Gateway Strategies for VoIP Traffic over Wireless Multihop Networks

Kyungtae Kim[1], Dragoş Niculescu[2] and Sangjin Hong[1]

[1] Mobile Systems Design Laboratory, Dept. of Electrical and Computer Engineering,
Stony Brook University-SUNY, Stony Brook, NY 11794 - USA
[e-mail: {kyungtae, snjhong}@ece.sunysb.edu]
[2] ETTI, University POLITEHNICA of Bucharest, Bucharest, Romania
[e-mail: dragos.niculescu@elcom.pub.ro]
*Corresponding author: Sangjin Hong

Received November 6, 2010; accepted December 4, 2010;
published January 31, 2011

Abstract

When supporting both voice and TCP in a wireless multihop network, there are two conflicting goals: to protect the VoIP traffic, and to completely utilize the remaining capacity for TCP. We investigate the interaction between these two popular categories of traffic and find that conventional solution approaches, such as enhanced TCP variants, priority queues, bandwidth limitation, and traffic shaping do not always achieve the goals. TCP and VoIP traffic do not easily coexist because of TCP aggressiveness and data burstiness, and the (self-) interference nature of multihop traffic. We found that enhanced TCP variants fail to coexist with VoIP in the wireless multihop scenarios. Surprisingly, even priority schemes, including those built into the MAC such as RTS/CTS or 802.11e generally cannot protect voice, as they do not account for the interference outside communication range.

We present *VAGP (Voice Adaptive Gateway Pacer)* - an adaptive bandwidth control algorithm at the access gateway that dynamically paces wired-to-wireless TCP data flows based on VoIP traffic status. *VAGP* continuously monitors the quality of VoIP flows at the gateway and controls the bandwidth used by TCP flows before entering the wireless multihop. To also maintain utilization and TCP performance, *VAGP* employs TCP specific mechanisms that suppress certain retransmissions across the wireless multihop. Compared to previous proposals for improving TCP over wireless multihop, we show that *VAGP* retains the end-to-end semantics of TCP, does not require modifications of endpoints, and works in a variety of conditions: different TCP variants, multiple flows, and internet delays, different patterns of interference, different multihop topologies, and different traffic patterns.

Keywords: Wireless multihop networks, VoIP, TCP, gateway solution, VAGP

DOI: 10.3837/tiis.2011.01.002

1. Problem Statement

Most traffic that flows over the Internet makes use of the Transmission Control Protocol (TCP) and wireless multihop networks are one way to provide access extension. TCP is one of the protocols designed for wired networks and exhibits severe degradation in multihop networks. It was designed to provide reliable end-to-end delivery of data over unreliable networks and has been carefully optimized in the context of wired networks. For example, large TCP default window sizes that are appropriate for a wired network are too large for wireless links in multihop networks.

Another type of traffic that becomes more prevalent in homes and institutions is VoIP. This capability becomes available in most new cell-phones as well, due to convenience and cost savings. VoIP, however, is different from most other traffic in that it has quite stringent delivery requirements. While mechanisms to provide for this QoS exist in the wired networks, in the popular 802.11-based networks they were only an afterthought.

In this paper we show that coexistence between these two popular traffics is a difficult one in multihop networks, and investigate the different methods that can be used to facilitate it. Even if QoS enhancements such as 802.11e were added, it doesn't really address the central problem of multihop networks, which is interference, the main factor affecting the coexistence. The interaction between TCP and VoIP over a multihop network is very complex. We summarize the most important points:

- *TCP is an end to end protocol*. There are no explicit signaling mechanisms in the network to tell the TCP peers how fast to send, how much to send, or when to slow down a transmission. A peer is responsible for controlling these parameters from implicit knowledge it obtains from the network or from explicit knowledge it receives from the other peer. TCP needs to be aggressive in discovering link bandwidth because this way it can achieve high utilization. This is achieved using large windows, which aggravates channel contention on wireless links.
- *TCP produces bursty traffic, while VoIP is uniform*. In the so called 'slow' start phase, TCP doubles its window for each ACK received - in reality an exponential increases in bandwidth consumption. This creates trains of packets that hog the medium for prolonged times. VoIP on the other hand needs regularity in the network delay and a low loss rate. When the network is congested by interference or too much TCP data, VoIP traffic suffers from increased network losses and delays. However, TCP just goes into the recovery stage, reducing its sending rate until the network is recovered from congestion, and then sends all postponed packets. This cycle of burstiness leads to both low utilization for TCP and unacceptable quality for voice.
- *TCP assumes that losses come from congestion*. This observation has been the basis of many studies and proposed modifications focusing on preventing TCP congestion control mechanism to react to link layer errors. Many performance studies of the TCP protocol over 802.11-based multihop show standard TCP behavior may lead to poor performance because of packet drops due to hidden terminal induced problems such as channel interference and TCP data/ACK contention.
- *VoIP packets are small, while TCP packets are large*. For a given bit error rate, TCP packets will have less success, so many of them would be retransmitted across multihop links, thus generating even more load that in turn generates more interference.

Accurate and timely estimation of the available bandwidth is very important. Although there are many tools to estimate this parameter over multiple hops, most prior work has largely focused on improving TCP performance over multihop networks, and was not concerned with the coexistence of TCP with real-time applications such as VoIP. VoIP is mostly constant bit rate, has very tight delay and loss requirements, and should always be served prior to TCP traffic. Classical solutions such as priority queues, bandwidth limitation, and traffic shaping do not provide satisfactory solutions for the coexistence problem. Even if voice traffic has priority locally within a node, bursty TCP traffic affects voice packets on other nodes within the interference range.

This paper investigates the behavior of TCP and VoIP flows in a shared network and proposes a novel bandwidth control technique to enable this coexistence in interference-ridden conditions such as multihop networks and WLANs. We examine ways in which TCP and VoIP can coexist while satisfying two contradicting goals: maintenance of VoIP quality, but without sacrificing TCP performance and network utilization. We found that merely limiting bandwidth of TCP, while necessary, is not sufficient, as the bursty behavior of TCP results in poor utilization. Another finding is that it is beneficial to make TCP look more CBR like for two reasons: it is more predictable and therefore VoIP friendly, but also has hidden benefits for TCP itself.

The problem is not simply a matter of bandwidth estimation, even if VoIP takes a predictable share of the resources, because it matters not only how much TCP to allow in the network, but also what shape as well. We propose *Voice Adaptive Gateway Pacer (VAGP)* - a method that uses existing VoIP traffic that shapes incoming TCP to protect both VoIP and utilization. This estimation is a continuous process that has to adapt to a changing environment: wireless channel conditions, VoIP load, TCP flow arrivals, internet delays - all may change on time scales of seconds. *VAGP* limits TCP share within milliseconds of noticing reduced capacity, but still allows aggressive discovery when new bandwidth becomes suddenly available, all while protecting voice traffic.

2. Existing Work

A large amount of research has focused on the optimization of the TCP performance over wireless networks - single and multihop.

Studies have been conducted to understand the capacity of multihop network and to improve VoIP capacity over the wireless mesh networks, as presented in [1], [2]. Several performance optimization schemes have been proposed to improve the VoIP quality over a WLAN: [3] proposed the use of a dedicated queue to provide higher priority to VoIP traffic over data traffic, while in [4], packet aggregation is used to increase capacity.

Many bandwidth estimation techniques have been proposed for wired networks. In [5], the authors survey different methods to evaluate the capacity, the available bandwidth and the bulk-transfer capacity.

Ultimately, the question of knowing how much TCP to allow in the network is one of bandwidth estimation, as VoIP takes a predictable share of the resources. In 802.11-based ad hoc networks, few works deal with solutions for bandwidth estimation. in [6], each mobile estimates the available bandwidth by computing the channel utilization ratio and using a smoothing constant. The channel utilization ratio is deduced from a permanent monitoring of the channel states. The main idea presented in [7] is that a probe packet delay higher than the maximum theoretical delay can characterize the channel utilization. The authors propose a

method to compute the medium utilization from the delays and then derive the available bandwidth from the channel utilization.

In wired environments, TCP is considered to be too slow in capturing the available bandwidth of high-performance networks, especially over long-distance paths. The underlying reasons are:

1. Limited buffers at the TCP sender or receiver impose a conservative upper bound on the effective window of the transfer, and thus on the maximum achievable throughput.
2. Packet losses cause a large and multiplicative window reduction, and a subsequent slow (linear) window increase rate, causing an overall low average throughput.

Some TCP-related issues that often impede performance are: multiple packet losses at the end of slow-start (commonly resulting in timeouts), the inability to distinguish between congestion and medium packet losses, the use of small segments, the coarse granularity of the retransmission timeout, or the initial value of the *ssthresh* parameter. Networking research has focused on these problems, pursuing mostly modified congestion control algorithms. From the wireless perspective, the problem that arises in the usage of TCP over wireless networks comes from the fact that wireless links have different characteristics with respect to wired ones, namely lower reliability and time-variant behavior, node mobility, hand-offs, limited available bandwidth and large RTTs. Since the only reaction provided by TCP in the event of unsuccessful packet delivery is the congestion control mechanism, TCP implementations perform poorly in wireless environments. The majority of the solutions proposed by the research community fall in three main categories:

Connection splitting solutions: The key problem for TCP over hybrid wireless/wired networks lies in the different characteristics of wireless networks and wired Internet. While most of packet losses experienced in wireless networks are due to hidden terminals and channel contention at the intermediate nodes, drops in the Internet almost always are due to buffer overflows at the routers. A solution to this network convergence problem [8] lies in splitting the TCP connection at the node interfacing the wired and wireless part of the network, denoted as the Internet gateway. Connection splitting can hide the wireless link entirely by terminating the TCP connection prior to the wireless link at the base station or access point. With this approach, the communication in wireless network can be optimized independently of the TCP applications. However it requires extra overhead to maintain two connections for one TCP communication. It also violates end-to-end semantics and complicates the handover process.

Link layer solutions: These try to make the wireless link layer look similar to a wired layer from the perspective of TCP. The most relevant and interesting proposal is the snoop protocol [9]. A snoop agent is introduced at the base station to perform local retransmissions using information sniffed from the TCP traffic passing through the base station. Another link layer solution proposes QoS scheduling with priority queues in the access point (AP) [3] to improve VoIP quality by placing TCP data in a lower QoS level.

Gateway solutions: One way to address TCP performance problems within wireless networks is to evenly space, or pace data sent into the multihop over an entire round-trip time, so that data is not sent in a burst. Pacing [10][11] can be implemented using a data and/or ACK pacing mechanism. TCP-GAP [10] suggested congestion control scheme to reduce burst of TCP packets based on estimating 4-hop propagation delay and variance of recent RTTs at the Internet gateway for wired-wireless hybrid networks. TCP-GAP scheme is relatively responsive, provides fairness among multiple TCP flows and better goodput than TCP-NewReno. However, it still depends on network topology, and fails to estimate TCP

bandwidth correctly in the presence of real-time traffic, such as VoIP. Congestion control is operated by the general TCP scheme, which is too aggressive for wireless multihops. To the best of our knowledge, the interaction between TCP and VoIP has not been studied in detail before, particularily in interferrence ridden scenarios, such as the wireless multihop.

