Sync & Sense Enabled Adaptive Packetization VoIP

by

Boonchai Ngamwongwattana

B.Eng., King Mongkut's Institute of Technology, Ladkrabang, Thailand, 1994

M.S., Telecommunications, University of Pittsburgh, 2001

Submitted to Faculty of Telecommunications Program,
Graduate School of Information Sciences, University of Pittsburgh
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

University of Pittsburgh

2007

University of Pittsburgh
School of Information Sciences
Department of Information Sciences and Telecommunications

This dissertation was presented
by
Boonchai Ngamwongwattana

It was defended on
April 6, 2007

and approved by

Dr. Richard Thompson (Advisor)

Dr. David Tipper

Dr. Martin Weiss

Dr. Joseph Kabara

Dr. Stephen Walters

Sync & Sense Enabled Adaptive Packetization VoIP

Boonchai Ngamwongwattana, Ph.D.

University of Pittsburgh, 2007

The quality and reliability problem of VoIP comes from the fact that there is a mismatch between VoIP and the network. Namely, VoIP has a strict requirement of bandwidth, delay, and loss, but the network cannot guarantee such a requirement. A solution is to enhance VoIP with an adaptive-rate control, called adaptive-rate VoIP. Adaptive-rate VoIP has the ability to detect the state of the network and adjust the transmission accordingly. Therefore, it gives VoIP the intelligence to optimize its performance, and making it resilient and robust to the service offered by the network. The objective of this dissertation is to develop an adaptive-rate VoIP system, which is composed of three components: rate adaptation, network state detection, and adaptive-rate control. In the rate adaptation component, we study optimizing packetization for rate adaptation. The advantage is that rate adaptation is independent of the speech coder. The study shows that the VoIP performance is primarily affected by three factors: packetization, network load, and significance of VoIP traffic; and, optimizing packetization allows us to ensure the highest possible performance. In the network state detection component, we propose a novel measurement methodology called Sync & Sense. Sync & Sense is unique in that it can virtually synchronize the transmission and reception timing of the VoIP session without requiring a synchronized clock. The simulation result shows that Sync & Sense can accurately measure one-way network delay. Other benefits include the ability to estimate the available bandwidth and the full spectrum of the delays of the VoIP session. In the adaptive-rate control component, we consider the design choices and develop an adaptive-rate control that makes use of the first two components. The integration of the three components is a novel and unique adaptive-rate VoIP called Sync & Sense Enabled Adaptive Packetization VoIP. The simulation result shows that our adaptive VoIP can optimize the performance under any given network condition, and deliver a better performance than traditional VoIP. The simulation result also demonstrates that our adaptive VoIP possesses the desirable properties, which include fast response, aggressiveness, TCP-friendliness, and fair bandwidth allocation.

Acknowledgements

I would like to acknowledge many people for helping me during my doctoral work. First of all, I would like to thank my advisor, Dr. Richard Thompson. Throughout my doctoral work, he offered me generous guidance, time and support. I am grateful and greatly appreciate all the help. Many thanks go to the committee members: Dr. David Tipper, Dr. Martin Weiss, Dr. Joseph Kabara, and Dr. Stephen Walters. Their comments allowed me to improve this dissertation and make it even better. I would also like to thank the Telecommunications program, School of Information Sciences, at the University of Pittsburgh, for a wonderful study experience. I wish to thank all of my friends I met in Pittsburgh. I always remember the good times we had. Finally, I am grateful to my parents for their love and support. I am especially grateful to my mother and my wife for their patience, love, care, and support, and for believing in me. I would like to dedicate this dissertation to my daughter and son.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Voice over Internet Protocol, also called VoIP, IP telephony, and Internet telephony is one of the fastest-growing areas in communications today. VoIP allows the routing of voice conversations over packet-switched networks, including the Internet. Given the ubiquitous presence of IP, VoIP has brought tremendous attention and opportunities. The benefits of VoIP are numerous and can go beyond free long distance calls. The ability to carry traditional telephone traffic, in addition to data traffic, on a single network allows more efficient use of the IP network. This consolidation in turn allows the reduction of the overall network cost, which includes infrastructures, administration, skilled personnel, etc. VoIP can facilitate tasks that may be difficult to accomplish using the traditional telephone networks. VoIP can provide mobility that allows users to make a call anywhere as long as a network is available. VoIP can integrate with other services available on the Internet that brings new kinds of services. Click-and-talk is an example, which allows a website user to click a button and immediately speak with a customer service representative. Another example is unified communication services that allow users to communicate using data, voice, and video within the same platform or device.

