

Performance of VoIP in a 802.11 Wireless Mesh Network

Dragoş Niculescu, Samrat Ganguly, Kyungtae Kim, Rauf Izmailov
NEC Laboratories America, Princeton, NJ
{dragos, samrat, kyungtae, rauf}@nec-labs.com

Abstract—Performance in multihop wireless networks is known to degrade with the number of hops for both TCP and UDP traffic. For VoIP, the wireless network presents additional challenges as the perceived quality is dependent on both loss and delay. We investigate several methods to improve voice quality and present experimental results from an 802.11b testbed optimized for voice delivery. Use of multiple interfaces, path diversity and aggregation are shown to provide a combined improvement of 13 times in number of calls supported in our 15 node 802.11 mesh system.

VoIP, multihop, mesh

I. INTRODUCTION

In the recent past, there has been a tremendous proliferation of VoIP services in both residential homes and corporate offices. For example, FCC data indicates that there will be 3-7 million residential VoIP lines by the end of 2005. In the corporate sector, the percentage of VoIP lines will become 44% by 2008. In addition, the Skype service [1] providing free internet calls has recorded more than 10 billion minutes of call time in its first year of inception. The cost savings achieved by VoIP by using existing data infrastructures along with easy deployment benefits are the main reasons driving the steady growth of VoIP.

At the same time, VoIP over wireless LAN (WLAN) has the potential of becoming an important application due to the ubiquity of the WLAN in homes and offices. With the advent of dual cell phone handset with WiFi capabilities and soft-phones over PDAs, carrying voice over the WLAN is gaining a significant importance. Once VoIP over WLAN becomes widespread, most cell phone or WiFi handset owners will migrate to using VoIP over WLAN inside the administrative boundaries of the enterprise buildings, campuses, public places such as airports or even in WLAN equipped homes.

Providing VoIP users with true mobile phone services having the freedom of roaming requires wide area wireless coverage, and IEEE 802.11-based multihop wireless mesh networks have been considered a practical solution for wide area coverage. The benefits of mesh network compared to wired LAN connecting WiFi access points are: i) ease of deployment and expansion; ii) better coverage; iii) resilience to node failure; iv) reduced cost of maintenance. Such a mesh network has the potential of creating an enterprise-scale or community-scale wireless backbone supporting multiple users while driving these users from using fixed phones to wireless VoIP phones. A typical usage scenario is shown in Figure 3.

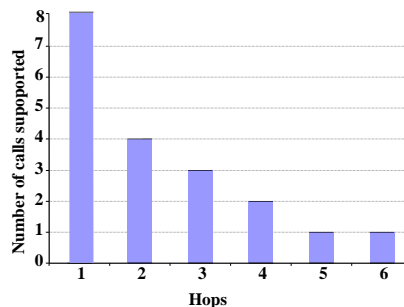


Fig. 1. In a linear topology, capacity degrades with the number of hops.

However, supporting delay sensitive realtime applications such as VoIP over wireless mesh networks is challenging. Although convenient and cheap, voice service over WLAN faces a number of technical problems: a) providing QoS sensitive VoIP traffic in presence of best effort TCP data traffic; b) packet loss due to channel interference by using unlicensed bands (2.4GHz, 5GHz); c) high overhead of the protocol stack - 802.11/IP/UDP/RTP for each VoIP packet with 20bytes payload. The above problems become even more severe when supporting VoIP over multihop mesh networks. In a multihop wireless network operating on a single channel, the UDP throughput decreases with number of hops for properly spaced nodes and is shown to be between 1/4 and 1/7 that of single hop capacity [2]. This phenomenon of self interference is produced by different packets of the same flow competing for medium access at different nodes. When all nodes are within interference range, the UDP throughput in a linear topology can degrade to $\frac{1}{n}$, where n is the number of hops.

As shown in Figure 1, our experiment on a real mesh testbed with G.729 encoded VoIP calls indicates that the number of supported medium quality calls decreases with the number of hops for a simple linear topology. In a mesh network with 2Mbps link speed, the number of supported calls reduces from 8 calls in single hop to one call after 5 hops. This significant reduction in the number of supported calls can be attributed to following factors: a) decrease in the UDP throughput because of self interference; b) packet loss over multiple hops and c) high protocol overhead for small VoIP packets. In this work we focus on designing a 802.11 based wireless mesh network that can efficiently support the VoIP calls. Specifically, our main objective is to increase the number of calls that a multihop

mesh network can support. We address several performance optimization issues that lead to significant benefits in capacity and in the quality of VoIP calls.

We describe our implementation of a 802.11 wireless mesh designed specifically to provide various VoIP related services. We focus on two important problems in supporting VoIP over wireless mesh network: increasing VoIP capacity and maintaining QoS under internal and external interference. We evaluate the performance of VoIP over the mesh network and provide various approaches for optimization of the overall system.

In particular, for increasing capacity in supporting more number of VoIP calls, we investigate on the following three directions: use of multiple interfaces, efficient routing, and use of multihop packet aggregation to reduce overhead. We present the individual performance benefits obtained by each of the above directions. For routing, we use label based forwarding and adaptive path selection to support fast path switching, call admission and mobility support.

We provide a multihop aggregation mechanism that uses the “natural” waiting time of packets in a loaded network. We show that our aggregation scheme does not increase delay while providing significant benefit in term of capacity increase. Each of the optimization schemes proposed are distributed in nature and does not rise scalability problems for large mesh networks. The above performance optimization techniques are implemented in a 15 node indoor wireless mesh network. The experimental results show an increase of 13 times for a six hop string when all optimizations are used (Figure 18).

A. Related work

Recently, with growing importance of VoIP, several research works have addressed the performance issues of supporting VoIP over Internet. The use of switching among multiple paths to reduce delay was proposed in [3] and recovering from packet loss was proposed in [4]. These strategies were used for delivering VoIP using an overlay network. When transporting VoIP over the Internet, the major factor affecting the performance is path delay as for good quality, VoIP requires 200ms or less one way delay. In supporting VoIP over wireless network, the main factors affecting performance are the low capacity and the variable loss rate.

Some initial studies on the performance of real-time applications over 802.11 were presented by Sobrinho and Yeh in [5], [6]. References [7], [8] focused specifically on VoIP over 802.11 considering the delay and loss characteristics under PCF and DCF modes. Another recent work on VoIP over WLAN [9] presents analytical studies on the number of calls that can be supported in a single hop WLAN. The study reports that increasing the payload per frame increases the number of supported calls. Our work uses this observation for a multihop scenario to design the voice packet aggregation scheme.

Several performance optimization schemes were proposed for VoIP over WLAN: Yu et al. [10] propose the use of dual queue of 802.11 MAC to provide priority to VoIP, while Wang et al. [11] propose packet aggregation to increase capacity.