3. TCP and VoIP: Difficult Coexistence

It is well understood from queueing theory that bursty traffic produces higher queuing delays, more packet losses, and lower goodput. It has been observed that TCP's congestion control mechanisms and self-clocking create extremely bursty traffic in networks with large bandwidth-delay products and causes long queues and the likelihood of massive losses. In addition, wireless multihop network traffic tends to have a self-similar behavior [12] making it difficult to provide stable rates necessary for VoIP.

Fig. 1 shows the wired/wireless hybrid networks we consider in this paper. A multihop extension carries traffic from wired Internet, or PSTN (through IP-PBX). The multihop leg is where VoIP needs to be protected from TCP.

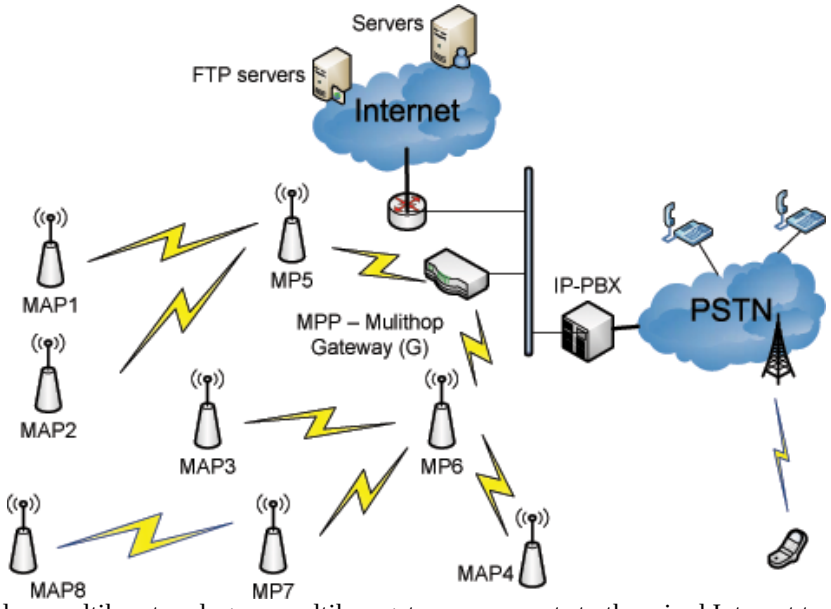

Fig. 1. Wireless multihop topology: a multihop gateway connects to the wired Internet to deliver TCP traffic, or to PSTN via IP-PBX for VoIP calls. A Multihop Point (MP) just forwards traffic, whereas a Multihop Access Point (MAP) also allows stations (STA) to associate with it. In this paper we consider downlink TCP traffic originating on the Internet, destined to a client associated with a MAP.

3.1 Bursty Traffic

To understand the difficulties in supporting VoIP, we start with a short primer on VoIP quality requirements. VoIP traffic is CBR, and for certain vocoders (G711, G729), its quality can be estimated using packet loss and mouth-to-ear (one way) delay. **Fig. 2** shows the values of *MOS-score* with respect to network delay and total loss for 60ms playout-buffer and 25ms vocoder delay. In order to obtain *MOS-score* of 3.6 (comparable to GSM quality), the network has to deliver all packets in less than 160ms, or deliver 98% in less than 104ms. For G.729a used in the rest of the paper, 3.9 is the maximum quality achievable, but we consider 3.6 to be “acceptable quality”.

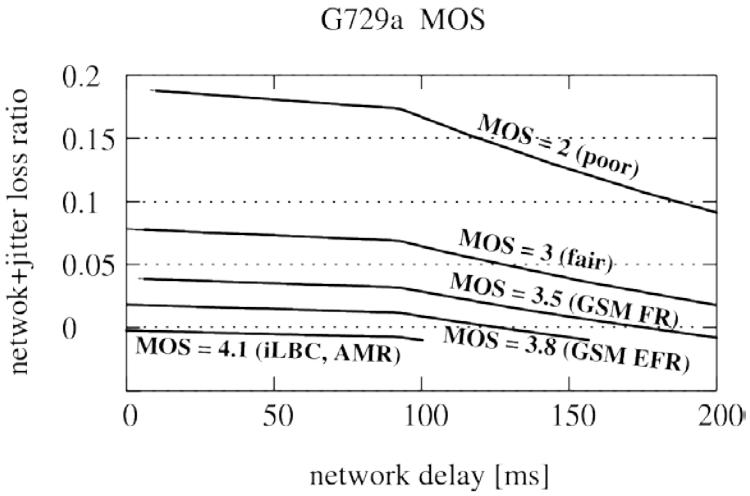

Fig. 2. For certain vocoders, such as G729a, VoIP quality (*MOS-score*) can be computed as a function of loss and one way delay. Loss include packets lost in the network, and packets which miss their deadline because of jitter. For more details, see [13].

First, we show the burstiness is the main cause of reduced VoIP quality. To this end, we set up a string with four wireless nodes, equivalent with the sequence MPP - MP6 - MP7 - MAP8 - Client in **Fig. 1**. All these hops operate on the same channel. In this setup we experiment with various packet patterns as shown in **Fig. 3**. In these scenarios we have the same mean offered rate for data (550Kbps), but with different burst lengths. The rest of the capacity is filled by VoIP packets.

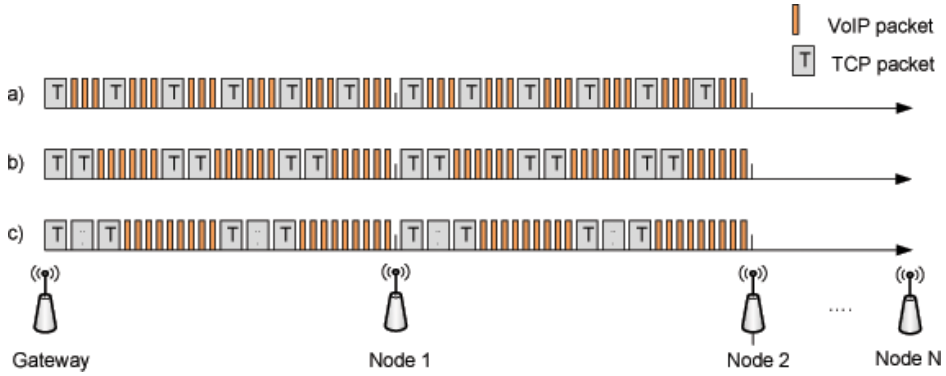

Fig. 3. VoIP statistics and data goodput as the burstiness increases; 5 VoIP calls and 550Kbps of data offered; 4-hops string topology, 12Mbps, 802.11a.

Table 1. VoIP statistics and data goodput as the burstiness increases; 5 VoIP calls and 550Kbps of data offered; 4-hops string topology, 12Mbps, 802.11a.

Burst length	VoIP calls	VoIP loss (%)	VoIP delay (ms)	Data Goodput (Kbps)	Data delay (ms)
1	5	0.92	13	509	15
2	5	1.26	16	501	19
3	4	1.44	17	495	23
4	4	1.64	19	488	26
5	3	2.06	21	476	31

The results corresponding to different burst lengths are shown in **Table 1**. Virtually all quality indicators for both VoIP (loss, one-way delay) and data (goodput, one-way delay) suffer because of the increased burst length. In fact, we can support 5 voice calls with 1 data packet bursts, but only 3 with 5 data packet bursts. In Internet scenarios, when long delays can be present on the Internet portion, even one TCP flow is expected to require windows much larger than 5 packets, and therefore produce even more degradation for itself and for VoIP.

Table 2. Max number of VoIP calls and TCP throughput (Mbps) as the hop count changes, string topology, one channel, 12Mbps, 802.11a.

Hops	No. of VoIP	TCP (Mbps)	Reason for degradation
1	45	9.5	Contention
2	23	4.9	Contention
3	16	3.2	Contention, Interference
4	12	2.5	Contention, Interference
5	9	2.0	Contention, Interference

In the same topology of four hops we try to establish what kind of performance we can expect from each type of traffic, and in combination. Running each of the four hops at 12Mbps, we can either support 11 VoIP calls, or 1.35Mbps of TCP, as illustrated in **Table 2**.

However, if we mix 5 VoIP calls and 3 TCP flows, we found that voice quality is below the minimum acceptable (*MOS* < 2) while TCP flows get a cumulative goodput (i.e. total bytes delivered by TCP receiver to the application layer) of 615Kbps using a throughput (i.e. total bytes sent by the TCP sender) of 903Kbps. This shows that simply sharing the network fails to protect the VoIP traffic, and also yields lower utilization. TCP uses a sliding window-based protocol which determines the number of packets that can be sent, and uses the receipt of acknowledgments to trigger the sending of packets. The window used by a TCP sender is chosen based on its view of the congestion in the network and based on the receiver's acceptable number of bytes. If the window size is too large, then the sender is allowed to inject more traffic than the network can handle. Given a wireless multihop network, there exists a TCP window size W^* at which TCP's bandwidth consumption is appropriate. This number depends on many conditions, including presence of real time traffic, but the main point is that default TCP algorithms are not able to discover this W^*. The current TCP protocols do not operate around W^* but instead typically grow their average window much larger. This results in VoIP degradation or in low TCP performance even if VoIP traffic is not present [14].

3.2 Behavior of Mixed Traffic

Continuing with the four hop scenario, we then mix these two types of traffic: five voice calls are started at the beginning of the experiment, and after 10 seconds, three TCP flows are started in parallel with the voice. **Fig. 4** shows how VoIP quality is degraded as TCP window size increases. VoIP quality is shown in the low-right of **Fig 4**, which starts at *MOS-score* of around 3.9 and decreases to *MOS-score* of around 2.2 as soon as the TCP flows with default window size 32 are introduced. TCP flows initially go through the ``slow-start'' phase increasing the number of packets in transit exponentially, which means every ACK packet triggers twice as much data as it acknowledges. After reaching the congestion threshold, TCP switches to linear rate increase to maintain high goodput without causing congestion in the congestion avoidance phase. These two alternating phases produce the well known saw-tooth pattern shown in the upper left of **Fig. 4**. In the lower-left of **Fig. 4**, we see an additional disadvantage that is wireless specific: the self-interfering nature of the wireless multihop, coupled with well known 802.11 unfairness issues [15] causes severe unfairness between the

TCP flows. In the top right of **Fig. 4**, we trace the delays experienced by VoIP packets of one of the flows during the same experiment: due to TCP packet bursts and large packet size, network delay jumps from 10ms to about 250ms mostly due to extra wait in the queues. This is the main driver of reduced voice quality, as large queues and 802.11 retries largely cover most network losses in this example.

