$ term in Equation 4 provides an exponential, probabilistic back-off which aims to achieve fairness between flows having different base RTT. $P[\text{back-off}]$ is given by Equation 5.

$$P[\text{back-off}] = 1 - e^{-(\bar{g}_n / G)} \quad (5)$$

were $G > 0$ is a scaling parameter and $\bar{g}_n$ will either be $\bar{g}_{\min,n}$ or $\bar{g}_{\max,n}$.

A source with a small RTT (which sees smaller differences in the RTT measurements) will have the same average $P[\text{back-off}]$ of a source with a longer RTT (which sees larger differences in the RTT measurements). CDG's FreeBSD implementation uses a lookup table for $e^x$ and a configurable FreeBSD kernel variable for $G$.

*2) Ineffectual back-off detection:* Equation 4's delay-gradient back-offs are presumed to be ineffectual if $\bar{g}_{\min}$ or $\bar{g}_{\max}$ are still not negative after CDG has backed-off multiple times, $b$. CDG then presumes it is competing with one or more loss-based flow(s) and suppresses further delay-gradient back-offs until either $b'$ further delay gradient congestion indications have occurred or one of $\bar{g}_{\min}$ or $\bar{g}_{\max}$ have become negative during this period. Both $b$ and $b'$ are configurable kernel variables in CDG's FreeBSD implementation.

[3]CDG increments a byte-based $w$ by the maximum segment size every RTT.

[4]TCP Vegas [22] uses a similar idea during slow start

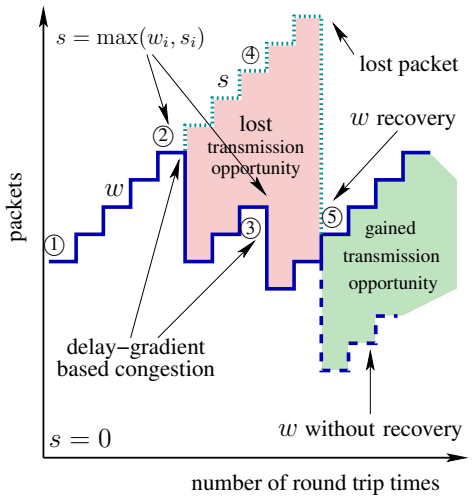



Fig. 2. Behaviour of shadow window ($s$) and congestion window ($w$) when competing with loss-based TCP flows.

### C. Updating congestion window on packet loss

Packet loss triggers two possible outcomes. If the path bottleneck queue is presumed full (section III-A), CDG uses the concept of a *shadow window* ($s$) from [30] to control how $w$ is adjusted. Otherwise $w$ is left unchanged.

Succinctly, when a packet loss occurs, $w$ is updated as follows:

$$w_{i+1} = \begin{cases} \max(s_i, w_i)\beta_l & Q = \text{full} \wedge \text{packet loss} \\ w_i & \text{otherwise} \end{cases} \quad (6)$$

where $\beta_l = 0.5$ (the classic NewReno-like loss-based multiplicative decrease factor). The shadow window $s$ represents a NewReno-like alternative to $w$ that has not been decreased by intervening delay-gradient back-offs. It allows CDG to "even the playing field" when confronted by bottleneck queues being filled by loss-based TCPs. Figure 2 illustrates the relationship between $s$ and $w$ over time. Referring to the regions indicated by circled numbers:

1) $w$ grows as normal for TCP congestion avoidance (one packet per RTT)
2) Equation 4's delay-gradient congestion criteria is met, $s$ is initialized to $\max(w_i, s_i)$ and $w$ is set to $w\beta_d$
3) $w$ continues to react to delay-gradient congestion indications

4) $s$ shadows NewReno-like growth while $w$ reacts to delay-gradient congestion indications
5) A packet loss occurs ($Q$ = full), so $w$ is set to $s\beta_l$ rather than $w\beta_l$ (where $\beta_l = 0.5$)

The shadow window is re-initialized by a number of possible events. A delay-gradient back-off causes $s_{i+1} = \max(w_i, s_i)$, if CDG guesses a bottleneck queue has emptied then $s_{i+1} = 0$, and in all other cases $s$ is unchanged (growing NewReno-like per Figure 2, and capped at the maximum window size for the connection).