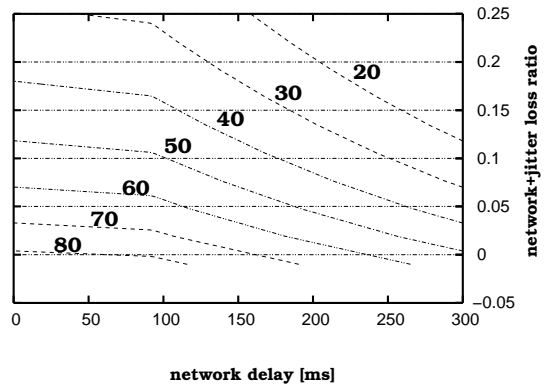


Fig. 2. *R-score* for 60ms jitter buffer.

Recently, research has been conducted in the area of 802.11 based wireless multihop mesh networks. A study conducted to understand the capacity of multihop network was presented in [2]. Research on improving the end-to-end performance of application on multihop network by employing multiple radios was considered in [12] and [13]. Further work on finding better routing metrics and strategies for multihop networks was presented in [14].

II. VOIP BASICS

A VoIP system consists of an encoder-decoder pair and an IP transport network. The choice of vocoder is important because it has to fit the particularities of the transport network (loss and delay). One of the popular voice encoders is G.729, which uses 10ms or 20ms frames. It is used by some available 802.11 VoIP phones (such as the Zyxel Prestige, Senao S7800H, but other wired VoIP phones as well). The Zyxel Prestige for example, sends 50 packets per second, of 20 bytes each, and we chose this traffic specification for experiments and simulations throughout this paper. Although a 30% utilization increase is generally expected when accounting for periods of silence when no packets are sent, we do not consider silence periods (the Zyxel phones and the Skype application also do not use silence suppression). To measure the quality of a call, we used a metric proposed in [15], which takes into account mouth to ear delay, loss rate, and the type of the encoder. Quality is defined by the *R-score*, which for medium quality should provide a value above 70:

$$\begin{aligned}
 R &= 94.2 - 0.024d \\
 &\quad - 0.11(d - 177.3)H(d - 177.3) \\
 &\quad - 11 - 40\log(1 + 10e)
 \end{aligned}$$

where:

- $d = 25 + d_{jitter_buffer} + d_{network}$ is the total ear to mouth delay comprising 25 ms vocoder delay, delay in the de-jitter buffer, and network delay
- $e = e_{network} + (1 - e_{network})e_{jitter}$ is the total loss including network and jitter losses

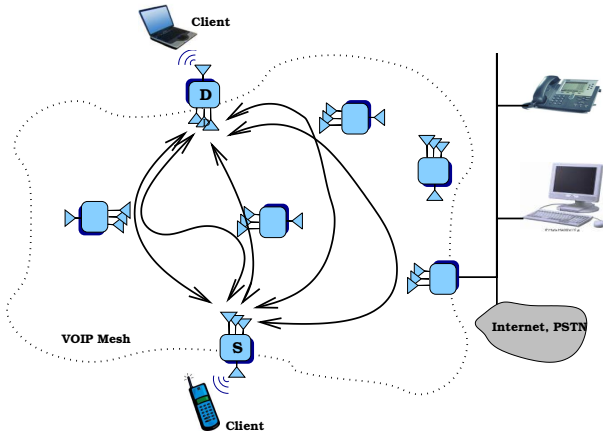


Fig. 3. Mesh system showing two clients connected, and the paths maintained between them. Each mesh node has one separate interface for the clients, in addition to backhaul interfaces. Clients can connect across the mesh to other wireless devices, in the institution intranet to wired VOIP phones, to the internet, or to the PSTN.

- $H(x) = 1$ if $x > 0$; 0 otherwise is the Heaviside function
- the parameters used are specific to the G.729a encoder with uniformly distributed loss

The constants consider the delay introduced by the encoder for its lookahead buffer, and the delay introduced by the jitter buffer. We considered a jitter buffer of 60ms, which has two contradictory effects: it increases end to end delay, therefore degrading the quality, but it also reduces the jitter, which has an overall better effect. The R -score is finally computed only from the loss and the delay in the network, which can be measured directly in our testbed. In order to emulate the behavior of a simple jitter buffer, we assume that playout starts at the destination after the arrival of 4 packets from start (60 ms jitter buffer = 3 packets). Therefore all the deadlines for the packets at the receiving side are established at this point. Loss in the jitter buffer is computed as the fraction of packets which do not meet their deadlines. In order to compute loss probabilities and average delay in the network, all packets from all flows in an experiment are considered together by macro-averaging.

Figure 2 shows the values of the R -score with respect to network delay and total loss for 60ms jitter buffer and 25ms vocoder delay. The interpretation of the iso- R -score curves is that for example to obtain an R -score of 70, the network has to deliver all packets in less than 160ms, or deliver 98% in less than 104ms. From the figure we can see that the quality is sensitive to even a couple of percents of loss, whereas the delay tolerates differences in tens of milliseconds. In 802.11, loss has a high variance, as it depends on the quality of the channels and the cards, and on the interference from external or internal sources. In a multihop setup, end to end loss is difficult to control and needs to be maintained under 2%. Using the retry mechanism of 802.11, this loss can be reduced at the cost of increasing delay.

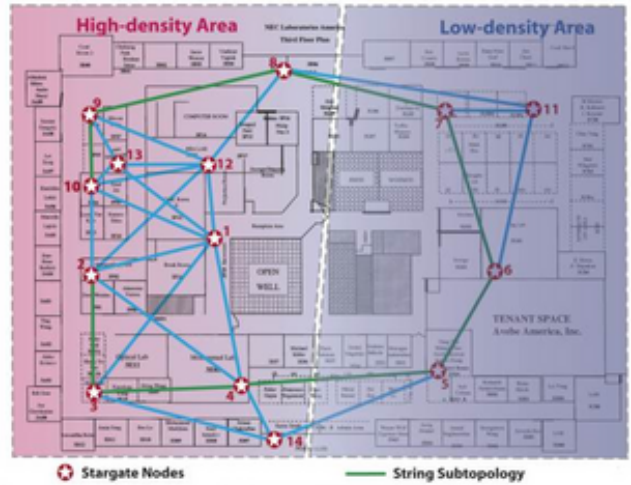


Fig. 4. 15 node testbed in a 70m x 55m building

III. VOIP MESH SYSTEM

To evaluate performance and capacity of voice in a mesh setup, we deployed a 802.11b based wireless mesh network for supporting VoIP traffic. In our implementation, we considered the mesh network as a multihop extension of the access point infrastructure existent in most institutions. It is useful to use the concept of a layer 2 switch to see the entire mesh as a single element that switches packets between its ports. A port is in fact a mesh node which has at least two interfaces: one in ad hoc mode for the backhaul in the mesh, and one in infrastructure mode to connect to clients (Figure 3). These clients can be VoIP wireless phones, laptops or other wireless handhelds. To the clients, the mesh acts as a switch or hub in the sense that they are not concerned with the internal routing of the mesh. However, the implementation of the mesh is based on IP, even though it offers a layer 2 abstraction outside. In our implementation, the clients can have connections across the mesh to other wireless devices as shown in the figure, through the institution intranet to other wired VoIP phones, out to the internet with the help of a SIP server, or to the PSTN through local PBX.