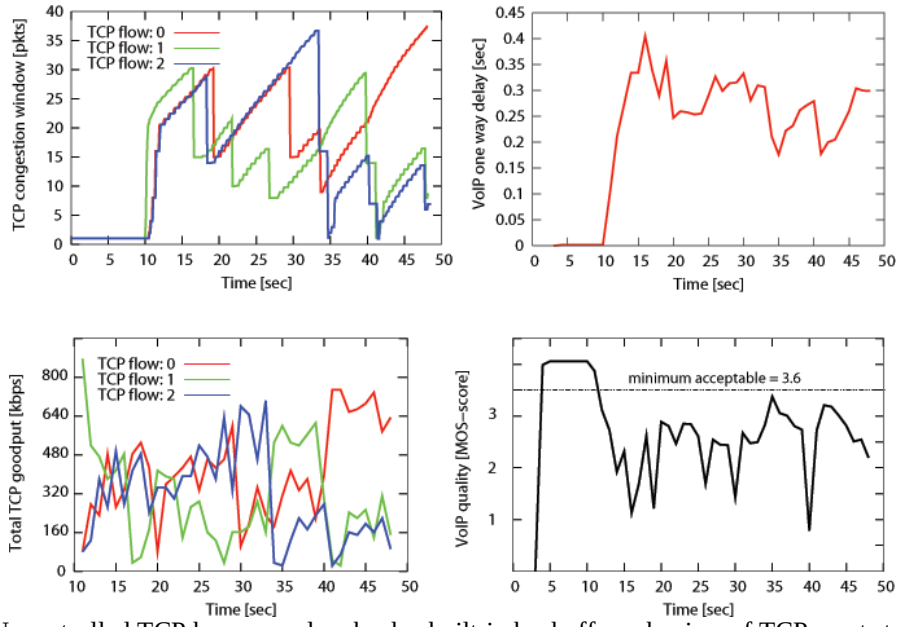

Fig. 4. Uncontrolled TCP has many drawbacks: built-in backoff mechanism of TCP reacts too late to protect VoIP; Increased one-way network delay for VoIP; unfairness between TCP flows; low total utilization.

4. Candidate Solutions

It is clear from the previous section that VoIP and TCP cannot simply share a multihop network without experiencing severe reduction in TCP capacity and voice quality degradation. We first consider the various enhancements to TCP proposed by the research community in the recent years, namely: Reno, Vegas, Westwood, CUBIC, and Compound TCP (C-TCP). Then, we look at external control methods, that would leave TCP unchanged, but would police or instrument traffic at the multihop gateway.

4.0.1) TCP Reno: This is the traditional algorithm in most operating systems currently deployed, and we consider it as a base case. TCP Reno defines four key mechanisms: slow start, congestion avoidance, fast retransmission and fast recovery. In the ‘slow’-start phase, the congestion window grows exponentially increasing *cwnd* by 1 with every acknowledgement, until a timeout occurs or a duplicate ACK is received. The latter implies that a packet has been lost and this signals that the sender transmits packets faster than the network can handle. Congestion control algorithm is used to slow down the transmission rate.

In the congestion avoidance phase, the sender grows its window linearly assuming that the sending rate is close to the bottleneck capacity, until it detects a packet loss or a timeout. Reno also includes a fast retransmit and recovery mechanisms which make it possible to quickly recover lost packets.

4.0.2) TCP Vegas: TCP Vegas was introduced with the idea that it is more efficient to prevent congestion than to fix it. One of the core features of Vegas is that all changes are confined to the sender side, including loss detection, estimation of the available bandwidth, and the new slow start behavior. These modified mechanisms use observed delay to detect an incipient stage of congestion and try to adjust the congestion window size before packets are lost. Thus, Vegas attempts to determine the correct window size without relying on packet losses.

4.0.3) TCP Westwood: TCP Westwood enhances the window control and backoff process. Westwood relies on end-to-end rate estimation. The key innovative idea is to continuously measure at the TCP sender the packet rate of the connection by monitoring the rate of returning ACKs while trying to find the bandwidth estimate which is defined as the share of bottleneck bandwidth available to the connection. The estimate is then used to compute congestion window *cwnd* and slow start threshold *ssthresh* after a congestion episode, that is, after three duplicate acknowledgments or a timeout. Westwood is a sender side modification of the congestion window algorithm aiming to improve the performance of Reno in wired as well as wireless networks. However, the available bandwidth estimation algorithm is complex and may not be able to follow the rapid changes in a hybrid wireless network.

It fully complies with end-to-end TCP design principle. Whenever a Westwood sender detects a packet loss that indicates a timeout has occurred or that three duplicate ACKs have been received, the sender estimates the bandwidth to properly set the congestion window and the slow start threshold. Westwood avoids overly conservative reductions of *cwnd* and *ssthresh*. The available bandwidth estimation algorithm is complex and cannot follow the rapid changes in a hybrid mobile network.

4.0.4) TCP CUBIC: TCP CUBIC [16] was proposed to address the under-utilization problem due to the slow growth of TCP congestion window in high-speed networks. The window growth function is updated with the elapsed time since the last loss event, so that its growth is independent on network delay. This means the sender is allowed to put more packets without long waiting for the acknowledgements in a network with large bandwidth delay products, probing the bottleneck bandwidth quickly. CUBIC has been used by default in Linux kernels since version 2.6.19.

Its window growth function is based on the elapsed time since the last loss event independently of RTT. The congestion control algorithm of CUBIC increasing sender's window aggressively and fast responses in fast long distance networks. The window growth function of CUBIC is updated in a real-time so that its growth is independent of RTT by a cubic function in terms of the elapsed time. This is the preferred algorithm in the newest Linux kernels (after 2.6.13).

4.0.5) Compound TCP (C-TCP): With the idea that pure loss-based or delay-based congestion control approaches that improve TCP throughput in high-speed networks may not work well, this algorithm is designed to combine two approaches. C-TCP [17] can rapidly increase sending rate when network path is underutilized, but gracefully retreat in a busy network when bottleneck queue grows. C-TCP is the algorithm included with Windows Vista and Windows Server 2008. However, due to the loss-based component, CUBIC and C-TCP are not designed for high-loss wireless paths.

What is true for all TCP variants is that data packets arrive at the receiving host at the rate that the bottleneck link will support. A TCP sender's self-clocking depends on the time spacing of ACKs being preserved end to end. Jitter introduced in network queues misleads the sender into pushing more data than the network can accept. Cumulative acknowledgement or ACK compression may cancel the spacing of the ACKs and result in bursty traffic with a high risk of

high peak rate beyond network capacity. A single ACK can acknowledge several thousands of packets, opening up the window in a large burst.

We compared these five TCP variants with respect to their capacity to coexist with VoIP and utilization of the multihop. **Fig. 5** shows that all TCP variants fail to protect VoIP in a simple shared environment. Most of the time, they simply increase TCP goodput with large window size. Vegas exhibits both a better VoIP protection and utilization of the multihop links, due to its balanced congestion control in low number of VoIP flows. Surprisingly, Westwood, which is designed specifically for lossy links performs worst on both measures, as it wastes half of the capacity on retransmissions $\frac{goodput}{total\_sent} \approx 0.5$ (not shown in the figure).

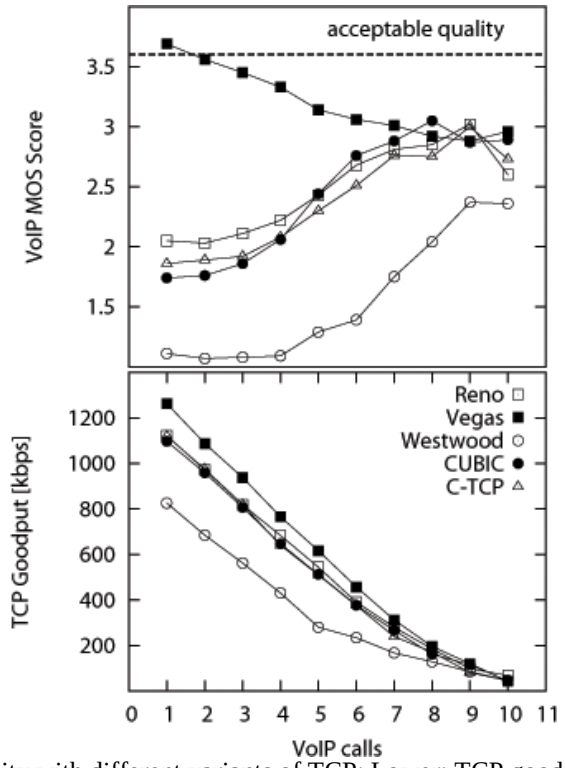

Fig. 5. Upper: VoIP quality with different variants of TCP; Lower: TCP goodput. TCP variants are too aggressive and use all available bandwidth, reducing voice quality. However, with more voice calls TCP experiences more packet loss, which leads to retransmissions and frequent slow start phases.

4.1 Policing TCP Traffic

In fact, an even more likely situation is that none of the TCP endpoints can be controlled because upgrading TCP is unfeasible or undesirable for other reasons. Even enhanced TCP endpoints cannot possibly protect wireless multihop networks in the path. We therefore explore other methods to enable the coexistence at the gateway into the wireless multihop. Policing of TCP traffic can be performed using classical methods such as priority queues and traffic shaping, or by instrumenting TCP packets to manipulate receiver's advertised window (*awnd*). Any of these methods has the goal of reducing the amount or the shape of the TCP data pushed into the multihop.

Priority Queues: One solution to harmonize VoIP and TCP traffic is the use of priority queues. We simulated priority queues in *ns*-2 allocating the highest priority for VoIP traffic in

all nodes. We found that only 20% of the voice capacity can be used, and only for one or two hops. For cases of three or more hops, priority queues are not able to support any amount of VoIP traffic. The reason is that priority queues, or even 802.11e (802.11e uses multiple queues for downlink traffic, and preferential contention parameters for uplink traffic in order to offer priority to QoS traffic) cannot protect from interference generated two or three hops away. As far as IP level is concerned, in each intermediate node, downlink/uplink VoIP is given priority against TCP. For the experimentation, we enabled 802.11e, and obtained the similar results. The priority to VoIP may improve the VoIP quality in a node with the presence of TCP traffic. However, the main reason of the VoIP quality degradation in the wireless mesh network is the interference from the nodes a few hops away, hidden-terminal protocol and 802.11e enhancement designed to QoS differentiation within a node fails to solve the problem.

On the contrary, this approach even increases packet burstiness while building up TCP packets in the queues. These localized approaches cannot provide solution to a global problem of hidden terminals interfering across several hops.

Fig. 6 shows the number of voice calls that can be supported in the presence of TCP traffic, and compares it with maximum possible values listed in **Table 2**. For 1 hop, the maximum number of VoIP calls is around 45, however with TCP traffic, it is reduced to 4 calls. In case of more than 3 hops, it becomes even worse - no voice calls are supported. The throughput of lower priority TCP is 6.3Mbps in the presence of higher priority VoIP calls (1 hop, 3 TCP flows, 7 VoIP calls supported).