Whereas the concept of VoIP itself is brilliant, the challenge of VoIP is due to the fact that it depends upon the network connection. The quality and reliability of the VoIP call relies entirely on the quality-of-service provided by the network. Packet-switched networks are well known to have a bursty traffic pattern, in which quality-of-service cannot be guaranteed. Variable and high network delays and excessive packet loss can significantly reduce the voice quality and cause problems that distract the call conversation. Besides the quality and reliability, other major challenges of VoIP include interoperability and security. Interoperability involves standardization of the issues within the industry, which can ease,

encourage, and accelerate the adoption of VoIP. The two major competing standards for VoIP are the ITU standard's H.323 and the IETF standard's SIP. Because the Internet is an integral part of a VoIP system, VoIP is subject to all of the security risks that affect data networks. The main security issues are authenticity, privacy, and availability. VoIP security is an emerging issue. There is an increasing awareness of the potential security problems, as well as initiatives to improve VoIP security. Although these challenges have slowed VoIP progress, they cannot stop the growth of VoIP. The benefits of VoIP are tremendous and appear to overshadow the challenges. VoIP has come a long way and is continuing to improve in all aspects.

1.1 Challenge in Achieving Quality of Service for VoIP

The quality and reliability problem of VoIP comes from the fact that VoIP relies on the network, particularly packet-switched networks, to transport the voice packets. Thus, the problem lies on the quality-of-service provided by the network. A high network delay and excessive packet loss can easily reduce the quality of VoIP. Whereas packet-switched network was originally designed for data traffic, it offers best-effort service in which quality-of-service cannot be guaranteed. Specifically, network delay, packet loss, and available bandwidth can be variable, unpredictable, and unbounded. The best-effort service model works best for data traffic, but not VoIP traffic. VoIP has specific and somewhat strict requirements that must be met; namely, a fixed amount of bandwidth, a low packet delay, and minimal packet loss. The fundamental mismatch between the needs of VoIP and the service provided by the network is indeed the inherent problem of VoIP. Researchers have spent a great deal of effort attempting to overcome this challenge. The solution can be classified into two broad categories: network approach and endpoint approach.

The network approach aims at improving the network so that it can support the need of VoIP. This can be done by adding some kind of quality-of-service mechanisms to the network. Integrated Services [1], for example, is an architecture that specifies the elements to guarantee quality-of-service. It allows each individual application that needs some kind of guarantees to make a reservation of the requirements. As opposed to Integrated Services that provides a fine-grained quality-of-service system, Differentiated Services is an alternative

that offers a coarse-grained control system. Differentiated Services [2] operate on the principle of traffic classification, where each packet is placed into a limited number of traffic classes. This allows the network to offer preferential treatment for higher-priority traffic. Overprovisioning network capacity [3] also falls into this category. Although it may not provide any specific mechanism, it simply makes network congestion a rare event. Thus, overprovisioning can implicitly improve quality-of-service. Nonetheless, the network approach has challenges of its own. Deployment in a large scale is typically difficult. It also requires tremendous collaboration among the Internet service providers.

