Coexistence of VoIP and TCP in Wireless Multihop Networks

Kyungtae Kim and Dragoş Niculescu, NEC Laboratories America
Sangjin Hong, Stony Brook University — SUNY

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 many solution approaches, such as enhanced TCP variants, priority queues, bandwidth limitation, and traffic shaping, do not always achieve the coexistence goals. Enhanced TCP variants (Reno, Vegas, C-TCP, CUBIC, Westwood) generally fail to protect VoIP in wired-wireless multihop scenarios. Priority schemes, including those built into the 802.11 MAC such as RTS/CTS or 802.11e, do not account for the interference nature of wireless multihop. Finally, bandwidth shaping and window control are valid tools to control TCP, but come with their own trade-offs.

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 becoming more prevalent in homes and institutions is voice over Internet Protocol (VoIP). This capability is becoming 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 very stringent delivery requirements. While mechanisms to provide for this quality of service (QoS) exist in wired networks, in the popular 802.11-based networks they were only an afterthought.

In this article we show that coexistence between these two popular traffic types is difficult in multihop networks, and investigate different methods that can be used to facilitate it. Although QoS enhancements such as 802.11e were added, they do not really address the central problem of multihop networks, which is interference — in particular, self-interference for wireless multihop. Self-interference is the situation in which packets of the same flow compete for the medium when they are transmitted from successive hops. This can be transmission interference (carrier sense level) or reception interference (hidden terminal level).

The interaction between TCP and VoIP over a multihop network is complex; here is a summary of the most important points:

- *TCP is an end-to-end protocol*. There are no explicit signaling mechanisms in the network to tell 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 explicit knowledge it receives from the other peer. TCP needs to be aggressive in discovering link bandwidth because that is how it can achieve high utilization. This is achieved using large windows, which aggravates channel contention on wireless links.
- *TCP produces bursty traffic, while VoIP produces uniform traffic*. In the so-called *slow* start phase, TCP doubles its window for each acknowledgment (ACK) received, in reality an exponential increase in bandwidth consumption. This creates trains of packets that hog the medium for prolonged times. VoIP, on the other hand, needs regularity in 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 recovers 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 the TCP congestion control mechanism reacting to link layer errors. Performance studies of TCP over 802.11-based multihop show that standard TCP behavior may lead to poor performance because of packet drops

 0163-6804/09/$25.00 © 2009 IEEE

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 can be implemented using a data and/or ACK pacing mechanism.

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 have less success, so many of them would be retransmitted across multihop links, thus generating even more load that in turn generates more interference.

•*Multihop interference is a non-local phenomenon*. A wireless node cannot determine by itself what the interference conditions are in its neighborhood. Factors affecting regional interference are actual paths, load, physical distance between nodes, collision domains, and hidden terminal relationships.

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. VoIP is mostly constant bit rate, has tight delay and loss requirements, and should always be served prior to TCP traffic. Surprisingly, 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 article investigates the behavior of TCP and VoIP traffic in a shared network. 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 examine some recently proposed TCP variants, some of which have been tailored especially for wireless networks, and a number of classical techniques of supporting traffic with different regimes.

Existing Work

A large amount of research has focused on the optimization of TCP performance in wireless networks. 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. Most packet losses experienced in multihop wireless networks are due to a) wireless links with high error rate; b) hidden terminals and channel contention at the intermediate nodes; or c) buffer overflow at the ingress node in bandwidth-asymmetric networks.

On the other hand, packet drops in the Internet are almost always due to buffer overflows at the routers. A solution to this network convergence problem [1] lies in splitting the TCP connection at the node interfacing the wired and wireless parts of the network, called 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 a wireless network can be optimized independent of the TCP applications. However, it requires extra overhead to maintain two connections for each flow. It also violates end-to-end TCP semantics and requires a complicated handover process.

•*Link layer solutions*: The idea is to make the wireless link layer look similar to the wired case from the perspective of TCP. The most relevant and interesting proposal is the snoop protocol [2]. 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 within the intermediate nodes of a multihop network [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 [4, 5] can be implemented using a data and/or ACK pacing mechanism.

Another recent work [6] acknowledges that congestion control in multihop networks depends on complex interference patterns, and proposes additive increase/multiplicative decrease (AIMD) based mechanisms for fairness and efficiency. All the mentioned solutions focus only on TCP, without regard to its effect on real-time traffic.