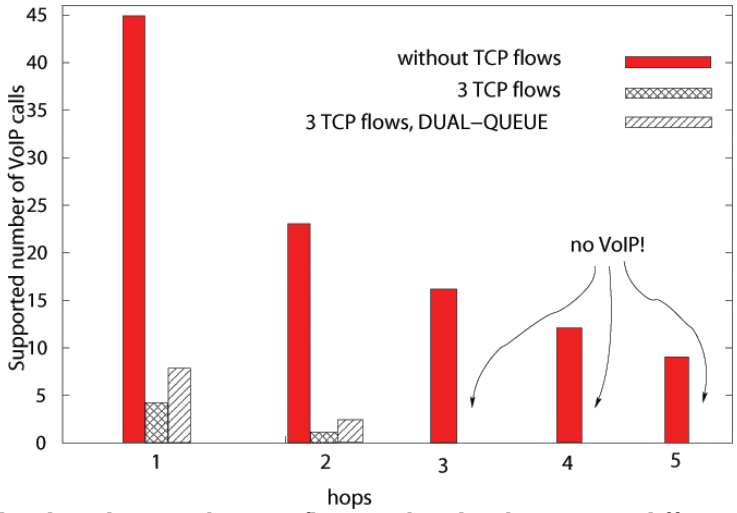

Fig. 6. VoIP quality degradation with 3 TCP flows, with and without service differentiation. Maximum number of calls are values from **Table 2** repeated for reference.

Window Resizing: TCP bandwidth discovery operates from the sender, and cannot be easily manipulated. The advertised window of the receiver however, can be instrumented in the network to reflect the actual bandwidth available in the wireless network. In concordance with previous studies, we found that limiting TCP sending behavior has beneficial effects even in the case when only TCP traffic is present in the network. In order to control TCP sending rate without modification of TCP endpoints and maintain end-to-end semantics, we modify the advertisement window in each ACK packet at the gateway (G in **Fig. 1**). This method limits the total number of TCP data packets in transit between the end points. If the gateway changes TCP advertisement window based on the network status, TCP throughput can be limited close

to its entry point. By keeping the window size small to protect VoIP, retransmission and fairness problems among TCP flows are also relieved.

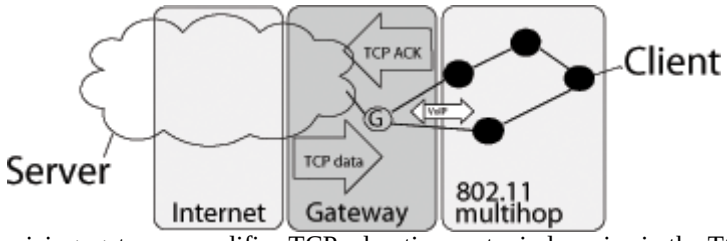

Fig. 7. Window resizing: gateway modifies TCP advertisement window size in the TCP ACK packets according to the traffic in network. TCP data packets pass through unchanged, but the server across the internet pushes less data as for a slower client.

As shown in **Fig. 4**, default TCP behavior does not discover the proper W^* and generally pushes too much data. If the knowledge of optimal W^* is available, the edge routers in the wireless domain could modify the advertisement window size of TCP ACK packets, an operation transparent to TCP applications. This window-controlled TCP traffic does not produce more data packets than the path capacity which the network is able to support, allowing VoIP packets to be transferred within the delay and jitter budget. **Fig. 7** shows the architecture of window resizing mechanism. Gateway node G peeks into all TCP ACK packets to modify TCP advertisement window field such that VoIP traffic can at least have a proper share of the bandwidth. TCP data passes through G untouched. How to harmonize the window size with VoIP? In **Fig. 8** the maximum VoIP capacity is shown with solid bars. The patterned bars indicate the VoIP capacity with the window limitation feature. While the actual result is dependent on the characteristics of the network simulated, the trend shows that smaller window sizes bring more benefit for VoIP support and even for TCP traffic itself in case of no VoIP traffic. With 3 TCP flows, optimum window size (number of unacknowledged packets) is one or two. As the window size increases, the number of TCP packets in transit gets larger, and the timely delivery of VoIP packets is affected. TCP clients with large default window size (64KB in Linux 2.4, 47KB in *ns*-2) consumes almost the whole bandwidth allowing only 5 VoIP calls to be supported even in a single hop network. It is clear that smaller windows are more beneficial than larger windows, and this is a consequence of the fact that TCP's share of the wireless medium needs to be reduced [14]. Unfortunately, this solution does not seem scalable with the number of TCP flows.

TCP Data/ACK Pacing: One problem that is not solved by window resizing is that of packet bursts. TCP pacing promises to reduce burstiness of TCP traffic and alleviate the impact of packet loss, network delay, and delay jitter of VoIP traffic. TCP pacing evens out the transmission of a window of packets based on a shaper parameter *R*. After a packet of size *pkt_size* goes out in the air, the next packet is scheduled no earlier than $\frac{pkt\_size}{R}$. The gateway chooses a rate *R* based on the network status to determine how much to send, and when to send. One way to understand the impact of pacing is to consider burstiness from network delay, jitter and packet loss perspective. With bursty traffic, packets arrive all at once at the gateway. As a result, queueing delay and delay jitter of VoIP packets grow linearly with TCP load due to large packet size, even when the load is below capacity. From the viewpoint of TCP, 802.11 links which support VoIP traffic seem to have large capacity left for TCP data. This ignores the interference side effects that are felt several hops away from the link in question.

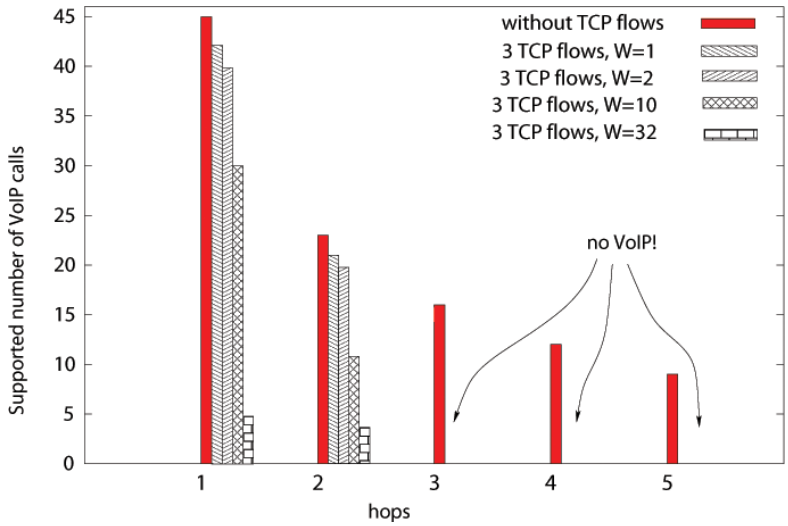

Fig. 8. Maximum number of VoIP call supported when TCP advertisement window size is instrumented to reduce TCP share. The actual window needed is dependent on configuration: number of hops, number of TCP flows, RTT.

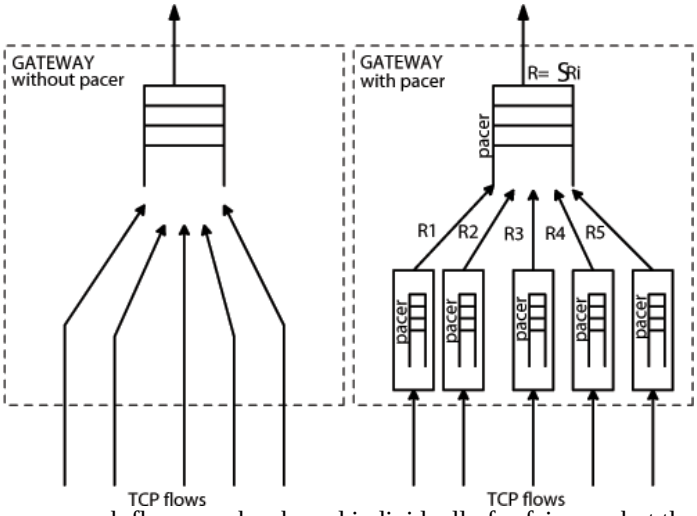

Fig. 9. TCP data pacer: each flow may be shaped individually for fairness, but the total TCP traffic should also be shaped to make it VoIP friendly.

TCP data shaper shown in **Fig. 9** evenly spaces data packets sent into the network for each TCP flow as well as for multiple TCP flows. TCP shaper first calculates the fair share R_i for each TCP flow from the total share for TCP. Data from each TCP shaper might still create burst packets, and to handle this, there is one more pacer with rate R for all TCP flows. This policy enforces fairness between TCP flows, but other policies or priorities can be used.

In **Fig. 10** the performance of the TCP data shaper is compared with the basic case. Although it supports a lower number of voice calls, the token bucket offers some protection to VoIP traffic. The disadvantage of this method is that it may allow some unfairness when several TCP flows are shaped together along the same path. In addition, buffer overflows at the gateway due to capacity fluctuation will cause drops, unlike the window resizing solution.

However, the main advantage is that it works with higher number of hops, and does not require instrumentation of TCP packets.

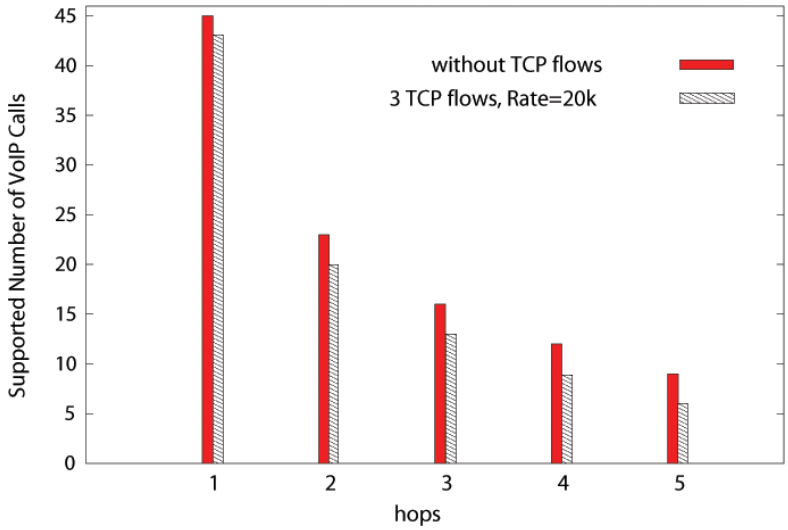

Fig. 10. Maximum number of voice calls supported when TCP data is policed with a shaper: it scales better to higher number of hops, and it provides reasonable utilization.

In our measurements, the pacer offered protection to VoIP at the cost of sacrificing available bandwidth for retransmissions. While providing benefits such as small buffer size at the pacer, ACK pacing may fail to prevent bursty data packets which results in low TCP performance and degradation of VoIP quality. The disadvantage of pacing is that buffer overflows at the gateway due to capacity fluctuation will cause packet drops and increase queueing delay. The increased queueing delay easily causes TCP retransmission timer to expire, which results in retransmitting the packets already transferred to the receiver, unlike the window resizing solution. However, the main advantage is that it works with higher number of hops, and does not require instrumentation of TCP packets.