As a matter of fact, it is almost impossible to rely on the network approach because such a quality-of-service guaranteed network may not be available. The bottom line is that VoIP still has to operate on best-effort service networks. The endpoint approach is a necessity as it aims at improving the endpoint to be resilient and robust to the service offered by the network. For example, low bitrate speech coders and voice activation detection help to reduce the bandwidth requirement, which could potentially minimize network congestion. A jitter buffer allows VoIP to deal with variable and unpredictable network delays. Enhancing VoIP with an adaptive-rate control allows the endpoint to be network-aware and intelligent enough to optimize its performance. However, the endpoint approach has challenges of its own as well. Its effectiveness may be limited if the network exhibits large delays and excessive packet loss. Due to the fact that VoIP traffic is relatively small, the solutions may not be able to make a significant impact to the performance gain. The solutions can at least allow the endpoint to mitigate the problem and improve a certain degree of performance. An important advantage of the endpoint approach is its independence from the network. Given the limitations of both the network and endpoint approaches, no single solution can practically achieve the quality-of-service goal. A combination of the solutions offers a great potential to achieve quality-of-service for VoIP.

1.2 Problem Statement and Research Objectives

This research addresses the problem of the fundamental mismatch between VoIP and the network. Whereas VoIP has strict requirements for bandwidth, delay, and loss, the network (namely, best-effort service networks) cannot guarantee to provide such

requirements. VoIP typically uses UDP as its transport protocol, with no control mechanism. VoIP simply transmits packets at a constant rate regardless of the state of the network. Such an implementation cannot deal with varying network condition. In addition, network congestion can further degrade the performance of VoIP. We focus on the endpoint approach by enhancing VoIP with an adaptive-rate control, called adaptive-rate VoIP. Adaptive-rate VoIP has the ability to detect the state of the network and adjust the transmission accordingly. This solution gives VoIP the intelligence to optimize its performance, making it resilient and robust to the service offered by the network. An advantage of this solution is that it follows the end-to-end principle [4], which is implicitly enforced by the Internet. Whereas the best-effort network is relatively dumb, the end-to-end principle implies that the endpoints must be network-aware and intelligent. Hence, adaptive-rate VoIP is well suited the current Internet model.

Research in the area of adaptive-rate VoIP is somewhat in its infancy. Only a limited number of studies are available [5, 6, 20, 21, 22, 35]. Most of the studies focus on a specific element in developing an adaptive-rate VoIP system. For example, Qiao et al [5] develop an objective measure of perceived speech quality to be used with an AIMD-based control mechanism. Beritelli et al [6] study an adaptive-rate VoIP system that is based on TCP-friendly algorithms. More related research is discussed in the later chapters. Here, we take a comprehensive approach in studying and developing an adaptive-rate VoIP system. Adaptive-rate VoIP is generally composed of three components: rate adaptation, network state detection, and adaptive-rate control. We carefully look at each component, identify the problems, and find the solution to overcome them.

Rate adaptation is a fundamental basis for adaptive-rate VoIP. In fact, the idea of enhancing real-time applications with an adaptive-rate control has been around for many years. Though, it has been largely studied in the context of video, not VoIP. Several factors are believed to limit initiatives for adaptive-rate VoIP. An important factor is that variable bitrate speech coders were virtually non-existent in the past. Recent development of variable bitrate coders, such the GSM Adaptive Multi-Rate (AMR) coder [7], enables VoIP to perform rate adaptation, which results in more studies about adaptive-rate VoIP. A problem of using variable bitrate coders is that they typically operate by trading off quality for lower bitrate. As a consequence, the change in voice quality due to rate adaptation could distract

the user. In the rate adaptation component, we propose an alternative of using packetization as a means for rate adaptation. This means rate adaptation can be done by using any constant bitrate speech coder. Because the output bitrate from the coder remains the same, there is no impact on the audio quality of the produced voice frames.