Using [30]'s shadow window concept improves CDG's coexistence with loss-based flows. Lost transmission opportunities are not reclaimed, but the impact of the extra delay-gradient based back-offs is lessened.

## IV. Experimental Setup

Our goal is to evaluate the goodput and latency experienced by 802.11 WLAN networks running CDG version 0.1 [16] and compare these to similar networks running NewReno [5] (the 'standard' TCP) or CUBIC [6] (default in the Linux kernel). Performance is to be evaluated as a function of channel contention levels, 802.11 rate adaptation being on or off and direction of the traffic (uplink vs. downlink).

This section outlines our experimental approach, conducted using live trials on *Emulab* [17] (an open networking test-bed located at the University of Utah, providing access to a variety of hardware for researchers). The PC nodes used in our experiments are in a cluster of concentrated nodes stacked in lab environment, where their close proximity provides a suitable situation for the experiments that require homogeneous channel conditions.

Each wireless node is an Intel PIII 600 MHz with 512 MB RAM, an Atheros-based AR5413 802.11 chipset, FreeBSD 9.0 kernel and standard Atheros driver operating in 2.4 GHz 802.11b/g mode for indoor activities. Our experiments are conducted with no network operating on the overlapping channels, thus minimizing the potential interference from the neighbouring networks.

Figure 3 depicts our network – two different WLANs (operating on different channels) whose respective access points (APs) are connected via two intermediate routers and 100 Mbps links (a dumbbell topology). Sending hosts are associated with one WLAN, receiving hosts with the other.

We used *dummynet* [37] and *ipfw* on Router #1 to emulate a wired path between the WLANs having one-way propagation delay of 20 ms (for a minimum path RTT of 40 ms). Dummynet's buffer was configured to

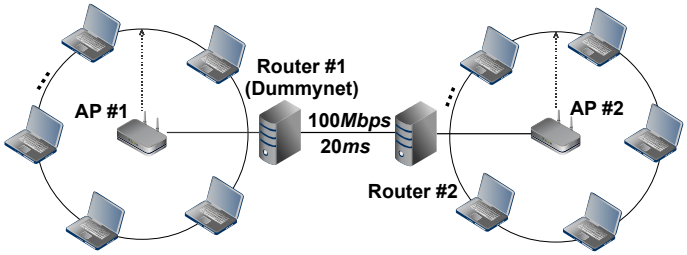



Fig. 3. A dumbbell network topology with 802.11 end-nodes

absorb at least 20 ms of artificially delayed traffic at 100 Mbps, to ensure the the WLANs remained the true bottlenecks.

Each experiment is a 60 sec run of TCP traffic from *iperf* repeated 10 times. All nodes are synchronized in time using *ntp*. TCP's send and receive buffers were set large enough (send socket buffer=1588 KB and receive socket buffer=3176 KB) so TCP's *cwnd* could grow enough to ensure maximum link utilization. The Maximum Segment Size (MSS) was set to 1448 bytes (MTU=1500 bytes).

We captured all traffic using *tcpdump* on the outgoing and incoming interfaces of each WLAN station. From this we passively extracted per-packet RTT statistics (using the *Synthetic Packet Pairs* (SPP) tool [38]) and TCP goodput (using *tcptrace* [39]). Some experiments involved injecting previously captured VoIP traffic on the wireless interfaces using *tcpreplay* [40].

To confirm that all trial runs had a similar channel condition pattern, we also measured the RSSI of each frame carrying TCP payload at its respective receiving node. Our detailed RSSI measurements show that all trial tests had a similar RSSI pattern with the median RSSI values of all frames with TCP payload captured by the NICs varying in the range of -47 dB∼-37 dB with 5th percentile being in range of -30 dB∼-38 dB and 95th percentile being in range of -42 dB∼-56 dB.

## V. Performance Evaluation

In this section, we present the results of our experimental evaluations. We look at five scenarios – different numbers of identical stations sending concurrently from one WLAN to the other (Section V-A), performance with rate-adaptation turned on or off (Section V-B), performance when traffic is primarily in the uplink or downlink directions (Section V-C) and impact on VoIP intermingled with TCP traffic (Section V-D).