4.2 Window Resizing versus Pacing

In this section, we examine in detail which of the two candidates is more appropriate to protect voice traffic and provide better utilization of the multihop. For voice emulation, we generated and analyzed flows of 50 packets per second, 20 bytes per packet in each direction that emulate G729a traffic. The reason for using this type of traffic is that most VoIP SIP phones (Zyxel, Utstarcom, Netvox) support it, and can be evaluated using the loss and the delay measured in the network (**Fig. 2**), without employing waveform analysis such as PESQ. In our experiments, G729a like traffic is considered supported if it achieves a quality better than MOS=3.6.

Testbed setup: We implemented TCP data pacing and window control in an 802.11a test-bed using Atheros based cards using the madwifi driver under Linux. The test-bed consists of 5 nodes spread over the floor of a building such that a string with appropriate combination of contention and interference is created. In particular, nodes that are two hops apart are out of communication range, but within carrier sense range. Nodes three hops apart are out of contention range but within interference range. First and last nodes are outside contention and interference range of each other. This particular setup captures a variety of situations that is likely to be encountered in indoor 802.11 networks - both WLAN and

multihop. An additional reason for aiming at a interference/contention ridden setup is that without interference, wireless networks would behave much like wired networks, in which more traditional methods of separation of traffic are available.

Communication range was verified using icmp traffic. Carrier sense range of two nodes was verified using broadcast traffic at maximum capacity from both nodes. If the amount they put on the air sums up to 100%, they are in carrier sense range. If they output a total of 200%, they are not deferring to each other. Interference relations are verified by running triplets of source, destination and interferer, with the interferer outside the carrier sense range of the source. An actual interferer would reduce the throughput between source and destination. For more details on how to measure carrier sense and interference conditions, see [18].

Network utilization with TCP and VoIP: In **Fig. 11**, we look at how TCP and VoIP can share the available bandwidth using window control and data/ACK pacing. On the horizontal axis, we increased the number of calls from 1 to 11, and attempted to maximize the TCP throughput while still maintaining *MOS-score* of 3.6 for the VoIP traffic. While all three control methods achieve some amount of sharing between the two types of traffic, the window control attains better utilization. The benefit of TCP policing is visible even without VoIP traffic when plain TCP wastes capacity on retransmissions achieving a lower goodput.

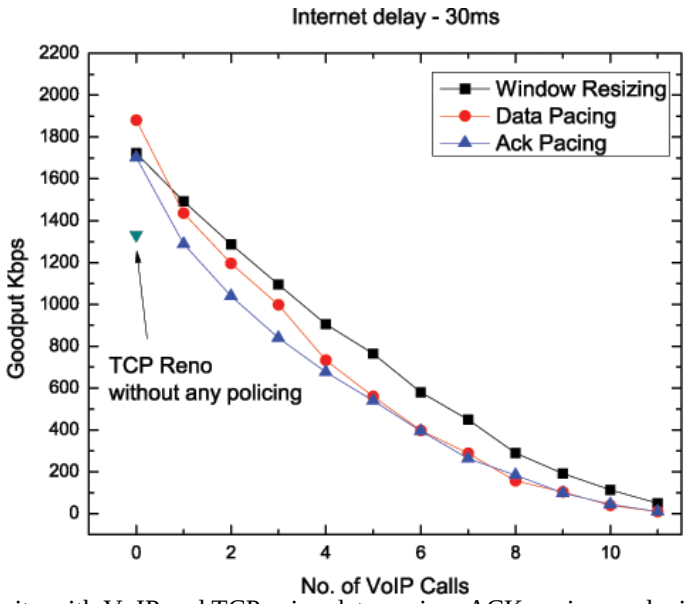

Fig. 11. Shared capacity with VoIP and TCP using data pacing, ACK pacing, and window control. All methods are bounded by the nominal capacity of the network. Window resizing reduces total number of retransmissions, which leads to a higher TCP goodput compared to data/ACK pacing.

Scalability with respect to number of TCP flows and amount of internet delay: From previous experiments we can conclude that while all methods can be used to control TCP rate, with the window control having a slight advantage by providing higher utilization. We then experimented with increasing number of TCP flows and found that window control cannot support more than 11 TCP flows when 3 voice calls are present as it requires a window size less than 1 packet. When using internet delay of 150ms (RTT), the required window sizes are larger, but only 15 TCP flows can be supported due to the same reasons.

We now look in more detail at the effect of larger number of TCP flows and of internet delay (**Fig. 12**). First, we inject between 1 and 10 TCP flows over a fixed voice load of 3 voice

calls. As before, the goal is to maintain an *MOS-score* of 3.6 for these three voice calls, while maximizing the throughput of TCP. The curve labeled 'window control' in **Fig. 12** shows the degradation caused by the excessively small TCP window required. The shaping method is more immune to the amount of TCP flows, but has a lower utilization when there are few TCP flows. Another interesting aspect is when TCP flows are terminated across the internet, incurring additional delay. We added 150ms one way delay to the TCP traffic in our testbed, while maintaining the VoIP traffic at the speed dictated by the load. We see a general reduction in utilization, but the difference between the two methods is smaller, as well as the dependence on the number of TCP flows.

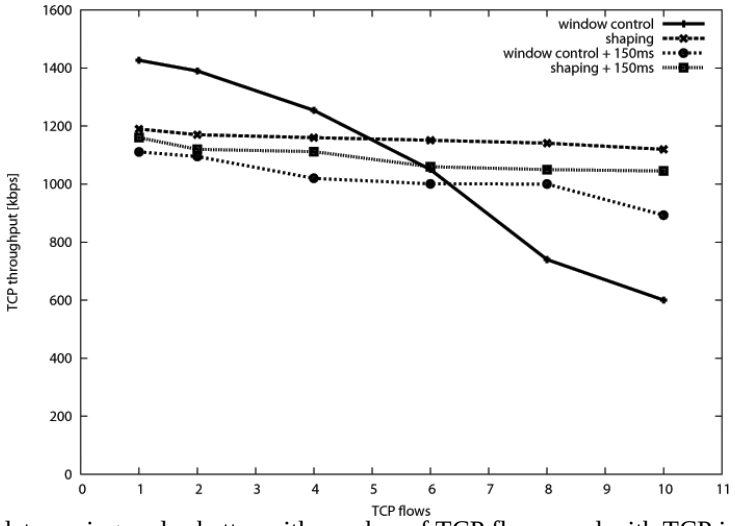

Fig. 12. TCP data pacing scales better with number of TCP flows, and with TCP internet delays.

If TCP traffic terminates across the Internet, connections with high bandwidth-delay product might still require a large window in order to achieve the desired TCP throughput. Consider the example when the optimal window size is $W = 2$ on a 4 hop topology: 6 calls are being supported, and a remaining bandwidth of 600Kbps can be used by TCP when $RTT = 2 \times 8 \times 1500 / 600000 = 40$ms across the multihop, according to $RTT = \frac{W}{Bandwidth}$. When an internet delay of 100ms is included and TCP faces an $RTT = 140$ms end to end, in order to achieve the 600Kbps available, a window $W = 7$ is needed, which is larger than the optimal window size. While achieving the job of limiting the data in the wireless string, W=7 also allows bursts of 7 packets of 1500 bytes, thus disturbing VoIP flows. The window control is not able to prevent packet burstiness with many TCP flows, but even for even one flow, when the delay is large. We conclude that shaping is more robust against varying conditions such as number of TCP connections and increased internet delay, although it comes at the cost of slightly reduced utilization, especially in the case when there are few TCP flows. TCP window control has the potential to achieve better utilization, but is more sensitive to external conditions, so it requires a highly dynamical behavior that depends on load, delay, and number of connections. In addition, it requires instrumentation of TCP traffic, which can be undesirable for high speed implementation, or when transporting encrypted traffic.

4.3 Testbed comparison of window resizing and pacing

We implemented TCP data pacing and window control in an 802.11a test-bed using Atheros based cards using the madwifi driver under Linux. The test-bed shown in **Fig. 13** consists of 5 nodes spread over the floor of a building such that a string with appropriate combination of contention and interference is created. In particular, nodes that are two hops apart are out of communication range, but within carrier sense range. Nodes three hops apart are out of contention range but within interference range. Nodes A and E are outside contention and interference range of each other. This particular setup captures a variety of situations that is likely to be encountered in indoor 802.11 networks - both WLAN and multihop.

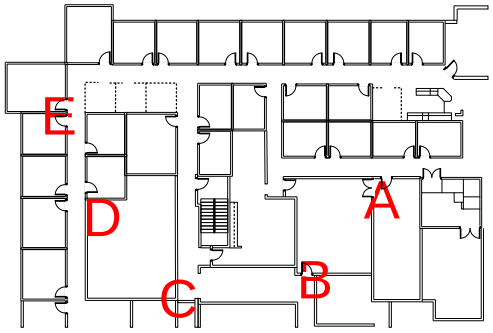

Fig. 13. Testbed with four hop setup using 802.11a Atheros cards at 12Mbps. Configuration experiences a variety of interference conditions: carrier sense and hidden terminals.

Communication range was verified using icmp traffic. Carrier sense range of two nodes was verified using broadcast traffic at maximum capacity from both nodes. If the amount they put on the air sums up to 100%, they are in carrier sense range. If they output a total of 200%, they are not deferring to each other. Interference relations are verified by running triplets of source, destination and interferer, with the interferer outside the carrier sense range of the source. An actual interferer would reduce the throughput between source and destination.

Testbed raw capacity We focused the experimentation on a subset of the situations explored in the previous simulations, namely a 4 hop situation using 802.11a, channel 64, and 12Mbps bit-rate. The capacity of the testbed is indicated in **Table 3** (please note that these numbers are a little different from the results shown in **Table 2** as those numbers are based on ns-2 simulations). For TCP we measured using `iperf` the TCP and UDP capacities using the default settings. The 2.6 kernel default settings and `iperf` defaults allowed a TCP window of 64KB which is typical for many application/OS combinations. For UDP, the packet size used was of 1470 bytes, therefore the capacity shown is the maximum supported by the architecture. For voice capacity, we used UDP packet generator `rude/crude` to generate and analyze flows of 50 packets per second, 20 bytes per packet in each direction that emulate G729a traffic. The reason for using this type of traffic is that most VoIP SIP phones (Zyxel, Utstarcom, Netvox) support it, and can be evaluated using only the loss and the delay measured in the network (**Fig. 2**), without employing waveform analysis such as PESQ. In our testbed, G729a like traffic is considered supported if it achieves a quality better than MOS=3.6.

For instrumentation of TCP and UDP traffic, we used the click modular router [20], installed in the kernel. It allows injection of traffic at any point through a `tap/tun` interface, and instrumentation of any backhaul traffic. Node E is considered as the gateway into the wireless domain and it is the one receiving the TCP traffic, that is transported to node A to be

relayed across the internet. For our experiments TCP data is traveling in the direction E → A , while voice traffic is bidirectional between A and E. Node A may emulate long internet like links by adding some delay before passing the data to the TCP endpoint on the tap/tun interface. Node E contains all the gateway logic to monitor the voice quality and instrument the TCP traffic. All the other nodes operate unmodified, as do the application endpoints for TCP and voice.