Network state detection is a crucial component of adaptive-rate VoIP because the control needs to know the state of the network before making a decision. The perceived quality of VoIP depends upon both end-to-end delay and packet loss. End-to-end delay affects conversational interactivity and echo, and packet loss affects clarity. Detecting the state of the network for VoIP must serve the objective of obtaining a measure of both the delay and loss. This task, however, cannot be accomplished by the standard RTP (Real-Time Transport Protocol) and its associated RTCP (Real-Time Transport Control Protocol). This is because RTP is a protocol framework that is deliberately not complete and only provides functions expected to be common for real-time applications [8]. As the matter of fact, the application must have its own mechanisms for functions not provided by RTP, because the application has the best knowledge of its data and control decision. For adaptive-rate VoIP, obtaining the one-way network delay is a daunting task. The challenge is that the endpoints typically operate independently and without support from the network. Without a synchronized clock, it is almost impossible to measure the one-way delay. In the network state detection component, we propose a novel measurement methodology called Sync & Sense of periodic stream. Sync & Sense has the ability to virtually synchronize the transmission and reception timings, which enables it to obtain the full spectrum of the delays of the VoIP session. In addition, Sync & Sense can estimate the available network bandwidth. Sync & Sense truly serve the objective of detecting the state of the network for VoIP, and provide a wider range of indications about the network condition.

Adaptive-rate control is at the center of adaptive-rate VoIP. It makes a decision based on information as provided by the network state detection component. Most of the earlier works in adaptive-rate VoIP rely on packet loss to detect the state of the network because they lack a method to measure the one-way network delay. Sync & Sense allows us to overcome this barrier. In the adaptive-rate control component, we design the control that makes use of the proposed components of rate adaptation and network state detection. The integration of the three components is a novel and unique adaptive-rate VoIP called Sync &

Sense Enabled Adaptive Packetization VoIP. Sync & Sense enables the adaptive-rate control to make a decision based on both delay and loss. Observing the trend of one-way network delay allows our adaptive VoIP to gain more insight about the network state and better respond, in order to optimize the performance. While any proposed adaptive-rate VoIP must demonstrate that it can deliver a performance improvement over traditional VoIP, other important issues have barely been explored. A key objective in our design is that the adaptive VoIP must possess the desirable properties. The adaptive VoIP must be quick to adjust the transmission to the changing network condition. The adaptive VoIP must have the adequate aggressiveness to compete with TCP for its needed share of bandwidth. At the same time, the adaptive VoIP must be responsive to network congestion and demonstrate a degree of TCP-friendliness. In a homogeneous adaptive VoIP network, the adaptive VoIP must be able to provide a fair bandwidth allocation to all the competing flows.

1.3 Dissertation Outline

This dissertation is organized following the three components that comprise an adaptive-rate VoIP system. Chapter 2 addresses the rate adaptation component, where we study optimizing packetization for rate adaptation. Chapter 3 addresses the network state detection component, where we propose a novel measurement methodology called Sync & Sense of periodic stream. Chapter 4 addresses the adaptive-rate control component, where we design a control based on the two proposed components of rate adaptation and network state detection. In addition, chapter 4 includes the integration of the three proposed components that results in a novel adaptive-rate VoIP called Sync & Sense Enabled Adaptive Packetization VoIP. Each chapter begins with its own background and related research in which problems and issues are discussed. Then, the proposed research work is presented. This dissertation is especially organized in such a way that each chapter is complete on its own and can be read individually or as part of the overall dissertation. This is because the proposed work in each component is unique in itself. For instance, the knowledge and findings about optimizing packetization can be applied in any adaptive-rate VoIP system. Sync & Sense is not restricted to be used for adaptive-rate VoIP. It can be use for other purposes, for example, to monitor and collect statistics of the routes of the VoIP sessions.

Chapter 2

Optimizing Packetization for Improving VoIP Performance

A problem of VoIP is that its performance is usually unpredictable and could be degraded at any time. This is because packet-switched networks, including the Internet, never guarantee the requirement of any traffic. On the contrary, VoIP has several specific requirements of available bandwidth, delay, and loss. The solution to this problem is that VoIP should be more flexible in its requirements. Adaptive-rate VoIP is an approach that allows the bandwidth requirement to be variable, instead of fixed. Adaptive-rate VoIP attempts to adjust the transmission rate to match the available network bandwidth. This can help to minimize network congestion, which in turn can lower the delay and reduce packet loss. The challenge of this approach is how to make VoIP to be able to transmit packets at variable rates. Most speech coders are typically constant bitrate. Variable bitrate speech coders were non-existent in the past, until recently. This is believed to be a reason that limited initiatives in research about adaptive-rate VoIP. Many of adaptive-rate VoIP systems that have been proposed recently are based on variable bitrate speech coders.