A. Hardware/software configuration

Our VoIP mesh testbed consists of 15 nodes based on the Stargate architecture from Intel, using the XScale processor, 32MB of RAM, and 64MB of compact flash. Each node is equipped with two 802.11b wireless interfaces (compact flash and USB 1.1) and has an open slot for a third one (PCMCIA 16bit). The testbed is spread over the third floor of NEC Research Labs in Princeton NJ, and the layout is shown in Figure 4. A high-density area on the left side of the building provides a proximity of nodes which allows the study of interference, and a lower density area to allow for longer paths in the network. The wireless cards are running at the fixed rate of 2Mbps which has the advantage of providing more stable results for the indoor setting. Each node operates with two

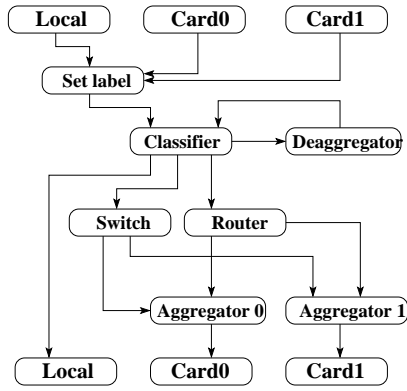


Fig. 5. Click node component for label based forwarding, routing , aggregation

interfaces: one that is used to get client traffic from the VoIP 802.11 phones, and the other one for backhaul in the mesh. If a third interface is available, it can be used to improve the capacity of the backhaul. To experiment with this setup using only two interfaces, we generated traffic locally at the nodes, in order to have both interfaces available for backhaul.

B. Mesh node

In order to provide routing, forwarding and other VoIP specific services, we used the Click modular router [16] on each mesh node. The router architecture is shown in figure 5. Voice packets use label based forwarding and routing, while other traffic uses regular routing. In this router configuration, when a packet is received from the cards, it may get labeled if it is a voice packet which needs to be routed over mesh network. For test purposes, labeling of traffic that is generated locally at the node is also allowed. To increase the capacity to carry VoIP traffic, we implemented a packet aggregation service which encapsulates multiple small VoIP packets into larger packets and forwards it. For each interface, a corresponding aggregator as shown in Figure 5 handles outgoing packets. Similarly, there is de-aggregator to decapsulate the aggregated packet into the original VoIP packets. A classifier decides whether a) packet is destined for the local machine (signaling, aggregated packets); b) has to be routed (best effort, signaling); c) has to be label routed (voice). For aggregated packets, after the decapsulation, the resulting packets are fed back to the classifier.

C. Label based forwarding

In order to label the IP packets, we use TOS field of each IP packet that provides 255 labels at each node. Packets with non zero label are forwarded based on their label. On the other hand, packets with label zero follow underlying routing protocol (DSDV) used in our mesh network. In order to perform label based forwarding, each node maintains an additional table with an entry like:

in_label	out_label	interface	gateway
----------	-----------	-----------	---------

Any packet that arrives on interface and has a non zero label (*in_label*) is stamped with the corresponding *out_label* and sent

to the *interface* to be delivered to *gateway* which is the next hop in the path. Once the outgoing label is set (by the Switch in Figure 5), and the outgoing interface is determined (for both routing and label based forwarding, by their respective lookups), packets are pushed to “pull” queues associated with each interface. These queues perform the aggregation of packets with have the same next hop (only voice and probe packets). The meaning of “pull” in Click terminology is that these queues are queried by the cards when the transmission is possible, so the waiting time in these queues is used for the purpose of the aggregation. A naive implementation of aggregation would delay small packets in order to club them together in larger packets. Using this implementation feature ensures that no forced delay is introduced during forwarding.

D. VoIP call routing

Voice packets have a hard deadline of about 200ms mouth to ear in order to achieve reasonable quality, while 400ms is acceptable for intercontinental calls. This means that for the wireless leg of the setup, there is a fairly tight budget in time, so it can not be relied on the routing protocol to reconfigure paths during the call, or even search for the path when the call is placed. Our design decision was to use pinned down paths using label based forwarding to service the voice calls, where paths are continuously refreshed in the background by some routing protocol.

Up to five pre-computed paths are maintained for all voice communications of a pair of nodes, obtained from a route discovery protocol such as DSDV[17] to update the list of paths in the background. The use of pre-computed label based paths is appropriate for mobility as well. While one hop handoff can be achieved in 60ms in 802.11 networks [18], updating a path in a mesh, or using triangle routing may not satisfy delay requirements. For example, a loaded 4 hop path in our testbed has a delay of 80ms, and still provides QoS, but a signaling scheme or routing protocol might require several round trips along this path to set up a route. With pre-computed paths, after the one hop handoff, the new node has five proven good paths to choose from.

The main reason to use label based pre-computed paths is *to have several alternative paths between the same source destination pair, available at all times*, but there are other advantages to be considered as well: a) no time to update paths during calls or after handoff, b) flows may have to be switched fast to alternative paths to maintain QoS, c) fast call admission (no waiting for routing setup or reservations); d) pinned down paths allow various path selection strategies at source (e.g. aggregation, interference). Multipath routing also comes with generic advantages which are not specific to realtime traffic, providing natural load balancing, increased resilience, path diversity, increased capacity and reduced interference (when several cards are available).

To summarize, the implemented VoIP mesh system uses routing (DSDV) for signaling and best effort traffic, and label based forwarding for voice traffic. Each node maintains statistics about the popular routes used by DSDV and keeps

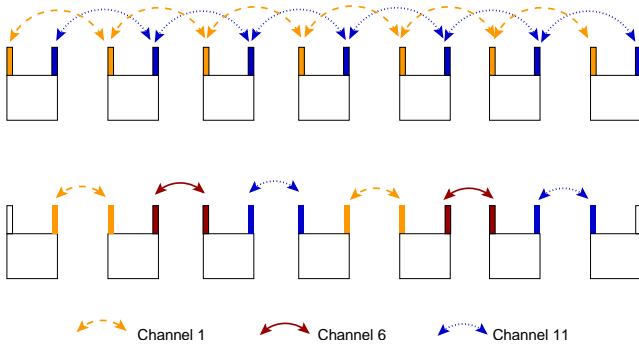


Fig. 6. Each node with two 802.11b interface. case A: two non overlapping channels for forward and reverse direction; case B: three channels used with reduced self interference

the top five most popular routes to each destination to be used for label based forwarding. Mobility is handled using these precomputed paths so that after handoff from one node to another, several paths are always available to continue the voice forwarding.