Table I summarises the CDG parameters used during most of our experiments. Aside from $\beta_d$ we use default

values from [15]. As noted in [15], tuning $\beta_d$ higher can provide more goodput at the risk that delay-gradient back-offs will be less effective in keeping queuing delays down. For the homogeneous scenarios (where every station was using CDG, NewReno or CUBIC) we explored the potential downside to CDG's goodput of tuning $\beta_d$ down to the 'classic' value of 0.5 used for NewReno's loss-based back-off ($\beta_l$).

TABLE I
DEFAULT CDG PARAMETERS SETTING

| parameter | value |
|---|---|
| delay-based $\beta_d$ | 0.5 |
| loss-based $\beta_l$ | 0.5 |
| back-off exponent ($G$) | 3 |
| $AI$ factor | 1 |
| smoothing factor ($a$) | 8 |
| ineffectual consecutive back-offs ($b$) | 5 |
| hold ineffectual back-off ($b\prime$) | 5 |

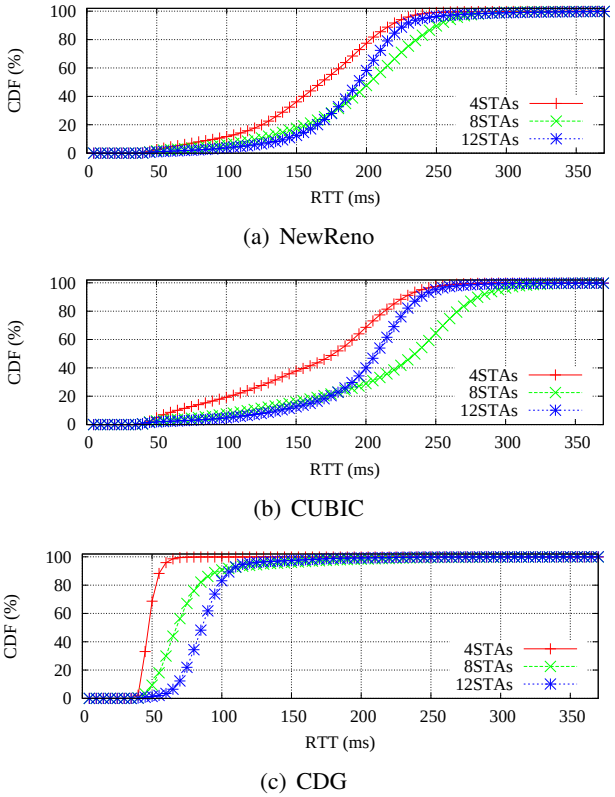
### A. Contention Level

We first explore the performance of CDG under different 802.11 MAC contention levels (Section II-B) using the methodology from Section IV. For CDG, NewReno and CUBIC in turn we ran, for 10 times, a 60 sec single TCP flow per-station with 4, 8 and 12 contending 802.11 stations on each side. Each sender (on the first WLAN) transmitted their 60 sec flows to a corresponding peer on the other WLAN (Figure 3).

Figure 4 shows that CDG induces significantly lower RTTs than either NewReno or CUBIC with 4, 8 and 12 stations. The median RTT of all NewReno flows' packets are 170 ms, 205 ms and 195 ms respectively, they slightly increase to 180 ms, 235 ms and 210 ms for CUBIC. In contrast, CDG induces median RTTs of 45 ms, 65 ms and 85 ms respectively.

The spread of RTTs in Figure 4 also reveals that CDG flows experience noticeably smaller RTT *variations* than NewReno and CUBIC flows. Figure 5 illustrates this further with a fragment of the time-varying RTTs of an individual CDG or NewReno flow in the four-station (4STA) scenario. NewReno induces a somewhat cyclical (and high) RTT while CDG's (relatively low) induced RTT looks more like line-noise.

Figure 6 shows that while CDG was achieving noticeably lower RTTs, it was also achieving broadly similar (and more consistent) goodput to that experienced by NewReno and CUBIC flows (calculated per-station using a 5 sec time window). The "stability" of CDG's goodput (evidenced by somewhat more vertical lines in the CDG goodput distributions) is a favourable feature for TCP under 802.11 networks.