Here, we propose an alternative, based on constant bitrate coders, that uses packetization as a means to vary the bandwidth requirement of VoIP. We explore how packetization can change the transmission rate. We then study the effect of packetization on VoIP performance. Through simulation-based study, we found that optimizing packetization can help to improve the delay and loss performance. In addition, cross traffic and significance of VoIP traffic play an important role in determining the performance. Under different levels of network load, there is an optimal packetization that can minimize the end-to-end delay. The findings from the study provide insights about when and how adaptive-rate

VoIP can improve the performance. This study also demonstrates the feasibility of adaptive-rate VoIP based on packetization.

2.1 Background and Related Research

As is well known, a packet-switched network service does not guarantee available bandwidth, and delay and loss bounds. At any time, it is possible that network congestion can cause large delay and excessive packet loss. The TCP protocol has a congestion control mechanism that backs off in the event of packet loss, ensuring that congestion can be resolved. Real-time applications, including VoIP, on the other hand, usually send packets at a constant rate without a control mechanism. The inherent problem is that they cannot react to network congestion. As a consequence, the performance is degraded when the requirement is not met. The idea of enhancing real-time applications with an adaptive-rate control has been around for a decade. However, the research in this area has been largely focused in the context of video, not VoIP. This is due to several reasons. Usually, there is more room to play with video because the bandwidth requirement is relatively large. Many adaptive-rate control schemes for video are well-known, for example, RAP [9] and TFRC [10]. Some argue that VoIP traffic will be a tiny percentage of all traffic on a transmission link since VoIP traffic is orders of magnitude less than video traffic. Another reason is that variable bitrate speech coders were virtually non-existent in the past. Most speech coders are generally model-based, which send parameters representing the model, independent of network condition. Some have proposed the use of banks of speech coders, each with a different bitrate, and then switching among them to perform adaptive-rate control [11]. This approach has some drawbacks. One is the problem of implementing many speech coders in the same platform. Another, the transition from one coder to the other might not be transparent to the user and could cause some distraction. Variable bitrate speech coders have been developed only recently. A well-known example is the GSM Adaptive Multi-Rate (AMR) coder [7]. The AMR coder uses eight different bitrates: 4.75, 5.15, 5.90, 6.70, 7.40, 7.95, 10.2 and 12.2 Kbps. The voice quality (e.g. MOS) varies depending on the bitrate. The higher the bitrate, the better is the voice quality. The AMR coder is expected to be used in many applications such as IP telephony and wireless systems. It was devised by ETSI for the third-generation mobile system. Optimizing packetization in order to improve VoIP

performance has been studied in [16]. Here, we further extend the study to examine the limitation of such an approach, as well as the feasibility of using such an approach for adaptive-rate VoIP.

2.2 Variable Rate VoIP Based on Packetization

Packetization is a parameter that must be specified when setting up VoIP applications or networks. Packetization determines how many sample voice frames, produced by the speech coder, are to be loaded into the same packet, before leaving the sender. Packetization delay refers to the delay incurred in this process. As more voice frames are to be loaded, the packetization delay increases. Since a speech coder produces a sample voice frame at a specific rate, it takes time to collect the needed number of voice frames. VoIP is delay-sensitive. Thus, packetization generally needs to be small so that it does not cause too much delay. A typical VoIP packet requires at least 40 bytes of overhead; 20 bytes of the IP header, 8 bytes of the UDP header, and 12 bytes of the RTP header. The overhead from the data link layer may be added, but it is usually not considered because the data link layer overhead varies when the packets travel across different physical networks. Depending on the speech coder type and packetization, the payload of a VoIP packet typically ranges from 10 to 40 bytes. It can be seen that, for VoIP, the packet overhead is usually larger than the payload. Thus, a large percentage of the bandwidth requirement is used for the transport of overhead bytes. Determining packetization is critical because it not only affects the packetization delay, but also the packet size and bandwidth requirement.