For mixed TCP and VoIP traffic, [7] dispels the fears that extensive use of unresponsive VoIP in the Internet would reduce the share obtained by TCP. Their model assumes that call drop probability increases at lower quality, so at a macroscopic level VoIP is actually responsive to congestion.

Mixing TCP and VoIP

It is well understood from queuing theory that bursty traffic produces higher queuing delays, more packet losses, and lower throughput. It has been observed that TCP's congestion control mechanisms and self-clocking create extremely bursty traffic in networks with large bandwidth-delay products, cause long queues, and increase the likelihood of massive losses. Wireless multihop network traffic tends to have self-similar behavior, which is harmful to traffic requiring a stable bit rate, such as VoIP or streaming.

Figure 1 illustrates the wired/wireless hybrid network we consider in this survey. The multihop extension forwards TCP traffic from wired Internet and VoIP calls to/from an IP-private branch exchange (PBX) through the gateway. As shown in the figure, we are concerned mostly with TCP data flowing from the gateway G to the client. The multihop leg is where VoIP needs to be protected from TCP. For the quantitative arguments, we use a topology functionally equivalent to the one in Fig. 1, which includes four wireless hops as the access network between the gateway and the mobile client.

Difficult Coexistence

To understand the difficulties in supporting VoIP, we start with a short primer on VoIP quality requirements. VoIP traffic is constant bit

rate (CBR), and for certain vocoders (G711, G729a), its quality can be estimated using packet loss and mouth-to-ear (one-way) delay. Figure 2 shows the values of mean opinion score (MOS) with respect to network delay and total loss for 60 ms playout buffer and 25 ms vocoder delay. In order to obtain an MOS of 3.6 (comparable to GSM quality), the network has to deliver all packets in less than 160 ms, or 98 percent in less than 104 ms. For G.729a used in the rest of the article, 3.9 is the maximum quality achievable, but we consider 3.6 to be acceptable quality.

First, we show that burstiness is the main cause of reduced VoIP quality. To this end, we experiment with various packet patterns as shown in Fig. 3. In these scenarios we have the same mean offered rate for data (550 kb/s using large 1500-byte packets), but with different burst lengths. The rest of the capacity is filled by small VoIP packets (20 bytes payload).

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 (throughput, one-way delay) suffer because of increased burst length. In fact, the multihop can support five voice calls with one data packet burst, but only three voice calls with five data packet bursts. In Internet scenarios, when long delays can be present on the Internet portion, a TCP flow is expected to require windows much larger than five packets, and therefore produce even more degradation for itself and for VoIP.

In the same topology of four hops we try to establish the kind of performance we can expect from each type of traffic, and in combination. Running each of the four hops at 12 Mb/s, we can support either 11 VoIP calls or 1.35 Mb/s of TCP. However, if we mix five VoIP calls and three TCP flows, we find that voice quality is below the minimum acceptable (MOS < 2) while TCP flows get a cumulative 0.84 Mb/s. *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 ACKs 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 the receiver's acceptable number of bytes. If the window size is too large, the sender is allowed to inject more traffic than the network can handle.

Given a wireless multihop network, there is a TCP window size W^* at which TCP's bandwidth consumption is appropriate. This window size depends on many conditions, including the 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 reduced TCP performance if VoIP traffic is not present.

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/or voice quality degradation. We first consider the various enhancements to TCP proposed by the research community in recent years: Reno, Vegas, Westwood, CUBIC, and Compound TCP (C-TCP). Alternately, VoIP can be separated from TCP by some reservation method that would require some form of constraining TCP at the gateway. In this article we consider a typical downstream traffic situation in which TCP sources across the Internet send bulk traffic to a user connected to the multihop network.

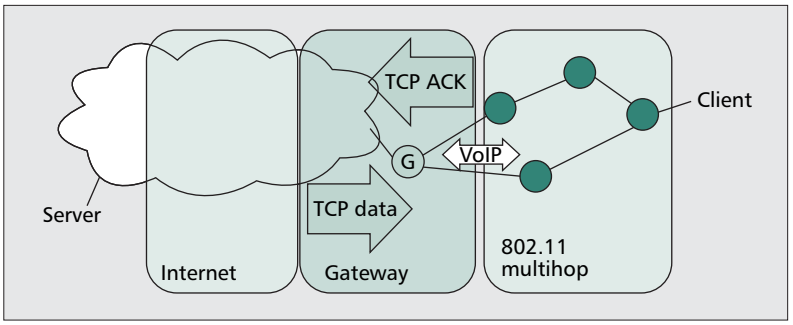

Figure 1. *802.11-based multihop as an access network for VoIP and TCP. We consider downstream TCP traffic in which data flows from servers across the Internet, through the gateway to wireless clients, and TCP ACKs travel in the opposite direction. VoIP traffic is symmetric and bidirectional.*