(a) NewReno

(b) CUBIC

(c) CDG

Fig. 4. RTT distribution of NewReno, CUBIC and CDG under different 802.11 contention levels

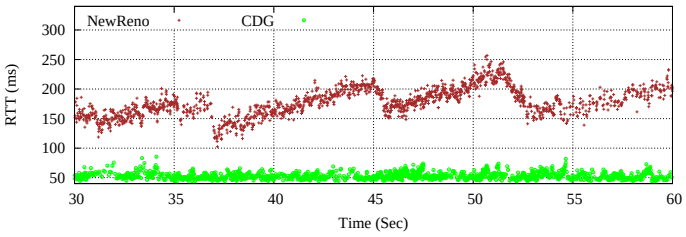
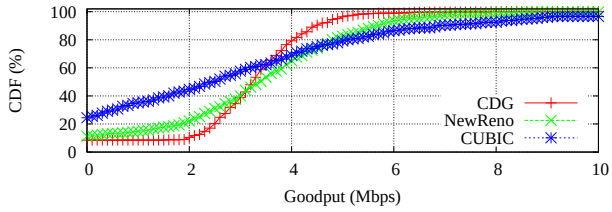


Fig. 5. RTT vs time for a single flow 4STAs over 802.11 – CDG lower and less cyclical than NewReno

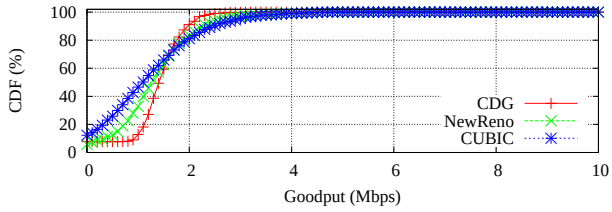
Overall CDG reduces RTT by 2.3x∼3.8x compared to NewReno and by 2.5x∼4x compared to CUBIC under medium-to-high contention levels without penalizing observed goodput.

### B. Rate Adaptation

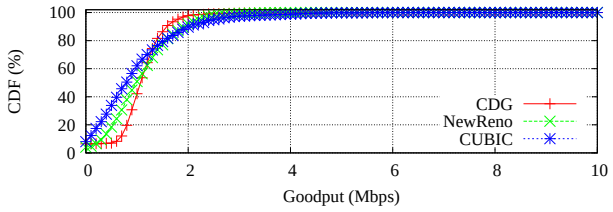
Next we explore the impact of 802.11 rate adaptation (RA) being enabled or disabled (Section II-B1). In this case our experiment used eight peer nodes. We used the SampleRate [32] RA mechanism (which happens to be the default RA of Atheros-based chipset in FreeBSD




(a) 4 STAs

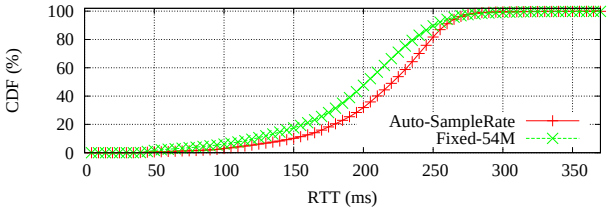



(b) 8 STAs

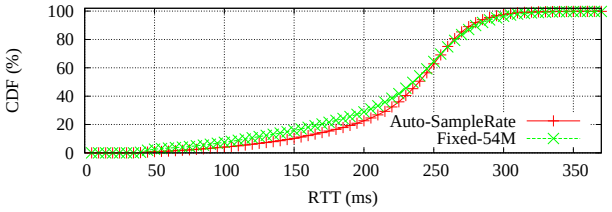



(c) 12 STAs

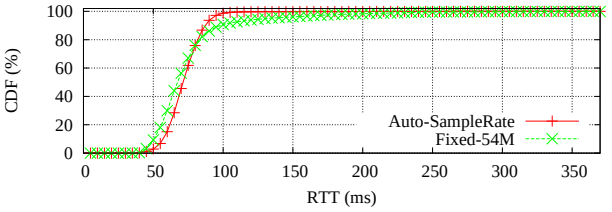
Fig. 6. Distribution of per-station goodput (measured over 5 sec intervals) for NewReno, CUBIC and CDG under different 802.11 contention levels




(a) NewReno




(b) CUBIC




(c) CDG

Fig. 7. RTT distribution of NewReno, CUBIC and CDG with RA (SampleRate) enabled/disabled

9.0). Figure 7 shows the RTT distributions observed with RA enabled ("Auto-SampleRate") or disabled ("Fixed-54M") when running CDG, NewReno and CUBIC respectively.