Table 2-1 Important characteristics of well-known speech coders

Standard	Coding	Effective Voice Bandwidth (Kbps)	Sample Voice Frame Delay (ms)	Sample Voice Frame Size (bits)
G.711	PCM	64	0.125	8
G.726	ADPCM	32	0.125	4
G.729	CS-ACELP	8	10	80

Table 2-1 shows important characteristics of well-known speech coders, which are needed for calculating bandwidth requirements. Effective voice bandwidth refers to the output bitrate of the coder. Sample voice frame delay refers to the time interval in which the coder samples voice signal and outputs a voice frame. Sample voice frame size refers to the size of the output voice frame. It is the smallest data unit for the payload of a voice packet. Below, we explore the relationship between packetization and bandwidth requirements.

Let

D_{frame}	Sample voice frame delay (ms)
F	Sample voice frame size (bits)
n	Number of sample voice frames in the payload
H	Header size of the voice packet (bits)

$$\text{Effective voice bandwidth} = \frac{F}{D_{frame}} \quad \text{Kbps} \qquad (2\text{-}1)$$

$$\text{Overhead bandwidth} = \frac{H}{nD_{frame}} \quad \text{Kbps} \qquad (2\text{-}2)$$

$$\text{Network bandwidth} = \frac{H + nF}{nD_{frame}} \quad \text{Kbps} \qquad (2\text{-}3)$$

Effective voice bandwidth in Equation 2-1 is the output bitrate of the speech coder, which is constant regardless of packetization. The effect of packetization can be seen in Equation 2-2, which depends on the number of sample voice frames in the payload. Small packetization results in a large overhead bandwidth. On the other hand, increasing packetization helps to reduce the overhead bandwidth. The network bandwidth requirement of VoIP (Equation 2-3) is the sum of effective voice bandwidth (Equation 2-2) and overhead bandwidth (Equation 2-3). Hence, packetization also has an impact on the bandwidth requirement.

Figure 2-1 is a composite plot of the above equations, which illustrates the effect of packetization. The plot is based on parameters from the G.726 standard. The lower horizontal scale is the number of sample voice frames in the payload. Accordingly, the upper horizontal scale is the packetization delay, which is the product of the number of sample voice frames and the sample voice frame delay. The dashed line represents the effective voice bandwidth.

The difference between the solid line and the dashed line represents the overhead bandwidth. It can be seen that packetization has an impact on the overhead bandwidth by which it exhibits a decreasing function. With small packetization, the payload-to-overhead ratio is small. That is, in addition to the effective voice bandwidth, a large percentage of the bandwidth is required for the transport of the packet overhead. This results in a large network bandwidth requirement. By using large packetization, we increase the payload-to-overhead ratio. This helps to reduce the overhead bandwidth, which in turn also reduce the bandwidth requirement. However, a drawback of large packetization is that it increases packetization delay. This is undesirable because it could affect the limited end-to-end delay budget.