IV. VOIP PERFORMANCE OPTIMIZATIONS

A. Evaluation methodology

In order to evaluate the performance of VoIP communication over this network, we used the **rude** UDP traffic generator/collector[19], which is able to generate CBR packet flows with given rates, packet sizes, and schedules. Detailed traces of the connection include the sending time, receiving time, flow ID, and sequence number for each packet. In order to have an accurate measure of the delay, all nodes are synchronized using NTP. To emulate voice conversation between two terminals, we set up two simultaneous rude sessions, one for each speaker. All calls have one minute in length, and use a traffic pattern corresponding to the G729 encoder, which produces 50 packets per second with 20 bytes of payload each.

B. Use of multiple interfaces

Referring back to Figure 1, we see that the main problem in a multihop network is performance degradation with increasing number of hops. A simple idea for improvement would be to just increase the number of interfaces in each node. A naive use of multiple interfaces in a string would be to use one interface on a channel for the forward traffic and a second interface on a second channel for the reverse traffic, which should provide double capacity. We verified this in our testbed on a string of six hops. However, for each of these flows, the same behavior as in Figure 1 is created by interference with neighbors which have cards on the same channel. An alternate method is to use more independent channels as shown in Figure 6. However, using 802.11b, only three channels are available, which limits the achievable improvement. Operating with only two backhaul interfaces and only three independent channels offered by 802.11b, we evaluated the following situations. Case A: two independent

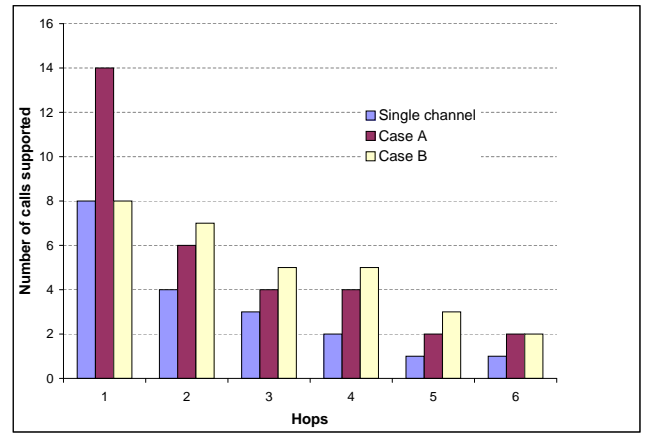


Fig. 7. Channel diversity: use of multiple cards with independent channels.

channels for forward and reverse traffic: (1,6)-(1,6)-(1,6)-(1,6)-(1,6)-(1,6). Case B: reduced self interference channel allocation: (1)-(1,6)-(6,11)-(11,1)-(1,6)-(6,11)-(11). The two solutions produce notable improvements, especially for longer paths (Figure 7). The lack of improvement for shorter paths is explained by a shortcoming of our testbed node, which only supports a limited number of interrupts per second. Using a better architecture, roughly a doubling of performance is expected with the addition of a second card, at least for the solution A. Solution B has even greater potential of improvement when more independent channels are available. If more channels were available, like in 802.11a, interference may be completely eliminated in a string, because a channel can be reused after 11 hops, which in most cases will be out of the interference range.

Further, one can use the multiple interfaces to create path diversity in addition to channel diversity. Here the forwarding path and the reverse path are disjoint, preferably at the interference level as well, except at the termination points. Each path independently also uses channel diversity.

To evaluate the multiple path option in the testbed, we pinned down two independent paths of five hops each, which are as far from each other as possible: 10-9-8-7-11-6 and 6-5-14-3-2-10 (see Figure 4 for the placement) and assigned channels for the forward and reverse paths: 1-7-1-7-1 forward and 4-11-4-11-4 reverse (case A), 1-1-7-7-1 forward and 11-11-4-4-11 reverse (case B). Using configuration B, five calls were possible between nodes 6 and 10, compared with just one call in the basic case (Figure 8).

Using several network interfaces provides scalability to the system, while using several channels across the mesh provides frequency diversity. These factors combined actually have a more than additive effect, meaning that the capacity improvement of using them together is greater than the sum of their independent improvements. The reason is the reduction of interference, but this gain is still limited by the number of interfaces used - 2, and the number of available independent channels - 3.

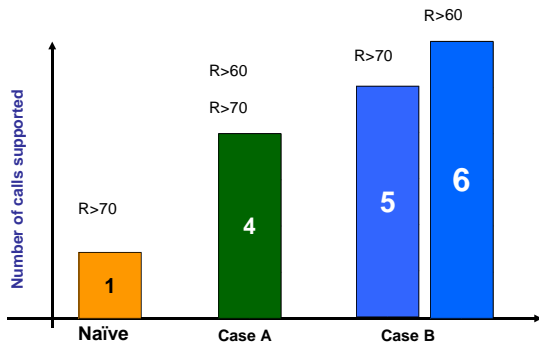


Fig. 8. Path and channel diversity: 2 disjoint paths, each using independent channels.

Currently, we are investigating the interference properties of 802.11a cards which offer better possibilities for denser meshes. On one hand, the range is shorter and the capacity higher, and on the other hand a larger number of channels is available, so a six hop setup like the one in our building should require no frequency reuse.

C. Routing

The design of good routing schemes for supporting real-time applications over wireless mesh is an inherently challenging problem. The difficulty in routing arises from the number of factors on which a good route depends: a) channel quality; b) dynamic condition due to interference caused by traffic inside and outside the mesh network; c) traffic load on routes in the interference range. Jointly considering the routing and channel assignment problem [12] is NP-hard even with a centralized solution and instantaneous global knowledge of network conditions. Voice calls pose additional problems because any decision taken in reaction to network conditions may affect voice quality: changes in routes, call admission and handoff, all have strict delay requirements to minimize the time during which packets are lost.

In our current setup, call admission has to be performed within seconds of placing a voice call. This is achieved by using pre-computed paths, even if the solution is suboptimal. Another important factor in this decision is that the call admission process should preferably be distributed. To achieve the above design goals, our voice call routing approach consist of two components: route discovery and adaptive path selection. To choose paths, we opted for a solution based on probing in order to cumulate all factors (interference, load, channel quality), which are otherwise hard to account for.