Enabling RA results in a slight increase in RTT for all three algorithms. Overall CDG shows significantly lower induced RTT whether RA is enabled or disabled (median RTT improvement of 3.14x and 3.15x compared to NewReno and 3.42x and 3.61x compared to CUBIC). Our tests used a relatively medium-to-low noise-level lab environment but we expect the same pattern to persist when the noise level increases.

### C. Uplink vs. Downlink

As noted in Section II-B2, TCP performance can be quite different in the uplink and downlink directions (to and from the access point). Our next experiments took eight WLAN stations on one WLAN and used iperf to send traffic (upload to) or receive traffic (download from) eight matching *wired* hosts on the other side of Router #1 (with dummynet adding 20ms delay in each direction). The WLAN would thus be loaded in the uplink (towards AP) or downlink (from AP) directions respectively.

Figure 8(a) shows the download results. NewReno, CUBIC and CDG induce median RTTs of 125 ms, 130 ms and 50 ms respectively (with CDG adding only 10 ms to the $RTT_{base}$ of 40 ms). CDG flows also experience noticeably smaller RTT variations than NewReno and CUBIC flows.

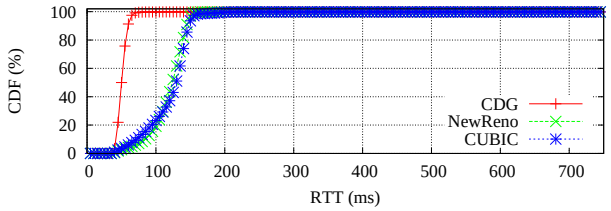
Figure 8(b) shows the upload scenario. NewReno, CUBIC and CDG induce median RTTs of 315 ms, 270 ms and 45 ms respectively with RA disabled[5]. In other words, CDG reduces RTT by 7x and 6x relative to NewReno and CUBIC respectively. In addition, NewReno and CUBIC flows experience significantly more variation between maximum and minimum RTTs than CDG flows.

By being responsive to the changes in delay-gradient mainly caused by the increased MAC-level induced delay (e.g. increased channel access time and potential collisions), all stations with CDG tend to keep their *cwnd* conservatively at an optimal level where the delay-gradient stays in the region of negative to zero. This decreases the collision probability at the MAC layer,

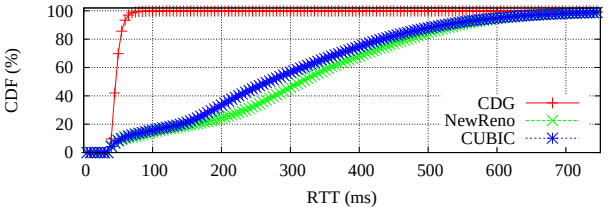
[5]As Section V-B notes, a more realistic deployment with RA enabled would likely see slightly higher RTTs

resulting in lower observed RTTs. This contrasts with the loss-based TCPs, where *cwnd* increases without regard for uplink contention delays, and hence suffer much higher RTTs in the upload scenarios.

Figure 9 shows the CDF of per-station goodput for both download and upload scenarios. For downloads NewReno, CUBIC and CDG achieve roughly similar median goodputs of 3 Mbps, 2.8 Mbps and 3 Mbps respectively. For uploads NewReno, CUBIC and CDG achieve quiet different median goodputs of 2.9 Mbps, 1 Mbps and 3.2 Mbps per station respectively (with CUBIC's aggressive behaviour leading to quite poor per-station performance). In both scenarios CDG flows experience far more stable goodput over time than either NewReno or CUBIC flows.