Table 3. Capacity of the testbed for TCP, UDP and VoIP traffic

Hops	TCP[Kbps]	UDP[Kbps]	Voice [#of calls]
1	8670	9910	54
2	4600	5110	24
3	2940	3410	15
4	1890	2060	11

Table 4. TCP and VoIP traffic with no special support

Hops	TCP[Kbps]	MOS	Voice [#of calls]
1	6300	3.69	18
2	4070	2.73	0
3	2290	1.60	0
4	1630	1.25	0

In the first set of experiments, we verified the capability of the testbed to support VoIP together with one TCP connection in the default setup. In **Table 4** it is shown that only in one hop case can VoIP and TCP coexist - and with a maximum of 18 voice flows out of the total possible of 54. For multiple hops, not even one call of VoIP can be supported, the MOS being well below the acceptable quality of 3.6. This is qualitatively consistent with the results obtained in section 4.1 (**Fig. 6** and **Fig. 8**).

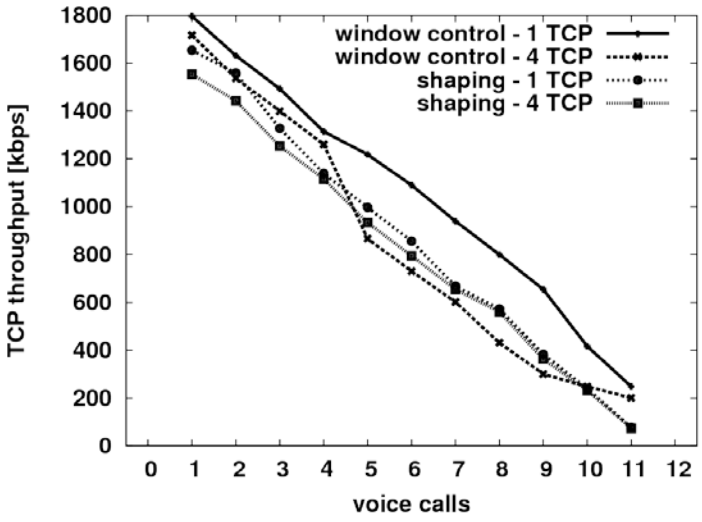

Fig. 14 Shared capacity with VoIP and TCP using shaping and window control. All methods are bounded by the nominal capacity of the network. Compare with **Fig. 11** obtained through simulation. In **Fig. 14**, we look at how TCP and VoIP can share the available bandwidth using window control or shaping. On the horizontal axis, we increased the number

of calls from 1 to 11, and attempted to maximize the TCP throughput while still maintaining an MOS score of 3.6 for the VoIP traffic. While both control methods achieve roughly linear sharing between the two types of traffic, the window control attains better utilization. When using 4 TCP flows in parallel, the window control method is degraded because of the small window required to support more than 5 calls. In these cases, the required window is below the MTU of 1500 bytes, which leads to TCP using smaller packets, which eventually leads to increasing overheads. They are caused by extra TCP, IP, MAC headers and especially by MAC protocol overhead - physical layer, contention, etc. The shaping method fares more consistently with 4 TCP calls – the shaping is performed with respect to all TCP traffic, not per each flow.

Both methods examined in simulation and testbed have a straightforward application only when the wireless capacity is fixed. In reality, the capacity is highly variable depending on a multitude of factors: number of hops and their configuration, the amount and type of interference, the actual capacity of each hop, the amount of voice to be served. To support VoIP under varying conditions, the basic policing tools examined here should be used in conjunction with methods to dynamically estimate available bandwidth in real-time.

5. Voice Adaptive Gateway Pacer

Having established that pacing is an appropriate method to control TCP in a shared network in Section 4.1.3, the question is what data rate should TCP get? We can divide bandwidth statically between the voice calls and the amount of TCP, but the static bandwidth sharing causes poor network utilization when usage is low for TCP and high for VoIP or even fails to protect voice in case of poor wireless link. In reality, the capacity is highly variable depending on a multitude of factors: network topology and its configuration, the amount and type of traffic, the actual capacity of each node, the amount of voice and other traffic to be served for each node. The method we propose - *Voice Adaptive Gateway Pacer* uses the existing voice traffic as probe traffic to estimate the rate that can be given to TCP:

$$R_{TCP}(t) = \sum_{i=1}^{N} R_{TCPi}(t) = \left(R_{TCP}(t-1) + R_{adj}(t) \right) \quad (1)$$

The bandwidth for TCP flows can be estimated by measuring the average of number of bytes transmitted over the interface, $R_{TCP}(t\text{-}1)$, adjusted with R_{adj}, the variation of VoIP quality. The adaptive transmission rate R_{TCP} for TCP bandwidth computed by the multihop gateway is allocated to each TCP flow following a bandwidth sharing policy. The period of adjustment and the time scale used to measure the rate depend on the estimated round trip time and the changing ratio of the voice quality. A basic approach to allocate the bandwidth would be to measure the maximum achievable bandwidth, and to calculate the target bit-rate of the different flows according to the weights assigned to each flow by the bandwidth sharing policy. This policy is outside the scope of this paper, and we use fair sharing for the rest of the experiments/simulations.

In order to smooth out TCP burstiness either for flows with a large BDP (Bandwidth × delay product (BDP) refers to the product of a data link's capacity and its end-to-end delay or sometimes the data link's capacity times its round trip time) which have intra-flow burstiness, or for other flows which may show inter-flow burstiness, rate based traffic control is necessary. *VAGP* is designed to reduce TCP packet bursts by releasing packets smoothly into the network rather than in bursts. With long end-to-end delays, TCP application tends to inject enough packets to fill high BDP paths [19]. With wireless multihops this creates congestion, and drops

in the multihop portion require retransmissions, which reduce overall utilization. The method we propose controls the data transmission rate and dynamically adjusts the parameters for the actual TCP sending rate to consume residual bandwidth while keeping good VoIP quality. *VAGP* has two main functions: to tame the bandwidth discovery nature of the TCP at least for the wireless multihop portion making the TCP friendly to VoIP traffic, and to maintain high utilization. These functionalities are implemented in two modules that monitor/control both the voice and the TCP queues.

5.1 Voice Protection

As shown in **Fig. 15**, Voice Protection Module (VPM) monitors the network parameters of VoIP traffic, including network delay, network loss, and jitter loss. VPM calculates *MOS-score* of the monitored VoIP and estimates the bandwidth portion of TCP traffic. It then translates the available bandwidth into a TCP sending rate for each TCP flow using a bandwidth sharing method (i.e. fair share). Algorithm 1 briefly shows how *VAGP* works. It estimates the bandwidth portion for TCP traffic with the measured *MOS-score* which is classified using three thresholds for triggering TCP rate adjustment: good, fair and poor. TCP rate is adjusted either by a modest amount A, or aggressive A_{agg}, depending on how much current voice quality is different from Q_{fair}.

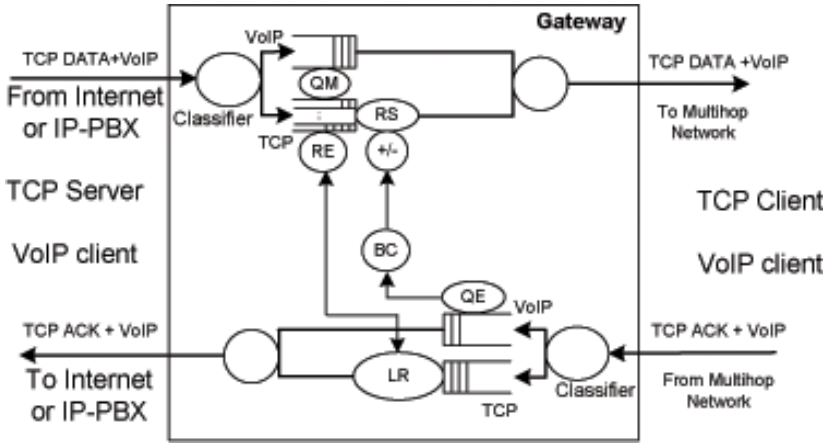

Fig. 15. *VAGP* functionality installed in the gateway monitors both VoIP and TCP traffic. It controls TCP sending rate based on the current estimation of VoIP quality. Voice Protection Module adjusts TCP data sending rates based on measured voice quality. TCP Improvement Module eliminates redundant packets which were reached to the destination and retransmits the dropped packets indicated by the duplication ACKs.

Algorithm 1 Voice Protection Module

Voice quality parameters: Q_{good}, Q_{fair}, Q_{poor};
TCP rate adjustment: A, A_{agg};
for all VoIP flow i **do**
  Measure *MOS-score*;
end for
for all TCP flow i **do**
  if *MOS-score* > Q_{good} **then**
    $R_i \Leftarrow R_{i-1}\ (1 + A_{agg})$;
  else if *MOS-score* $\in [Q_{fair}, Q_{good})$ **then**
    $R_i \Leftarrow R_{i-1}\ (1 + A)$;

```
    else if MOS-score ∈ [Q_poor, Q_fair) then
      R_i ⇐ R_i-1 (1 - A);
    else if MOS-score ≤ Q_poor then
      R_i ⇐ R_i-1 (1 - A_agg);
    end if
  end for
  wait(period);
```

VoIP quality assessment: In IEEE 802.11s standard, a tree-based routing (TBR) protocol is adopted as a viable proactive routing protocol for a WMN with user traffic flowing to/from a wired network through a gateway (i.e., a mesh portal). In this paper, we assume that the WMN of this paper supports a tree-based routing protocol to transfer the traffics from/to the VoIP clients through the same mesh paths to/from the gateway. VoIP quality can be assessed in two ways. First, the mesh router that manages the VoIP clients measures and reports the VoIP quality of the downlink VoIP flow to the gateway. Based on this information, VAP manages TCP traffic accordingly. However, this method increases the control signaling overhead between the routers and the gateway. It also increases responsiveness by congestion happened in the path due to the longer propagation delay from the mesh portal to the gateway. The second method is based on the assumption that the uplink and downlink travels the same path which shares the resources. Because uplink VoIP and downlink VoIP shares the same path, the uplink VoIP traffic is supported by the same QoS metric which downlink VoIP flow has. This allows the gateway to monitor the uplink VoIP flows to control TCP flows.

5.2 TCP Improvement

In Algorithm 1, VPM limits TCP share of the bandwidth in favor of VoIP traffic. However, it may cause TCP performance degradation due to queue overflow at the gateway or frequent TCP timer expiration at the sender while waiting too long for ACK delivery. TCP retransmission timer expiration may trigger unnecessary retransmission for packets in one of the following situations: 1) waiting in the queue at the gateway for the multihop access, 2) being delivered over the multihop network, and 3) already arrived at the destination. These retransmissions waste the multihop resource and decrease both VoIP and TCP performance. Because of larger packets, TCP also suffers higher packet drop ratio, which decreases the number of packets in the path, which finally results in poor performance in networks with high BDP.