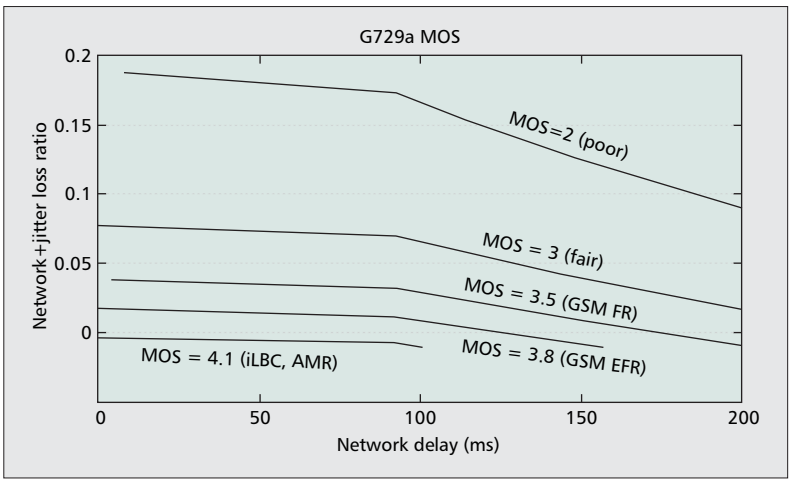

Figure 2. *For certain vocoders, such as G729a, VoIP quality (MOS) can be computed as a function of loss and one-way delay. Loss includes packets lost in the network and packets that miss their deadline because of jitter. For more details, see [8].*

TCP Variants

We first compare several TCP variants designed to improve the performance of TCP in traditional wired networks. For performance, the variants Reno, Vegas, and Westwood are compared in [9], and CUBIC and C-TCP in [10].

TCP Reno is the traditional algorithm in most operating systems currently deployed, and we consider it as a base case. 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 ACK, until a timeout occurs or a duplicate ACK

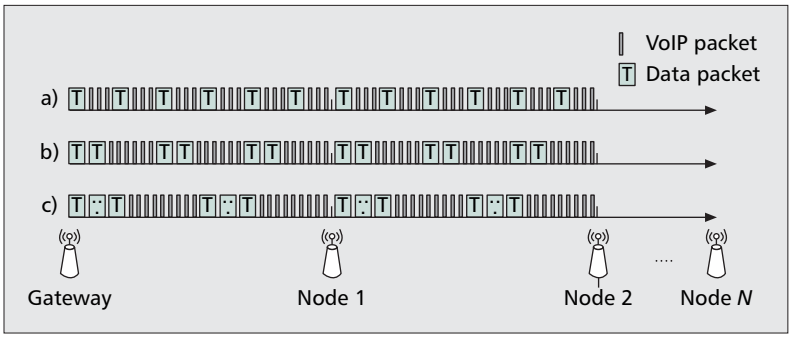

Figure 3. *Simulation setup to estimate impact of data burstiness on VoIP traffic: the data rate offered is the same in all situations, but the burst length is increased. We measure the amount of voice calls supported (MOS > 3.6) in the remaining space. Results are shown in Table 1.*

Burst length	VoIP calls	VoIP Loss (%)	VoIP delay (ms)	Data throughput (kb/s)	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

Table 1. *VoIP and data statistics as data burstiness increases as described in Fig. 3; 5 VoIP calls and 550 kb/s of data offered; 4-hops string topology, 12 mb/s, 802.11a.*

is received. The latter implies that a packet has been lost, which signals that the sender is transmitting packets faster than the network can handle. 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 timeout. Reno also includes fast retransmit and recovery mechanisms that make it possible to quickly recover lost packets.

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.

TCP Westwood enhances the window control and backoff process. Westwood relies on end-to-end rate estimation. The 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 *cwnd* and slow start threshold *ssthresh* after a congestion episode (i.e., after three duplicate ACKs 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 keep up with the rapid changes in a hybrid wireless network.