(a) Download




(b) Upload

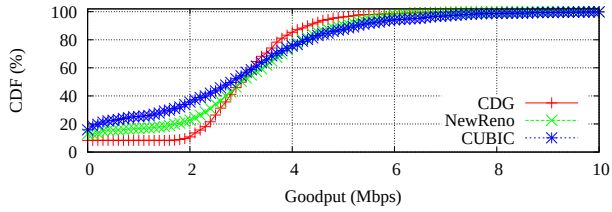
Fig. 8. RTT distribution of NewReno, CUBIC and CDG for downloads and uploads

### D. Coexisting VoIP traffic

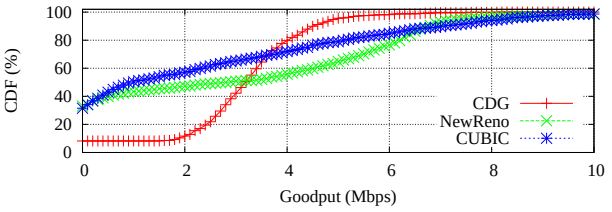
We next evaluate the impact of CDG, NewReno and CUBIC respectively on concurrent VoIP-like flows sharing a bottleneck WLAN AP. We consider two scenarios: 1) Both senders and receivers are 802.11 nodes; 2) TCP flows are of uplink type, with eight WLAN stations and RA disabled. Each sender/receiver station pair are also configured to transfer a 60 sec bi-directional VoIP-like flow consisting of UDP packets in 200 byte Ethernet frames with 20 ms inter-arrival times.

Figure 10 shows the distribution of RTTs experienced by VoIP traffic when coexisting with CDG, NewReno and CUBIC[6]. When both ends are 802.11 (Figure 10(a)),

[6]Although one-way delay is a common metric for UDP flows, we present RTT for comparability with our TCP results




(a) Download




(b) Upload

Fig. 9. Distribution of per-station goodput (measured over 5 sec intervals) for NewReno, CUBIC and CDG in uplink and downlink directions

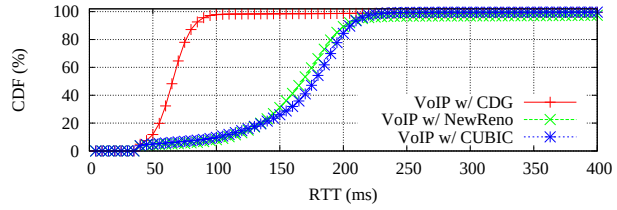
CDG improves the median latency experienced by VoIP packets to 65 ms from NewReno's 170 ms and CUBIC's 175 ms. When competing with uploads (Figure 10(b)) CDG also improves the VoIP flows' median RTT to 45 ms from NewReno's 195 ms and CUBIC's 130 ms. Overall CDG reduces the RTT relative to NewReno and CUBIC by 2.6x and 2.7x in the "802.11 both ends" scenario, and 4.3x and 3x in upload scenario.

Figure 11 shows per-packet jitter distribution of VoIP flows. Here, we define per-packet jitter as the difference between the RTTs of each two consecutively transmitted VoIP packets. It is observable that in upload scenario VoIP flows coexisting with CDG traffic experience a lower per-packet jitter level than other TCP variants.

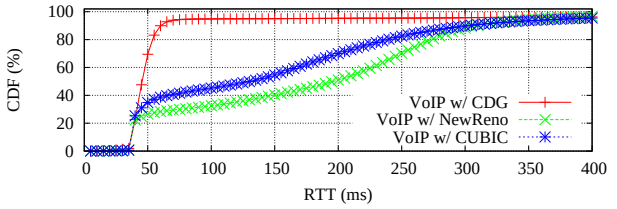
## VI. Conclusion

We have explored the potential for CAIA's delay-gradient TCP algorithm (CDG [15]) to enable improved coexistence between multimedia traffic (such as delay-sensitive VoIP, online gaming and audio/video conferencing) traffic and TCP traffic in 802.11 WLAN scenarios. Common loss-based TCP algorithms, such as NewReno [5] and CUBIC [6], are known (and shown) to induce significant levels of additional latency when running over 802.11 WLANs. The consequent increase in round trip time (RTT) across the WLAN negatively impacts on any multimedia traffic sharing the network.