Route discovery: For route discovery, DSR would have been a good choice, but we need to maintain multiple source destination paths, and the implementation available with Click performs poorly on our platform in terms of CPU usage and responsiveness. DSDV is the second option, but being distance vector based, it has the undesirable effect of frequently updating paths in the middle of the call. We experimented with several metrics: hop count, end to end loss, quantized end to end loss, a threshold based loss metric, and ETX [20] (which

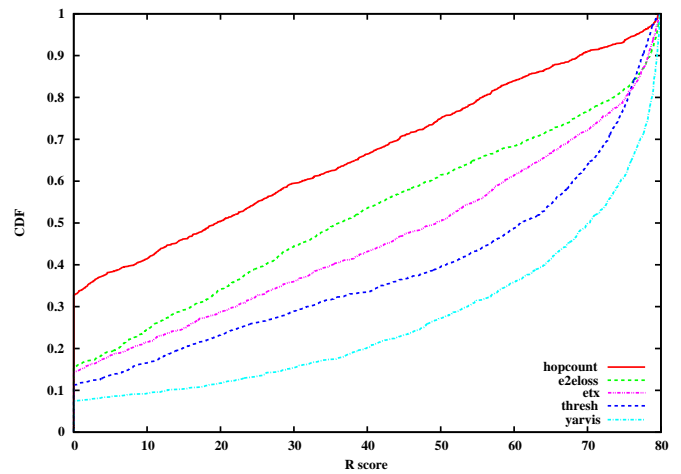


Fig. 9. DSDV has low performance for one call across the testbed using various metrics.

is also based on loss). All these metrics provide unacceptable performance (Figure 9) for voice in our testbed. This is mainly because of the default behavior of DSDV which is aging routes and is always ready to accept new ones. The wireless environment in the building is also a factor that degrades the performance of routing. Even the fairly stable hop count metric exhibits a lot of route variance because occasionally packets may travel across the entire building long enough to make the routing algorithm believe a one hop route is available. When this long link becomes unavailable, or is beyond a certain age, DSDV is looking for alternatives based on its current metric. It is in this case, during switching between routes, when voice packets get lost and quality gets degraded.

We opted instead for using DSDV to collect frequently used routes which are then pinned down and used with label based forwarding. Each node is maintaining the top five most frequently used routes for each destination, based on measured statistics. We ran DSDV with various loss based metrics (including ETX, end to end loss, quantized end to end loss). We retained the five most frequently used paths chosen during last 24 hours. These top five paths were almost the same for the different metrics, although with differing frequencies from one metric to another. This means that from the loss point of view, a similar set of path shows a consistently better quality over time.

Adaptive path selection: In order to use the paths for voice transport over our wireless mesh network, we pinned down the paths using label based forwarding as described in the last section. We chose a pair of nodes A and B at extremities of the building to maximize distance in hops and path diversity. To make use of the alternate paths and the possibility of fast switching without losing any packets we implemented a simple strategy in which one of the nodes monitors all the paths with a low bandwidth ping (one packet per second). When a call is placed, the five paths are probed with a low bandwidth probe to evaluate the delay of each path, which is the most critical component. The probing traffic has the

Path	R_{avg}	$cdf(R > 70)$	path usage	R_{avg} used
adaptive	71.2	0.86	-	-
a	40.4	0.48	47%	72.4
b	56.3	0.69	40%	73.2
c	17.1	0.19	11%	70.8
d	28.7	0.33	1%	5.1
e	6.5	0.07	1%	52.8

TABLE I
ADAPTIVE PATH SWITCHING VS. FIXED PATHS

same characteristics and treatment as the voice traffic, namely it can be aggregated or delayed based on localized conditions in the network. The size of the packet is chosen to be the same as the voice packet so that round trip times are good estimates that can be extrapolated for voice packets. When the exponentially averaged R -score of the voice call stays under 70 for an extended period, the decision is taken to switch to another path based on the monitored round trip times and loss rates.

Evaluation: In order to evaluate the above strategy on our mesh testbed, a single voice call is ran for 2500s between nodes A and B, and the R -score is recorded every second. Each of the five available paths is measured independently. To create a repeatable pattern of disruption similar to office use of laptops, we selected a number of jammers outside the testbed that follow a predefined random, but fixed, sequence of traffic on the same channel as the testbed.

In table I, we have the path labels in the first column, from a to e. The first line corresponds to the adaptive strategy. The second column of the table shows the average R -score achieved, and the third column the fraction of time when $R > 70$. By using the available alternate paths, the simple adaptive strategy is able to route the voice traffic around the interference and congestion providing a good R -score 86% of the time, with an overall average of $R = 71.2$. The fourth column shows how paths are used by the adaptive strategy, and the fifth column the average R -score obtained by the respective paths on behalf of the adaptive strategy. Most service is provided by just three paths, which could help in reducing the amount of probing traffic to only proven quality paths. Probing of additional paths may be enabled only when the reduced set doesn't provide the required quality.

Knowing that the R -score is a function of loss and delay, the question is which of the two factors is more important in our testbed. The network loss is less than half a percent for most paths, except one, so delay must be the deciding factor. In Figures 10a and 10b, the delay distribution histogram is shown for one of the participating paths and for the adaptive case. Times over 200ms are collapsed in the rightmost bin. These distributions confirm that the quality of our paths is dominated by delay. The source of this delay is cross traffic (from the jammers) and channel conditions (802.11 retry is set to 16). Since an unloaded path experiences about 2ms-3ms per hop, and our paths have 4 and 5 hops, the measured delay