TCP CUBIC was proposed to address the underutilization problem due to the slow growth of TCP congestion windows in high-speed networks. The window growth function is updated with the time elapsed since the last loss event, so its growth is independent of network delay. This means the sender is allowed to put more packets without waiting for the ACKs in a network with large bandwidth delay products, by probing the bottleneck bandwidth quickly. CUBIC or one of its variants has been the default in Linux kernels since v. 2.6.19.

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 the two approaches. C-TCP can rapidly increase sending rate when a network path is underutilized, but gracefully retreat in a busy network when bottleneck queues grow. C-TCP is the algorithm included in 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 the bottleneck link will support. A TCP sender's self-clocking depends on the arrival of ACKs at the same spacing at which the receiver generated them. If these ACKs spend any time sitting in queues during their transit through the network, their spacing may be altered. When ACKs arrive closer together than they were sent, the sender might be misled into sending more data than the network can accept, which could lead to congestion and loss of efficiency. Also, cumulative 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 packets, opening the window in a large burst.

The loss-based congestion avoidance mechanism adopted by TCP variants causes a periodic oscillation in the window size in wireless multihop with high packet loss. This variation in packet rates leads to a fluctuation in the delivery time of packets. In turn, window size oscillation results in larger delay jitter for other traffic and inefficient use of the available bandwidth due to many retransmissions.

We compared these five TCP variants with respect to capacity to coexist with VoIP and utilization of the multihop. Figure 4 shows that all TCP variants fail to protect VoIP in a simple shared environment. Even when TCP experiences higher packet error rate, VoIP flows are not getting reasonable quality: MOS < 3 with 10 VoIP calls.

What TCP variants do is to increase TCP throughput with large window sizes. Vegas exhibits both better VoIP protection and utilization of the multihop links due to its balanced

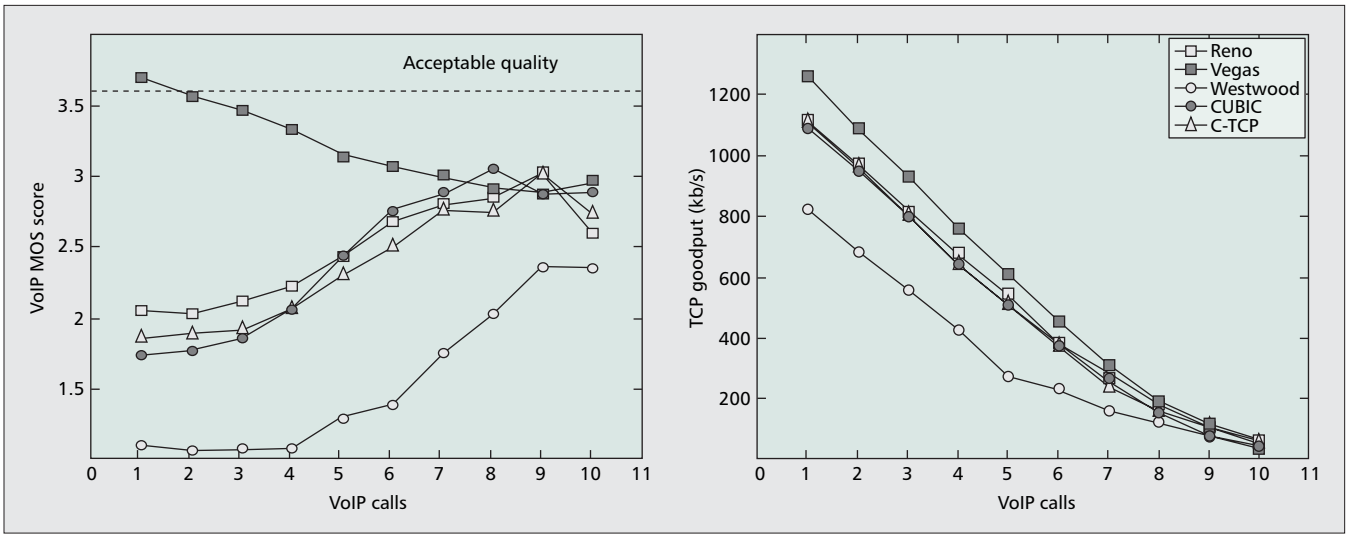

Figure 4. *Left: VoIP quality with different types of TCP; right: TCP goodput. A small number of calls are largely disrupted by TCP traffic, and a large number of calls leads to a higher TCP packet error rate, resulting in increased voice quality while it still fails to protect voice quality.*

congestion control with a low number of VoIP flows. Surprisingly, Westwood, which is designed specifically for lossy links, performs worse on both measures, wasting half the capacity on retransmissions (*goodput/total_sent* ~ 0.5, not shown in the figure).