To fully utilize the space left by voice traffic, TCP Improvement Module (TIM) uses two mechanisms - local recovery (LR) and redundant retransmission drop (RRD). Algorithm 2 outlines the functionality provided by these two procedures. When a duplicate ACK is received, LR retransmits the lost packet to the destination eliminating the delay of the packet transmission all the way from the sender. Packets with lower sequence number lower than *highest_seq* are dropped by RRD to prevent unnecessary transmission of packets already at the destination.

5.3 Queue Management

Instead of transmitting packets immediately upon receipt of TCP data, TIM delays transmitting packets to spread them out at the rate controlled by VPM, causing an increased queuing delay, which results in long RTT at the TCP sender. Queuing delay increases linearly until the number of packets reaches to min(*awnd*, *cwnd*). The local recovery mechanism of *VAGP* requires more delay while the lost packets in multihop are recovered by local

retransmission. With no packet drops and little few timer expirations, TCP sender may generate more packets perceiving the multihop as a large BDP network, which increases queuing even delay further. To address these problems, we use an adaptive queue-aware window control algorithm to minimize the queuing delay and buffer requirement at the gateway. When the multihop is bottlenecked, the queuing and network delay are much larger than the ones in the wired part:

$$q_i(t) + M_i(t) < \min(awnd(t), cwnd(t)) \quad (2)$$

Algorithm 2 TCP Improvement Module

```
for all received TCP packets do
  if packet type == TCP Data then
    read seq from TCP header;
    if seq ≤ highest_seq then
    Drop it (Redundant Retransmission Drop);
    else
      Send it to TCP Queue;
    end if
  else if packet type == TCP ACK then
    read seq from TCP ACK header;
    if duplicated ACK then
      Retransmit TCP data with seq + 1 (LR);
    else
      Update highest_seq;
    end if
  end if
end for
```

Algorithm 3 Queue-aware Window Control Operation

```
if q_i(t) < q_min_i then
  awnd ← q_max_i + M_i(t);
else if q_i(t) > q_max_i and R_i(t) > 0 then
  awnd ← q_min_i + M_i(t);
else if q_i(t) > q_max_i and R_i(t) == 0 then
  awnd ← 0;
else
  awnd ← (q_max_i+q_min_i)/2 + M_i(t);
end if
```

At an arbitrary time *t*, let us denote the flow queue length of the gateway as $q_i(t)$ and the number of packets in delivery over the multihop (*max_sent_seq - highest_ack*) as M_i. We use subscript *i* to index the connections. (2) represents the maximum number of packets in transit which are clearly smaller than TCP sender's window size. One can also see the queue length can be controlled by modifying the advertisement window *awnd* in the ACK packets. In the queue-aware window control management, we use the downstream queue length $q_i(t)$ to represent the queuing delay. The basic control strategy for the queue management is as follows: 1. TIM reduces *awnd* to $q_{min} + M_i$ as soon as the queue length exceeds a threshold, q_{max_i}; 2. TIM increases *awnd* to $q_{max\_i} + M_i$ when the queue length falls below a threshold q_{min_i}, here $q_{min\_i} < q_{max\_i}$; 3. if TCP bandwidth becomes 0, TIM reduces *awnd* to 0 to freeze all

timers at the TCP sender. Otherwise, *awnd* will be set to $\frac{q_{max\_i} + q_{min\_i}}{2} + M_i(t)$. We use q_{min_i} = 2 and q_{max_i} = 6 which is a reasonable choice to obtain low queuing delay while preventing buffer underflow.

This strategy is shown to reduce the queuing delay while keeping minimum number of packets - low queuing delay, as well as preventing the underflow of the queue. Our in-depth simulation results have shown reduction of RTT from 3 seconds to 310~msec in case of 4 hops in **Table 5** while the same TCP goodput and voice quality are achieved as the one without queue management strategy.

Table 5. Average VoIP quality (*MOS*), TCP goodput and throughput(Kbps) with the 70% VoIP traffic and 30% TCP; Internet delay = 30ms.

Hop count	Reno			TCP-GAP			VAGP		
	VoIP	TCP		VoIP	TCP		VoIP	TCP	
1	0.9	2596	2654	3.9	2587	2645	3.9	2587	2634
2	1.3	1070	1312	3.1	997	1189	3.6	750	791
3	1.9	552	869	2.3	545	853	3.7	664	705
4	2.5	309	558	3.3	287	509	3.7	325	374

5.4 *VAGP* Evaluation

In this section we evaluate *VAGP* for the voice quality achieved, TCP goodput (data rate seen by applications), and TCP throughput (data rate used by TCP). The difference between the latter two is due to losses in the wireless multihop which lead to TCP retransmissions.

String Topology: we first consider the simple string topology with both TCP and VoIP being transported across the same four wireless hops through the gateway *G* in **Fig. 1**. The TCP data originates across the Internet, on a node labeled *Server*, and is sent a wireless client attached to the access point labeled *MAP*8. VoIP traffic mayoriginate in the PSTN network, but travels across the same four wireless hops through the enterprise IP-PBX to/from another client attached to *MAP*8. *VAGP* functionality is added at the gateway node *G*. All the wireless links operate with 802.11a at 12Mbps. This is a good example of multihop connectivity in enterprise networks which use PSTN/IP-PBX for VoIP traffic and Internet for TCP traffic. G.729 codec produces 50 packets per second of 20 bytes each in each direction. We measure all the VoIP characteristics (delay, jitter and packet loss) contributing to *MOS-score*, and TCP goodput achieved using *Voice Adaptive Gateway Pacer*.

VPM monitors VoIP quality of the voice traffic from *G*, which is the entry point of the TCP flows and evaluates the *MOS-score* periodically. For these experiments, the parameter values which are used are Q_{good} = 3.9, Q_{fair} = 3.7, Q_{poor} = 3.6, Q_{choke} = 3.3 for VoIP quality and R_{good} = 5%, R_{fair} = 3%, R_{poor} = 10%, R_{choke} = 50% for TCP rate adjustment. These values are somewhat conservative to preserve voice quality, as drops are inevitable at sudden changes in channel conditions or traffic patterns. But, as we will show in the next experiments these values work across widely different patterns and conditions. They are in fact tuned to maintain the target of *MOS-score*=3.6, rather than specific to the topology and conditions.

In **Table 5**, we see that *VAGP* significantly outperforms TCP-GAP with respect to the basic premise of isolating the voice traffic. Using Reno, TCP traffic occupies the entire available bandwidth at cost of VoIP quality, while both flows in TCP-GAP share the available bandwidth giving a larger portion of the bandwidth to TCP traffic, which results in poor voice quality. In case of 4 hops running 7 voice calls and 3 TCP flows, *VAGP* achieves the fair share - *MOS-score* = 3.7 and 325Kbps TCP goodput, while TCP-GAP allocates more bandwidth to

TCP, which results in poor voice quality, *MOS-score* = 3.3. In fact, *VAGP* also achieves a higher aggregate goodput than TCP Reno and TCP-GAP. Thus, more VoIP calls can be supported using *VAGP*, when TCP gets an optimum share of the bandwidth.

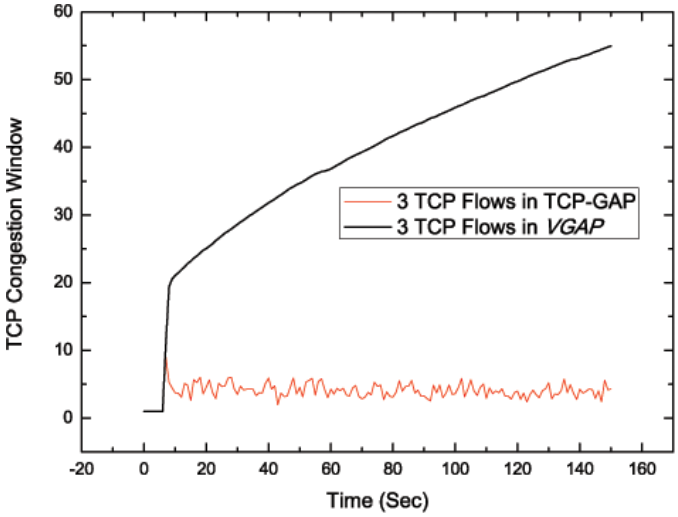

Fig. 16. Comparison of TCP Congestion window between TCP-GAP and *VAGP*, in case of 4hop in **Table 5**.

Fig. 16 shows that *VAGP*'s congestion window keeps increasing constantly, while TCP-GAP's fluctuates over time, although around a low value. The path is never perceived as congested with *VGAP*, and TCP's share is reduced due to pacing and possibly increased RTT. Being designed to operate over 4-hops, TCP-GAP shows poor performance within less than 4-hop topology. At 4 hops, *VAGP* produces a slight increase in RTT, but with a lower standard deviation as shown in **Table 6**.

Table 6. *VAGP* reduces variation in roundtrip time. This corresponds to 4 hops topology in **Table 5**.

RTT(ms)	Reno	TCP-GAP	*VAGP*
mean	242	288	312
std	271	209	100

In summary, for the string topology *VAGP* improves both VoIP performance and network utilization ($\frac{goodput}{throughput}$).

Mixed flow sources/destinations: We consider the case where three VoIP flows and one TCP flow originate in the wired domain and end at MP6, MAP3, MP7, and MAP8 respectively. Results are summarized in the first row of **Table 7**.

Table 7. *VAGP* performance compared with TCP-GAP with mixed flow destinations: string and tree. Columns show MOS score for voice, goodput and throughput for TCP. *VAGP* achieves VoIP protection as well as TCP performance with little sacrifice of the aggregate goodput. Standard TCP is not included because it doesn't perform in the simple string case.

Topology	TCP-GAP			*VAGP*		
	VoIP	TCP		VoIP	TCP	
String – 4 hops	2.6	437	732	3.7	309	346
Tree – 2 depth	1.1	985	1125	3.8	416	463

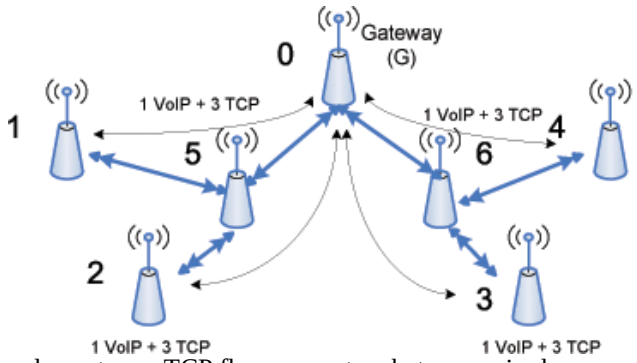

Fig. 17. Tree topology: downstream TCP flows are set up between wired servers (not shown) and leafs. VoIP sessions are set up between IP-PBX (not shown) and each leaf. Performance shown in **Table 7**.

Tree Topology: As a more complex topology, we consider a tree which consists of seven nodes placed in a tree two hops deep, with the gateway *G* is positioned at the root of the tree as shown in **Fig. 17**. When run separately, this topology can support either 2 VoIP calls per leaf, or 3199 Kbps of aggregate TCP goodput in case of 3 TCP flows per leaf node. If we mix 1 VoIP call and 3 TCP flows for each leaf, *VAGP* gets less goodput than TCP-GAP, but VoIP quality is maintained above the *MOS-score*=3.6 required (second row in **Table 7**).