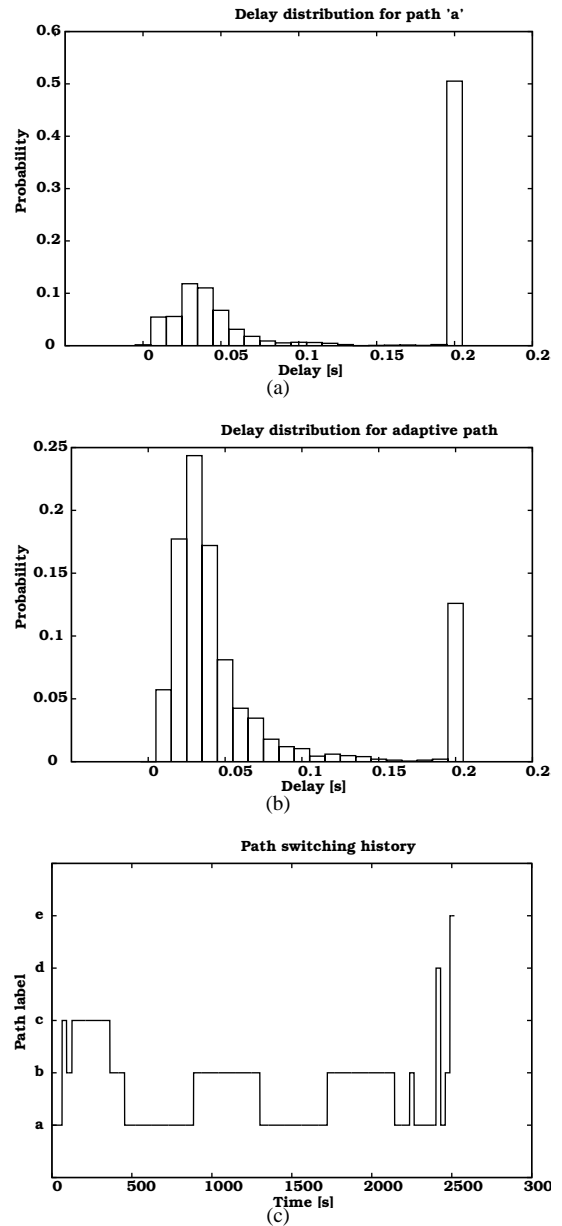


Fig. 10. Delay distribution adaptive vs. fixed. (a) A fixed path provide delays greater than 200ms 50% of the time. (b) Only 12% of the time the delay is greater than 200ms when the path is adaptive. (c) path labels used by the adaptive scheme.

was confirmed to be in the range 8-15 ms, which is almost negligible compared with the delays experienced by the voice traffic during the experiment. Figure 10c shows which paths were used during the experiment, corroborating the figures from table I.

D. Aggregation

As most vocoders use samples of 10-100 ms, a mesh node is expected to get a large volume of small packet traffic. However, 802.11 networks incur a high overhead to transfer one packet, therefore small sizes of packets reduce the network

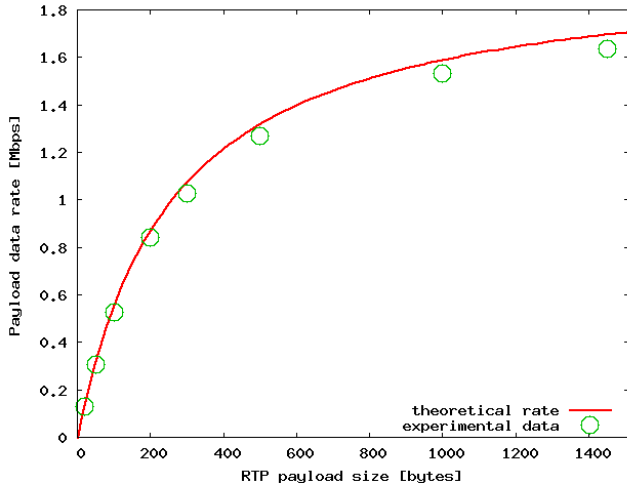


Fig. 11. Overhead measurement confirms analysis analysis in a 2Mbps 802.11 network.

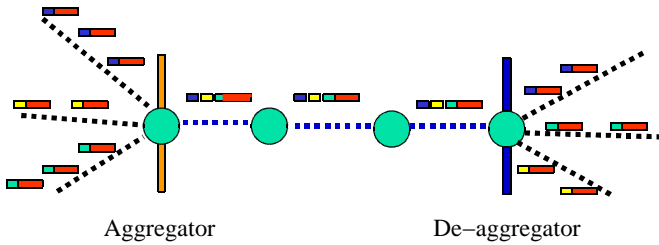


Fig. 12. Aggregation merges small voice packets from different calls into larger packets to improve channel utilization

utilization. The problem with small payloads is that most of the time spent by the 802.11 MAC is for sending headers and acknowledgments, waiting for separation DIFS and SIFS, and contending for the medium. For example, in order to send a 20 byte VoIP payload, a 60 byte packet is assembled from 20 bytes IP header, 12 bytes RTP header, and 8 bytes UDP header. This takes $43.6\mu s$ to send at 11Mbps, but MAC header and physical headers, trailers, inter-frame periods and ACK need a total of $444\mu s$. That however does not consider the amount of contention which is on average of $310\mu s$, and increases exponentially with contention. This way, to send a 20 bytes payload takes $800\mu s$ at 11Mbps, yielding approximately 1250 packets per second, which for a vocoder like G.729a means only 12 calls can be supported. At 2Mbps, a similar computation leads to 8 calls. When sending x byte voice samples, the overhead incurred is given by:

- RTP/UDP/IP $12+8+20=40$ bytes
- MAC header + ACK = 38 bytes
- MAC/PHY procedure overhead = $754\mu s$
 - DIFS($50\mu s$), SIFS($10\mu s$)
 - preamble + PLCP ($192\mu s$) for data and ACK
 - contention (approx $310\mu s$)

The throughput in Mbps is given by the relation

$$T(x) = \frac{8x}{754 + (78 + x)\frac{8}{B}}$$

where x is the payload size in bytes, and B is the raw bandwidth of the channel (1,2,5.5, or 11). In Figure 11, we measured actual data rate obtained between two nodes as a function of packet size employed, which proved to be fairly closed to the analysis of $T(x)$. When using 20 byte voice payload in a 2 Mbps network, the capacity of the network is only 10% of the maximum possible.

Two main techniques of reducing this overhead are packet aggregation and header compression. The basic idea of aggregation (Figure 12) is to combine together several small packets at the aggregator in the ingress nodes and forward them with one IP, MAC and PHY header across the air.

A common problem in packet aggregation is that it increases packet delay, reducing its suitability for VoIP services. But if the network is lightly loaded, the packet aggregation techniques do not have to be used for improving network performance. Under heavy load, small size packets would experience heavy contention which lead to retransmission, and drops. The packets then spend the largest part of network delay in the queues at the intermediate nodes. The higher the contention to access wireless media, the larger the network delay becomes. These small packets waiting for media access in the queues are the candidates for packet aggregation. Therefore, packet aggregation in the heavy load does not need to introduce additional forced delay to combine packets.

Packet aggregation can also be used for combining voice packets in the same flow by introducing an explicit delay at the ingress node of the call, if the extra latency does not degrade voice quality severely. This technique increases mouth to ear latency, but it also reduces queueing delay caused by network contention. On the other hand, too large forced waiting at the ingress node causes longer network delay, and lower quality. In our system, the delay budget available for aggregation is obtained from path monitoring. If measured delay on a path allows extra waiting without degrading the R -score, that amount is allocated to aggregator at the ingress node.

The aggregator treats differently flows which are forwarded and the ones for which it is the ingress node. At the ingress the packets can be delayed some amount of time, depending on the budget allowed by the probing of available paths (Algorithm 1). When forwarding however, no delay is introduced, but under higher than minimal load, packets still cumulate in the queues, waiting for medium access. During this wait, other packets may join the aggregated packet, provided that they have the same next hop or same destination. Alternatively, packets which take a different path are split and re-aggregated accordingly.