Controlling TCP Traffic

In fact, an even more likely situation is that none of the TCP endpoints can be controlled because upgrading TCP is infeasible or undesirable for other reasons. Even enhanced TCP endpoints cannot possibly protect wireless hops in the middle. We therefore explore other methods to enable coexistence at the gateway into the wireless multihop. TCP traffic control can be performed using classical methods such as priority queues and traffic shaping, or by instrumenting TCP packets to manipulate a receiver's advertised window. We now look at each of these mechanisms in more detail.

Priority Queues — One solution to harmonize VoIP and TCP traffic is the use of priority queues. As a scheduling mechanism, priority queues are used in one hop scenarios to implement classes of traffic.[1] We simulated priority queues in *ns*-2, allocating the highest priority to outgoing VoIP traffic at all nodes. We found that only 20 percent 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 cannot protect from interference generated two or three hops away. On the contrary, it increases packet burstiness while building up TCP packets in the queue. These localized approaches cannot provide a solution to a global problem of hidden terminals interfering across several hops.

Window Resizing — TCP bandwidth discovery operates from the sender and cannot be easily manipulated. The advertised window of the receiver, however, can be decreased 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 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. This method limits the total number of TCP data packets in transit between the endpoints. If the gateway changes the advertisement window based on the network status, TCP throughput can be limited close to its entry point. By keeping the window size small, retransmission and fairness problems among TCP flows are also relieved.

We experimented with various values for the advertised window and found that smaller windows are more beneficial than larger windows. This is a consequence of the fact that TCP's share of the wireless medium needs to be reduced [11].

TCP Data/ACK Pacing — One problem that is not solved by window resizing is that of packet bursts. TCP pacing promises to reduce the 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 over the air, the next packet is scheduled no earlier than *pkt_size*/R.

The gateway chooses a rate R based on the network status to determine how much to send as well as when to send. One way to understand the impact of pacing is to consider burstiness from the perspective of network delay, jitter, and packet loss. With bursty traffic, packets arrive all at once at the gateway. As a result, queuing delay and delay jitter of VoIP packets grows linearly with TCP load due to large packet size, even when the load is below capacity. When car-

[1] *802.11e uses multiple queues for downlink traffic and preferential contention parameters for uplink traffic in order to offer priority to QoS traffic.*

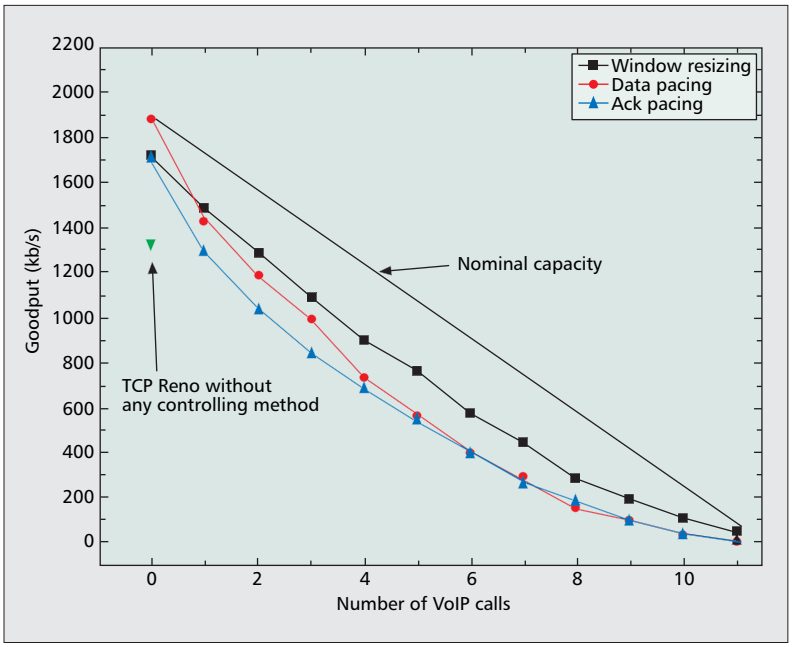

Figure 5. *Shared capacity between TCP and VoIP. TCP bandwidth is controlled using data pacing, ACK pacing, and window control. Simulation setup is that in Fig. 1, with an Internet delay of 30 ms. Each measurement point indicates the amount of (controlled) TCP supported such that the voice calls have acceptable quality (MOS > 3.6). All methods are bounded by the nominal capacity of the network.*

rying VoIP traffic at high load, 802.11 links are still perceived by TCP as being relatively free. This ignores the interference side effect large TCP packets produce several hops away.