Using live trials on *Emulab* [17] with a FreeBSD 9.0 implementation of CDG we evaluated the goodput and latency achieved by 802.11 WLAN networks running CDG, NewReno and CUBIC, evaluating performance

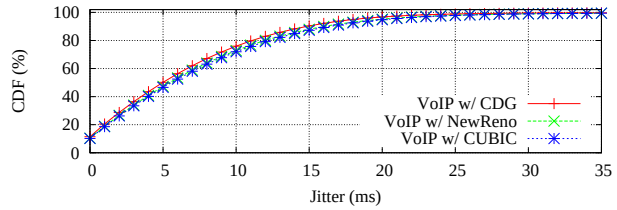



(a) 802.11 both end-nodes

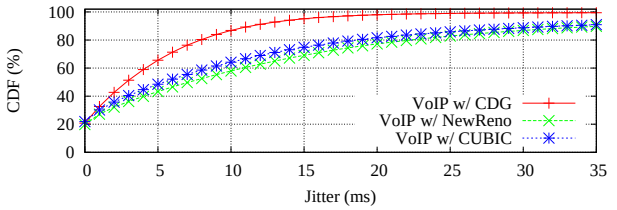



(b) Uplink

Fig. 10. Two-way delay distribution of coexisting VoIP flows (CDF)




(a) 802.11 both end-nodes




(b) Uplink

Fig. 11. Per-packet jitter distribution of coexisting VoIP flows (CDF)

as a function of channel contention levels, 802.11 rate adaptation being on or off and direction of the traffic (uplink vs. downlink). Our initial focus was on homogeneous WLAN to WLAN environments (where the administrators can control the TCP algorithm used by all sending hosts) with 4, 8 and 12 stations sending to 4, 8 and 12 recipients.

We found that CDG achieves comparable (and sometimes better) per-station goodput to NewReno and CUBIC while inducing roughly twofold to fourfold smaller additional latency than NewReno and CUBIC. Furthermore, in WLAN-to-wired environments, CDG reduces the latency of uplink TCP traffic up to sevenfold. In addition to improving the median RTT, CDG provides a more stable performance by yielding significantly smaller variations in RTT in all scenarios.

CDG appears worthwhile trialling in controlled, homogeneous WLAN environments that mix TCP bulk data transfers and multimedia traffic. The next challenge is to explore CDG's ability to coexist with loss-based TCPs in heterogeneous WLANs where end-hosts use a mix of TCP algorithms.

## VII. Acknowledgments

This project has been made possible in part by grants from the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation. We are also grateful for access to Emulab's 802.11 testbed.

## References

[1] J. Postel, "Transmission Control Protocol," RFC 793 (Standard), Sep 1981, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[2] "ITU-T G.114." [Online]. Available: http://www.itu.int/rec/T-REC-G.114

[3] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer Quake 3," in *11th IEEE International Conference on Networks (ICON 2003)*, Sydney, Australia, Sep 2003, pp. 137–141.

[4] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The effects of loss and latency on user performance in Unreal Tournament 2003," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, Aug 2004, pp. 144–151.

[5] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," RFC 3782 (Proposed Standard), Apr 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3782.txt

[6] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul 2008.

[7] L. Stewart, G. Armitage, and A. Huebner, "Collateral damage: The impact of optimised TCP variants on real-time traffic latency in consumer broadband environments," in *Proceedings of IFIP/TC6 NETWORKING 2009*, May 2009.

[8] L.Stewart, D. Hayes, G. Armitage, M. Welzl, and A. Petlund, "Multimedia-unfriendly TCP congestion control and home gateway queue management,," in *ACM Multimedia Systems Conference (MMSys 2011)*, Feb 2011.

[9] J. Gettys, "The criminal mastermind: bufferbloat!" Dec 2010. [Online]. Available: http://gettys.wordpress.com/2010/12/03/introducing-the-criminal-mastermind-bufferbloat/

[10] T. Li, D. Leith, and D. Malone, "Buffer sizing for 802.11-based networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 156 –169, Feb 2011.

[11] N. Khademi, M. Welzl, and S. Gjessing, "Experimental evaluation of TCP performance in multi-rate 802.11 WLANs," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2012, pp. 1–9.