Evaluation: In $ns-2$, we simulated a string of 6 nodes with and without aggregation, and verified the results against a similar string in the testbed. In Figure 13, we find that for the non aggregated traffic, the simulation matches the testbed results in most points, but for aggregated traffic the testbed

Algorithm 1 Aggregation logic for ingress nodes

P - packet being queued at a node;
P' - packet with the same next hop as P;
A - aggregation packet being prepared;
minPackets - number of packets from the same flow that have to be aggregated at the ingress (corresponds to the delay budget available for the flow);
MTU - maximum transmission unit = number of voice packets that can be fit in 1500 bytes;

```
find queue of P;  
1: if size(queue) > minPackets  
    add all packets from flow(P);  
    if size(A) < MTU  
        find a queue with the same dest  
        go to 1;  
    else  
        send A directly to destination;  
else  
if size(A) < MTU

---


```

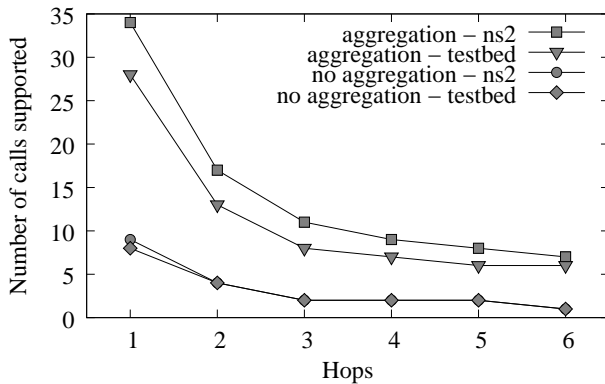


Fig. 13. Aggregation on a string: ns-2 vs. testbed.

performs worse. The cause of this was identified in the fact that the capacity of some of the hops in 2Mbps mode was less than the optimal 1.7Mbps. The first hop for example was measured to provide a capacity of only 1.38Mbps, with 1500 bytes packets, accounting for the difference between the aggregated calls supported - 29 in testbed versus 34 in *ns-2*. However, knowing that our simulation setup performs reasonably close to the testbed, we obtain the most of the next results in simulation only.

One of the main claims of our distributed aggregation method is that it does not introduce additional delay by using the wait for the MAC availability to club together packets destined to the same next hop. To verify this claim, we place five longer calls indicated by A in Figure 14 and five short calls, indicated by B. Aggregation is performed for each group of flows independently at the source by introducing a

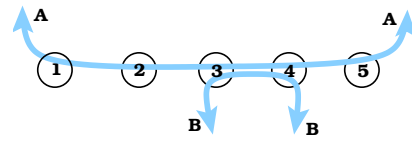


Fig. 14. Aggregation introduces only controlled delay at the source of flows. Intermediate nodes do not delay packets to improve aggregation, but use “natural” waiting required by MAC under load.

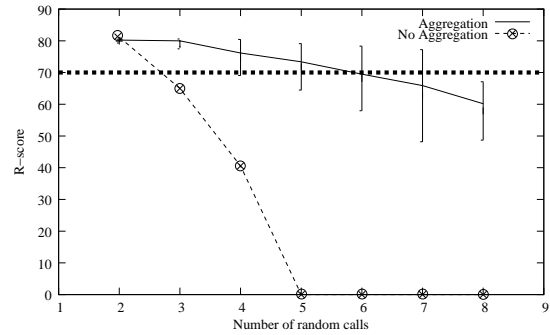


Fig. 15. Aggregation performance for random calls in a string

controlled delay of 80ms. This means that packets from A are not merged with packets from B during their common hop 3-4. The network time for flows A is about 61ms in the absence of B and increases to 89ms after B is added. This is normal, considering the increased interference for all the nodes and the longer queues at nodes 3 and 4. After we enable aggregation between A and B at hop 3-4 the network time for flows A decreases to 79ms. Not only a delay is not added to the long flow, but the creation of larger packets reduces the contention for the hop 3-4 thus reducing the load on the network. This experiment reveals some interesting properties of our aggregation scheme: first, it only kicks in at higher load when waiting in the queues can be used to group packets with the same next hop. Second, it is completely distributed inside the mesh. The endpoints need to specify initial delays depending on their time budgets, but the intermediate nodes have the simpler task of looking for packets with the same next hop. Third, and the most important, **short flows do not delay long flows for the purpose of aggregation**. It is true that the mere existence of short flows increases wireless medium load and therefore increases delay for long flows, but that is inherent to the behavior of the shared medium. In network distributed aggregation reduces load without impacting the network time of existing flows.

In another experiment we consider a more randomized situation in terms of sources and destinations of calls. In an eight hop string, calls between random sources and random destinations are placed. Considering 10 random situations for each configuration of two to eight calls, we compute the minimum, maximum, and average value of the *R-score* for the offered number of calls. In Figure 15 these values are plotted for each offered load. For the non-aggregation case

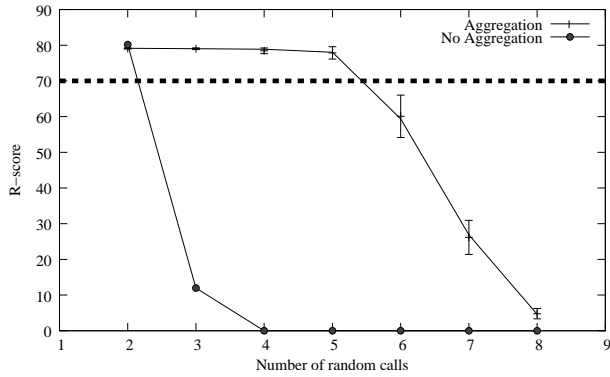


Fig. 16. Aggregation performance for random calls in a tree

minimum and maximum only take values 0 or maximum, and are omitted from the figure. If we consider an *R-score* of 70 as a threshold of call quality, then aggregation more than doubles the capacity, even with randomized traffic.

When the mesh is used as an extension of the access point infrastructure, a popular pattern is to have voice calls forwarded to the wired infrastructure or to the PSTN. In this case paths from the clients all lead to a common root which provides access to the wired leg of VOIP. We simulated a complete binary tree with 8 leaves, and with an additional link from the root to the wired access, so that there are 4 hops to forward from the leaves. Figure 16 shows that performance improvement is similar to the string case, by more than a factor of two. For the non aggregated case however, the capacity is much less, mainly because of the interference that now covers larger portions of the tree (more than the 4 hops that interfere in a string).

E. Aggregation and header compression

Header compression is a complementary scheme related to aggregation. It has the same goal of reducing the amount of overhead by exploiting headers that do not change, or whose change can be predicted. For a VoIP flow RTP/UDP/IP headers take 40 bytes, but only 12 of them are changing often. Schemes such as cRTP or ROHC aim at compressing the 40 bytes into a 2 byte connection ID, but they are appropriate for one link only. In order to emulate a simpler scheme that only transmits the changing fields, we reduce the header from 40 to 14 bytes so that more voice packets can fit in an 1500 byte packet.