In our measurements the shaper offered protection to VoIP at the cost of sacrificing available bandwidth for retransmissions. While providing benefits like small buffer size at the shaper, 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 queuing delay. The increased queuing delay easily causes the 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 a higher number of hops, and does not require instrumentation of TCP packets.

Reviewing the candidates, it is clear that in order to share multihop links with VoIP, TCP should use some form of bandwidth limitation. Priority queue mechanisms, including 802.11e, cannot really protect voice, especially in the hidden terminal case. If contending traffic is out of carrier sense range but still in interference range, it is not possible to use arbitration interframe space (AIFS) preferential priorities, or even request to send/clear to send (RTS/CTS).

Window Resizing vs. 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 multihop. For voice emulation, we generated and analyzed flows of 50 packets/s, 20 bytes of voice payload per packet in each direction that emulate G729a traffic. The reason for using this type of traffic is that most VoIP Session Initiation Protocol (SIP) phones (Zyxel, Utstarcom, Netvox) support it, and can be evaluated using the loss and 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.

Network Utilization with TCP and VoIP — In Fig. 5 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 of 3.6 for the VoIP traffic. While all three control methods achieve some amount of sharing between the two types of traffic, window control attains better utilization. The benefit of constraining TCP is visible even without VoIP traffic when plain TCP wastes capacity on retransmissions achieving a lower goodput.

Scalability with Respect to Number of TCP Flows and Amount of Internet Delay — From previous experiments we can conclude that all methods can be used to control TCP rate, with window control having a slight advantage by providing higher utilization. We then experimented with increasing the number of TCP flows and found that window control cannot support more than 11 TCP flows when three voice calls are present as it requires a window size less than one packet. When using internet delay of 150 ms (RTT), the required window sizes are larger, but only 15 TCP flows can be supported due to the same reasons. 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 four-hop topology: six calls are being supported, and a remaining bandwidth of 600 kb/s can be used by TCP when RTT $= 2 \times 8 \times 1500/600{,}000 = 40$ ms across the multihop, according to

$$RTT = \frac{W}{Bandwidth}.$$

When an Internet delay of 100 ms is included and TCP faces an RTT = 140 ms end to end, in order to achieve the 600 kb/s 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. Window control is not able to prevent packet burstiness created by either many TCP flows or large windows required by flows with large Internet delay. We conclude that *shaping is more robust against varying conditions such as number of TCP connections and increased Internet delay, but comes at the cost of reduced utilization*, especially when there are few TCP flows. In addition, it does not require any modification of TCP packets, which can be desirable for a high-speed implementation or when transporting

encrypted traffic. TCP window control has the potential to achieve better utilization, but is more sensitive to external conditions, and it exhibits highly dynamic behavior that depends on load, delay, and number of connections.

SUMMARY

TCP and VoIP cannot coexist in interference-ridden multihop networks by simply sharing the medium. Even enhanced TCP variants (Vegas, C-TCP, CUBIC, Westwood) do not offer any protection to VoIP traffic and generally lead to poor utilization, including the wireless-specific Westwood. Classical methods such as priority queues, 802.11e, and RTS/CTS do not really help in wireless multihops with hidden terminals. The reason hidden terminals are prevalent in multihop wireless networks is the non-local self-interference phenomenon. The solution is to control TCP traffic before entering the multihop network, which provides 40 percent improvement even when there is no VoIP to be protected. We examine two TCP control mechanisms: TCP advertisement window resizing, and TCP data and ACK pacing. We found that both control methods can limit the wireless resources taken by TCP, but have different trade-offs with respect to utilization and scalability.