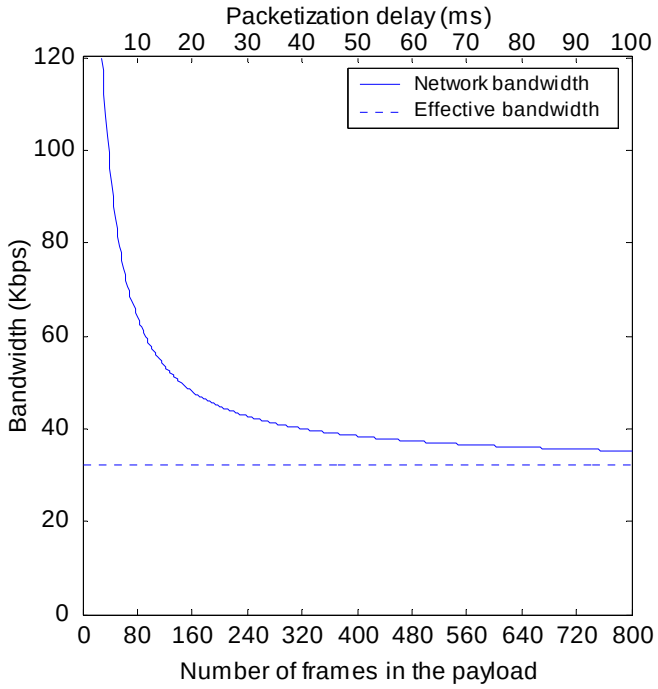

Figure 2-1 Relationship between packetization and bandwidth requirements

A typical setting for VoIP is to use a fair value of packetization that compromises between the bandwidth requirement and the incurred packetization delay. The problem is that pre-determined packetization cannot allow optimal performance. This is because the network condition varies over time. By varying packetization, we would be able to adjust the bandwidth requirement to match the available network bandwidth, which could help to optimize the performance. Figure 2-1 illustrates how we can make use of packetization.

When sufficient network bandwidth is available, using small packetization allows minimal packetization delay. When the network is congested, using large packetization reduces the bandwidth requirement, which could help reduce the congestion. Figure 2-1 also shows that only a certain range of packetization is feasible for varying the bandwidth requirement. Small packetization (e.g. less than 10-ms packetization delay) causes an extremely large bandwidth requirement, which is not worth the decreased packetization delay. Large packetization (e.g. more than 30-ms packetization delay) does not significantly reduce the bandwidth requirement, but may incur too much packetization delay.

Adaptive-rate VoIP can use packetization as a means for adjusting the transmission rate. This approach has an advantage that it can work with any constant bitrate speech coder. Another advantage is that rate adaptation is independent to the output bitrate of the coder. Therefore, rate adaptation is likely to be transparent to the user. A side effect of this approach is variable packetization delay, which must be managed at the receiver. Compared to variable bitrate coders, rate adaptation is done at the coder. There is no impact on packetization. However, varying the output bitrate of the coder can affect the output voice frame and the audio quality. Rate adaptation could cause distraction and may not be transparent to the user.

2.3 Effect of Packetization on VoIP Performance

The previous section describes the relationship between packetization and bandwidth requirements. It demonstrates that packetization is an excellent approach for varying the bandwidth requirement and for adaptive-rate VoIP. Here, we extend the study to determine whether varying packetization can offer benefits beyond the bandwidth requirement. Specifically, if we could optimize packetization under a current network condition, we should be able to improve the performance of VoIP. This study also demonstrates the feasibility of adaptive-rate VoIP based on packetization.

2.3.1 Study Methodology

We conduct a simulation study using the Network Simulator 2 or ns-2 [12]. The network topology for the simulation is shown in Figure 2-2. All nodes implement FIFO scheduling and drop-tail queuing. The link between node 0 and 1 has capacity of 10 Mbps

with propagation delay of 35 milliseconds. To ensure that the network is fairly congested, the offered load on this link is set to 60 percent on average. The link between node 1 and 2 has limited capacity so as to create the bottleneck. The link capacity varies as a factor in the simulation, with propagation delay of 5 milliseconds. Cross traffic on each link is generated from nine Pareto sources with the α parameter of 1.5, i.e. the inter-arrival time has infinite variance. The aggregation of many Pareto sources with α less than 2 has been shown to produce Long Range Dependent (LRD) traffic [13]. Measurement studies have shown that packet size distribution on the Internet is centered on three values [14, 15]. Namely, about 60% of the packets are 40 bytes, 25% are 550 bytes, and 15% are 1500 bytes. In our simulation, we follow such a finding when generating the cross traffic. Note that, in terms of load distribution, about 7% of the load is 40-byte packets, 35% is 550-byte packets, and 58% is 1500-byte packets. The VoIP session lasts 120 seconds in which the source is assumed using the ADPCM codec, with the effective voice bandwidth of 32 Kbps.