In Figure 17, we look at the performance gain given by header compression alone, aggregation alone and the combination of header compression and aggregation. Header compression by itself achieves an almost negligible improvement, because it only reduces overhead with 26 bytes out of the total 78 bytes + 754 μ s. Aggregation only curve is the same as in Figure 13, repeated here for reference. When combining header compression and aggregation we get another factor of two improvement in capacity. The reason is that once the aggregation reduces the MAC overhead of 78 bytes + 754 μ s, the saving of 26 bytes provided by header compression becomes significant by allowing more voice samples be stored

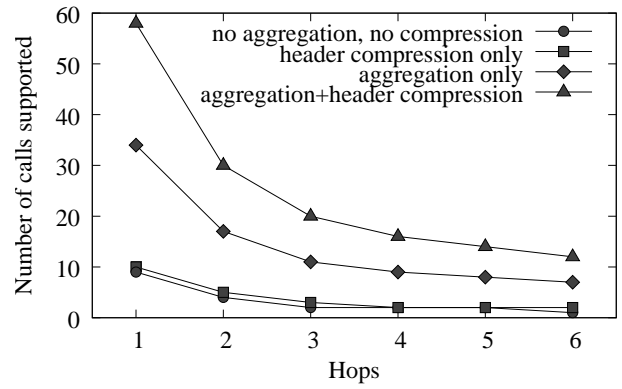


Fig. 17. Header compression increases capacity over simple aggregation.

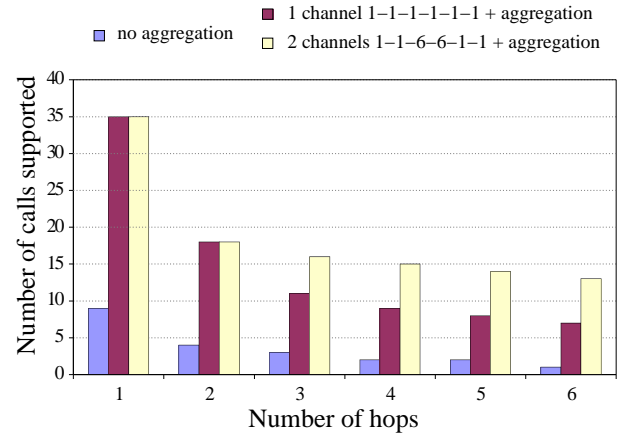


Fig. 18. To send a 20 byte packet over 802.11, 78 bytes are used by MAC, IP, UDP and RTP headers. Aggregating voice packets from different flows provides 13 times improvement for 6 hop calls.

in a 1500 byte wireless packet - 41 for header compression versus 24 for aggregation only.

F. Aggregation and multiple interfaces

To combine the advantages of using several interfaces and aggregation, we simulated in *ns-2* a string of 6 nodes that use one or two channels for the backhaul traffic. The improvement is most visible at 6 hops (Figure 18): a factor of 7 increase from the aggregation, and another factor of 2 from the multiple channel.

The combined results of the testbed experiments and the *ns-2* simulations show that with appropriate optimizations, the wireless mesh is appropriate for sending voice. Path adaptation enabled by label based forwarding is improving the QoS, while channel diversity, path diversity and aggregation improve capacity.

V. CONCLUSIONS

We experimentally investigated several methods to improve the quality of VoIP over a WLAN mesh. These are the use of multiple interfaces, label based forwarding architecture, and packet aggregation. Each of these methods produces

considerable improvement in the operation of the mesh - with respect to capacity, QoS, or both. After evaluating several design options we believe that a label based solution is the most appropriate for carrying realtime traffic in a wireless mesh operating in the unlicensed spectrum. Our architecture combines routing and label based forwarding, and addresses all aspects required to support VoIP over the WLAN mesh: call admission, mobility, QoS. We implemented a distributed packet aggregation strategy that is work conserving by using MAC waiting to perform aggregation, without introducing unbounded packet delays. These performance optimizations are implemented in a 15 node wireless mesh network, and the experimental results show an increase of 13 times for a 6 hop string when all optimizations are used.

REFERENCES

- [1] *Skype*. <http://www.skype.net>.
- [2] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *ACM MobiCom*, (Rome, Italy), pp. 61–69, July 2001.
- [3] S. Tao, K. Xu, A. Estepa, T. Fei, L. Gao, R. Guerin, J. Kurose, and D. Towsley, "Improving VoIP quality through path switching," in *In Proc. of IEEE INFOCOM*, March 2004.
- [4] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "1-800-overlays: Using overlay networks to improve VoIP quality," in *In Proc. of NOSSDAV*, 2005.
- [5] J. L. Sobrinho and A. Krishnakumar, "Real-time traffic over the IEEE 802.11 MAC protocol," in *In Bell Labs Technical Journal*, Autumn 1996.
- [6] J.-Y. Yeh and C. Chen, "Support of multimedia services with the IEEE 802.11 MAC protocol," in *In Proceedings of IEEE ICC*, 2002.
- [7] M. Veeraraghavan, N. Cocker, and T. Moors, "Support of voice services in IEEE 802.11 wireless LANs," in *In Proceedings of IEEE INFOCOM*, 2001.
- [8] D. Hole and F. Tobagi, "Capacity of an IEEE 802.11b wireless LAN supporting VoIP," in *In Proceedings of IEEE ICC*, 2004.
- [9] S. Garg and M. Kappes, "Can I add a VoIP call?," in *IEEE ICC*, (Anchorage, Alaska), 2003.
- [10] J. Yu, S. Choi, and J. Lee, "Enhancement of VoIP over IEEE 802.11 WLAN via dual queue strategy," in *In Proceedings of IEEE ICC*, 2004.
- [11] W. Wang, S. C. Liew, and V. O. K. Li, "Solutions to performance problems in VoIP over 802.11 wireless LAN," in *IEEE Transactions on Vehicular Technology*, 2005.
- [12] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," in *ACM Mobile Computing and Communications Review(MC2R)*, vol. 8, April 2004.
- [13] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *ACM MobiCom*, (San Diego, CA), September 2003.
- [14] J. P. R. Draves and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *In Proceedings of ACM SIGCOMM*, 2004.
- [15] R. G. Cole and J. H. Rosenbluth, "Voice over IP performance monitoring," *ACM Computer Communication Review*, vol. 31, April 2001.
- [16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, pp. 263–297, August 2000.
- [17] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, (New York, NY, USA), pp. 234–244, ACM Press, 1994.
- [18] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 93–102, 2003.
- [19] *Rude UDP packet generator*. <http://rude.sourceforge.net/>.
- [20] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *9th ACM MobiCom*, (San Diego, CA), September 2003.