In this article we examine the coexistence problem for downstream traffic from the gateway to the clients attached to the multihop, but there are a number of related problems left to address:

- The methods examined here have straightforward application only when the wireless capacity is fixed. In reality, the capacity is highly variable depending on a multitude of factors: the 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, and TCP traffic arrival model (HTTP, FTP, interactive). To support VoIP under such varying conditions, the basic control tools examined here should be used in conjunction with methods to dynamically estimate available bandwidth in real time.
- Uploading is becoming increasingly popular with P2P traffic, upload of large videos/images, and streaming. TCP traffic originating at multihop clients leads to a different coexistence problem as TCP flows have different origins in the multihop and cannot be easily controlled from a centralized location.
- There are alternative protocol solutions for both real-time media delivery and data transport. Some successful interactive streaming applications, such as Skype, use TCP for voice delivery. The emerging IP multimedia subsystem framework allows complex delivery schemes involving synchronization between streams with different transports. SCTP is a relatively new transport protocol that can handle both data and streaming media.

All these developments are likely to complicate the coexistence problem even more.

REFERENCES

[1] A. Bakre and B. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 15th Int'l. Conf. Distrib. Comp. Sys.*, May 30–June 2, 1995, pp. 136–43.
[2] H. Balakrishnan *et al.*, "Improving TCP/IP Performance over Wireless Networks," *Proc. 1st ACM MobiCom '95*, 1995, pp. 2–11.
[3] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?" *IEEE Commun. Mag.*, vol. 39, no. 6, June 2001, pp. 130–37.
[4] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "Gateway Adaptive Pacing for TCP Across Multihop Wireless Networks and the Internet," *ACM MSWiM*, Torremolinos, Spain, 2006.
[5] H.-Y. Wei, S.-C. Tsao, and Y.-D. Lin, "On Shaping TCP Traffic at Edge Gateways," *Proc. IEEE GLOBECOM '04*, vol. 2, Nov. 29– Dec. 3, 2004, pp. 833–39.
[6] S. Rangwala *et al.*, "Understanding Congestion Control in Multi-Hop Wireless Mesh Networks," *Proc. 14th ACM MobiCom '08*, 2008, pp. 291–302.
[7] T. Bu, Y. Liu, and D. Towsley, "On the TCP-Friendliness of VoIP Traffic," *Proc. 25th IEEE INFOCOM '06*, Apr. 2006, pp. 1–12.
[8] R. G. Cole and J. Rosenbluth, "Voice over IP Performance Monitoring," *Comp. Commun. Review*, vol. 31, no. 2, Apr. 2001, pp. 9–24.
[9] N.-C. Wang *et al.*, "Performance Enhancement of TCP in Dynamic Bandwidth Wired and Wireless Networks," *Wireless Pers. Commun.*, vol. 47, no. 3, 2008, pp. 399–415.
[10] F. Nemeth *et al.*, "TCP Limit: A Streaming Friendly Transport Protocol," *NGI 2008,* Apr. 2008, pp. 139–45.
[11] Z. Fu *et al.*, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," *Proc. 22nd IEEE INFOCOM*, vol. 3, Mar. 30–Apr. 3, 2003, pp. 1744–53.

BIOGRAPHIES:

KYUNGTAE KIM (kyungtae@nec-labs.com) received his B.S. degree from the Department of Electronics Engineering, Hanyang University, Korea, his M.S. degree in computer science from Columbia University, New York, and his Ph.D. degree at the Department of Electrical and Computer Engineering, Stony Brook University, New York, in 2006. He is currently working with NEC Laboratories America, Inc., Princeton, New Jersey, in the areas of multimedia communication over the wireless network, mobility management, fixed mobile convergence, and mobile unified communication.

DRAGOŞ NICULESCU (dragosg@nec-labs.com) received a B.S. in computer engineering from Politehnica University of Bucharest in 1994 and a Ph.D. in computer science from Rutgers University in 2004. He is currently a researcher at NEC Laboratories America in the Mobile Communications and Networking Research group working on wireless networking related problems: 802.11 meshes, VoIP routing and QoS, and software defined radios.

SANGJIN HONG [SM] (snjhong@ece.sunysb.edu) received B.S. and M.S. degrees in electrical engineering and computer science (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, New York. Before joining Stony Brook, he 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 low-power VLSI design of multimedia wireless communications and DSP systems, reconfigurable SoC design and optimization, VLSI signal processing, and low-complexity digital circuits. He has served on numerous technical program committees for IEEE conferences.

The reason hidden terminals are prevalent in multihop wireless networks is the non local self interference phenomenon. The solution is to control TCP traffic before entering the multihop, which provides 40 percent improvement even when there is no VoIP to be protected.