***VAGP* with TCP variants**: We consider the same string topology as depicted in **Fig. 1** to show fairness between TCP variants under the control of *VAGP*. Running 5 TCP flows consisting of five TCP variants between wired server and MAP8 and 5 VoIP calls between PSTN and MAP8, we see that *VAGP* works well with any type of TCP. A fair share of around 20% is maintained between TCP flows while 5 VoIP calls are supported with an acceptable quality of *MOS-score* = 3.8 (**Table 8**).

Table 8. *VAGP* ensures fairness between flows, regardless of the TCP flavor they use.

Hop count	VoIP (MOS)	Total TCP 627 (Kbps)				
		RENO	C-TCP	CUBIC	VEGAS	WW
4	3.8	125 (19%)	126 (20%)	126 (20%)	124 (19%)	126 (20%)

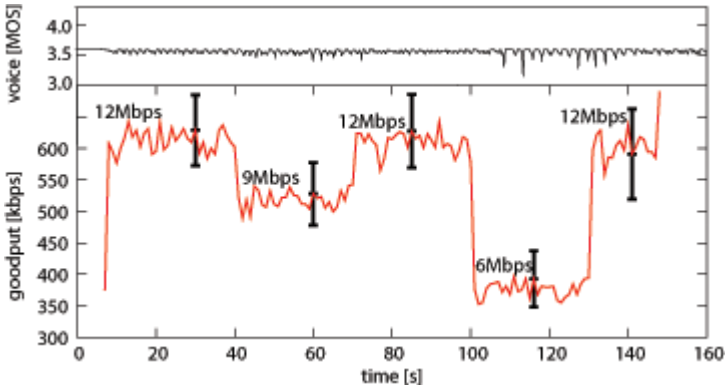

Fig. 18. TCP uses the residual bandwidth while VoIP quality is kept high, 5 calls 1 TCP over 4 hop string topology, G729a, 12Mbps, link capacity between MP6 and MP7 varies from 12 Mbps to 9 Mbps during 40s - 70s, 12 Mbps during 70s - 100s, 6 Mbps during 100s - 130s, 12 Mbps during 130s - 150s. The vertical bars represents the standard deviation of TCP goodput.

Dynamic Bandwidth Estimation: In this experiment using the same string topology as depicted in **Fig. 18**, we verify the capability of *VAGP* to support VoIP together with TCP flows when the actual capacity of the multihop link is varying. The timeline in **Fig. 18** shows how *VAGP* adjusts TCP bandwidth consumption for one TCP flow while keeping good VoIP quality for 5 VoIP calls when the capacity of a 802.11a link fluctuates. On the horizontal axis, we have time in seconds, the top graph shows the voice quality, while the bottom graph shows the TCP end to end goodput. The bitrate of the link between nodes MP6 and MP7 is changed in the following sequence: 12Mbps-9Mbps-12Mbps-6Mbps-12Mbps at times 40s, 70s, 100s, 130s. This emulates the behavior of a rate adaptation algorithm, or simply the variation in capacity of an indoor wireless channel. VoIP maintains average *MOS-score* above 3.6 for all calls during this period.

Quick Responsiveness: Looking at the detail of the timeline around 100s, we can see how *VAGP* swiftly reduces the TCP share *R* in order to protect VoIP. After an initial drop of R_{choke} = 50%, *R* is adjusted up using R_{good} = 5% to the feasible rate of around 0.38Mbps. This event is completed in 4s, ensuring that VoIP quality is maintained.

6. Conclusions

TCP and VoIP can coexist in interference-ridden multihop networks, but only with some policing help. None of the existing solutions (TCP variants, priority queues, MAC support) can protect VoIP, and produce low utilization as TCP wastes wireless capacity on retransmissions. We propose *VAGP*, a method that addresses both goals of VoIP protection and network utilization. It uses existing VoIP traffic to continuously estimate the amount of bandwidth that can be dedicated to TCP. Making TCP more CBR like is beneficial in two respects: it becomes more VoIP friendly, and minimizes self interference. The latter is also beneficial for TCP itself, which suffers disproportionately because of its large packets. Our approach, called as *VAGP*, has the following advantages:

- can be placed in the wireless gateway to monitor downlink TCP traffic.
- maintains end to end semantics and compatibility.
- doesn't require setting of parameters. An initial setting regarding the voice quality desired works across different conditions and situations.
- provides complete separation, good utilization, and swift responsiveness to changing conditions.
- performs well in a variety of conditions - various network topologies, several TCP flows, diverse network delays, different interference patterns, varying wireless capacity.

In real deployments, VAGP only needs modification of the access point, so it can be implemented as a tunable feature on the access points that support multihop. For future work, we plan on developing a multihop solution for TCP traffic originating from the multihop. This is likely to become a problem with the increase of the P2P traffic which produces more uplink traffic, and with an even more bursty nature. As opposed to the gateway approach in which all TCP traffic originating from Internet is distributed at MPs, the traffic originating from the multihop aggregates to MPs which results in bursty traffic even if the traffic from the client is regulated by MAPs.

Acknowledgments

This work was supported in part by POSDRU/89/1.5/S/62557 and by CNCSIS grant PN2

Resurse Umane/11 din 01.07.2009.

References

[1] J. Li, C. Blake, D. S. J. De Couto, H. K. Lee and R. Morris, "Capacity of ad hoc wireless networks," in *Proc. of 7th ACM International Conference Mobile Computing and Networking*, pp. 61-69, 2001. Article (CrossRef Link)

[2] K. Kim and S. Hong, "VoMESH: Voice over wireless MESH networks," in *Proc. of IEEE Wireless Communications and Networking Conf.*, pp. 193-198, 2006. Article (CrossRef Link)

[3] J. Yu, S. Choi, and J. Lee, "Enhancement of VoIP over IEEE 802.11 WLAN via dual queue strategy," in *Proc. of ICC*, pp. 3706-3711, 2004. Article (CrossRef Link)

[4] W. Wang, S. C. Dovrolis, M. Murray, and K. Claffy, "Solutions to performance problems in voip over 802.11 wireless lan," *IEEE Trans. on Vehicle Technology*, vol. 54, no. 1, pp. 366-384. Jan. 2005. Article (CrossRef Link)

[5] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol.17, no. 6, pp.27-35, Nov.-Dec. 2003. Article (CrossRef Link)

[6] K. Xu, K. Tang, R. Bargrodia, M. Gerla, and M. Bereschinsky, "Adaptive bandwidth management and qos provisioning in large scale ad hoc networks," in *Proc. of IEEE Military Communications Conf.*, vol. 2, pp. 1018- 1023, 2003. Article (CrossRef Link)

[7] F. Y. Li, M. Haugea, A. Hafslund, L. Kure, and P. Spilling, "Estimating residual bandwidth in 802.11-based ad hoc networks: An empirical approach," in *Proc. of 7th Int. Symposium on Wireless Personal Multimedia Communications* (WPMC '04), vol. 7, no. 10, pp. 1228-1241, 2004. Article (CrossRef Link)

[8] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proc. of 15th Int. Conf. on Distributed Computing Systems*, pp. 136-143, 1995. Article (CrossRef Link)

[9] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. of 1st Int. Conf. on Mobile Computing and Networking*, pp. 2-11, 1995. Article (CrossRef Link)

[10] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "Gateway adaptive pacing for TCP across multihop wireless networks and the internet," *ACM MSWiM*, pp. 173-182, 2006. Article (CrossRef Link)

[11] H.-Y. Wei, S.-C. Tsao, and Y.-D. Lin, "On shaping TCP traffic at edge gateways," in *Proc. of 4th GLOBECOM*, vol.2, no.3, pp.833-839, 2004. Article (CrossRef Link)

[12] H. Fei and B. Yu, "Performance evaluation of wireless mesh networks with self-similar traffic," in *Proc. of Int. Conf. on Wireless Communications, Networking and Mobile Computing*, pp.1697-1700, 2007. Article (CrossRef Link)

[13] R. G. Cole and J. Rosenbluth, "Voice over IP performance monitoring," *SIGCOMM Computer Communication Review*, vol. 31, no. 2, pp. 9-24, April 2001. Article (CrossRef Link)

[14] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. of 22nd IEEE Int. Conf. on Computer Communications*, vol. 3, pp. 1744-1753, 2003. Article (CrossRef Link)

[15] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130-137, Jun 2001. Article (CrossRef Link)

[16] L. Xu and I. Rhee, "CUBIC: A new TCP-Friendly high-speed TCP variant," in *Proc. of 3rd Int. Workshop on Protocols for Fast Long-Distance Networks*, vol. 42, 2005. Article (CrossRef Link)

[17] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proc. of 25th IEEE Int. Conf. on Computer Communications*, pp. 1-12, 2006. Article (CrossRef Link)

[18] Dragoş Niculescu, Interference map for 802.11 networks, *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 339-350, October 24-26, 2007. Article (CrossRef Link)

[19] K. Chen, Y. Xue, S. H. Shah, and K. Nahrstedt, "Understanding bandwidth-delay product in mobile ad hoc networks," *Elsevier Computer Communications Special Issue on Protocol Engineering for Wired and Wireless Networks*, vol.27, pp.923-934, 2003. Article (CrossRef Link)
[20] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek, "The Click modular router", *ACM Transactions on Computer Systems*, vol. 18, pp. 263-297, August 2000. Article (CrossRef Link)

Kyungtae Kim received a M.S. degree in Department of Computer Science from the Columbia University in 2000, NY and his Ph.D. degree in the department of Electrical and Computer Engineering at Stony Brook University in 2006 and currently is an adjunct professor in the Department of Electrical and Computer Engineering at Stony Brook University as well as working for NEC Laboratories America Inc during 8 years until now in the areas of multimedia communication over the wireless network, cognitive radio, wireless mesh networks, and mobility management over the heterogeneous networks including 802.16/802.11 networks.

Dragoş Niculescu received a B.E. from Politehnica University of Bucharest in 1996, and a Ph.D. from Rutgers University in 2004, both in computer science. He then spent five years as a researcher at NEC Laboratories America in the Mobile Communications and Networking Research group working on wireless networking related problems. He is currently an assistant professor at ETTI, University Politehnica of Bucharest, Romania, teaching and researching mobile networking, wireless meshes, and software defined radios.

Sangjin Hong received the B.S and M.S degrees in EECS from the University of California, Berkeley. He received his Ph.D in EECS from the University of Michigan, Ann Arbor. He is currently with the department of Electrical and Computer Engineering at Stony Brook University. Before joining Stony Brook University, he has worked at Ford Aerospace Corp. Computer Systems Division as a systems engineer. He also worked at Samsung Electronics in Korea as a technical consultant. His current research interests are in the areas of multimedia wireless communications and digital signal processing systems, reconfigurable VLSI Systems and optimization. Prof. Hong is a Senior Member of IEEE and a member of EURASIP journal editorial board. Prof. Hong served on numerous Technical Program Committees for IEEE conferences.