[12] N. Khademi and M. Othman, "Size-based and direction-based TCP fairness issues in IEEE 802.11 WLANs," *EURASIP J. Wirel. Commun. Netw.*, vol. 2010, pp. 49:1–49:13, Apr 2010.
[13] N. Khademi, M. Welzl, and R. Lo Cigno, "On the uplink performance of TCP in multi-rate 802.11 WLANs," in *IFIP NETWORKING 2011*, Valencia, Spain, May 2011, pp. 368–378.
[14] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 5, pp. 56–71, Oct 1989.
[15] D. A. Hayes and G. Armitage, "Revisiting TCP congestion control using delay gradients," in *IFIP NETWORKING 2011*, Valencia, Spain, May 2011, pp. 328–341.
[16] "NewTCP project tools," 2012. [Online]. Available: http://caia.swin.edu.au/urp/newtcp/tools.html
[17] "Emulab." [Online]. Available: http://www.emulab.net/
[18] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*. Boston, MA: USENIX Association, Dec 2002, pp. 255–270.
[19] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control ," RFC 2581 (Proposed Standard), Apr 1999, updated by RFC 3390. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt
[20] "Bufferbloat." [Online]. Available: http://www.bufferbloat.net/
[21] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 9–16, Apr 1992.
[22] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct 1995.
[23] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1246–1259, Dec 2006.
[24] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP," in *IEEE INFOCOM 2005*, Mar 2005, pp. 1838–1848.
[25] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *IEEE INFOCOM 2006*, Apr 2006, pp. 1–12.
[26] S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov, "Emulating AQM from end hosts," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, Aug 2007, pp. 349–360.
[27] A. Kuzmanovic and E. Knightly, "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 739–752, Aug 2006.
[28] D. Leith, R.Shorten, G.McCullagh, J.Heffner, L.Dunn, and F.Baker, "Delay-based AIMD congestion control," in *Proc. Protocols for Fast Long Distance Networks*, Feb 2007.
[29] L. Budzisz, R. Stanojevic, R. Shorten, and F. Baker, "A strategy for fair coexistence of loss and delay-based congestion control algorithms," *IEEE Commun. Lett.*, vol. 13, no. 7, pp. 555–557, Jul 2009.
[30] D. A. Hayes and G. Armitage, "Improved coexistence and loss tolerance for delay based TCP congestion control," in *35th Annual IEEE Conference on Local Computer Networks (LCN 2010)*, Denver, Colorado, USA, Oct 2010.
[31] D. Leith, R. Shorten, G. McCullagh, L. Dunn, and F. Baker, "Making available base-RTT for use in congestion control applications," *IEEE Communications Letters*, vol. 12, no. 6, pp. 429 –431, Jun 2008.
[32] R. T. Morris, J. C. Bicket, and J. C. Bicket, "Bit-rate selection in wireless networks," Master thesis, MIT, Tech. Rep., May 2005.
[33] "Minstrel description." [Online]. Available: http://madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt
[34] S. Choi, K. Park, and C.-k. Kim, "On the performance characteristics of WLANs: revisited," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 97–108, Jun 2005.
[35] J. Choi, K. Park, and C. kwon Kim, "Cross-layer analysis of rate adaptation, dcf and tcp in multi-rate wlans," in *IEEE INFOCOM 2007*, May 2007, pp. 1055 –1063.
[36] D. Hayes, "Timing enhancements to the FreeBSD kernel to support delay and rate based TCP mechanisms," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 100219A, Feb 2010. [Online]. Available: http://caia.swin.edu.au/reports/100219A/CAIA-TR-100219A.pdf
[37] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.
[38] S. Zander, G. Armitage, L. M. Thuy Nguyen, and B. Tyo, "Minimally intrusive round trip time measurements using synthetic packet-pairs," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 060707A, Jul 2006. [Online]. Available: http://caia.swin.edu.au/reports/060707A/CAIA-TR-060707A.pdf
[39] "tcptrace." [Online]. Available: http://www.tcptrace.org/
[40] "tcpreplay." [Online]. Available: http://tcpreplay.synfin.net/