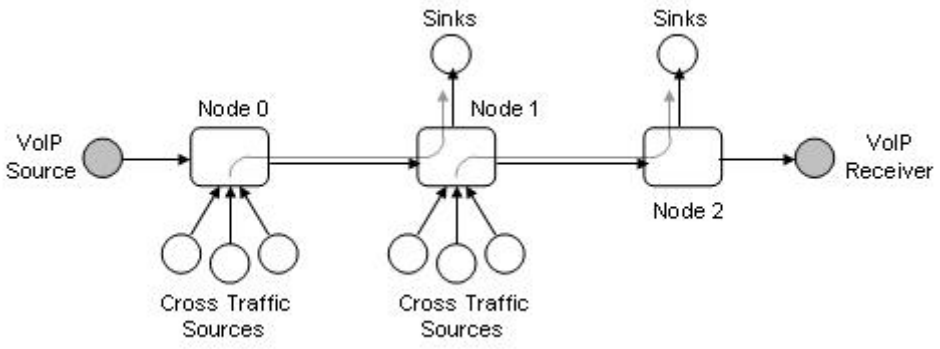

Figure 2-2 Simulation network topology

Packetization and cross traffic load offered to the bottleneck link are the key factors in the simulation. From Figure 2-1, we consider the feasible range of packetization from 80 to 240 bytes of payload; or, accordingly, from 10 to 30 milliseconds of packetization delay. With the small voice frame delay of 0.125 milliseconds, the packetization can have any value within the feasible range. In the simulation, we use the increment step of 5 milliseconds. So, the values of the packetization factor are 10, 15, 20, and so on, in milliseconds. Because the VoIP bandwidth requirement is relatively small, it is also interesting to study how that plays

a role on the performance. Thus, we add another factor: the significance of VoIP traffic, which is defined as the effective voice bandwidth divided by the bottleneck link capacity. It is used for comparative purpose only to estimate the percentage of VoIP traffic. Given a VoIP flow, the actual traffic varies as it depends on the packetization.

To evaluate the performance, we make measurements of one-way network delay and packet loss. Note that the network delay is associated with the packet level. Whereas packetization is the factor that contributes a delay to the sample voice frame, evaluation of the delay performance must be done at the voice frame level that includes the effect of packetization delay. Here, we define one-way end-to-end delay as the latency of a voice frame from when the codec begins to collect the voice samples to when that voice frame is received at the receiver and ready to be decoded. The end-to-end delay can be found by the sum of the measured network delay and the corresponding packetization delay. We use the 90^{th} percentile of end-to-end delay as a performance metric, instead of the commonly used mean delay. In VoIP, early arriving packets are held in the jitter buffer so that late arriving packets can still be in time for a smooth playout. The 90^{th} percentile of end-to-end delay virtually accounts for an estimate of the jitter buffer delay. Thus, it can better reflect the actual delay that the user may experience. Similarly, packet loss is associated with the packet level. Since packetization determines the number of voice frames in the payload, more voice frames are lost when a packet with large packetization is lost. Thus, the measured packet loss is converted to voice frame loss, based on the corresponding packetization. Voice frame loss rate is used to indicate the percentage of voice frames being lost in the network.

2.3.2 Simulation Results

Table 2-2, 2-3, 2-4, and 2-5 are the simulation results showing the effect of packetization when the bottleneck link capacity is 128, 256, 512, and 768 Kbps, respectively. The tables show the performance of the VoIP flow under the two factors: packetization and cross traffic load offered to the bottleneck link. Each cell in the table includes the 90^{th} percentile of end-to-end delay (on the top) and voice frame loss rate (on the bottom). Focusing on the delay performance, we accordingly use the data from the tables to plot the 90^{th} percentile of end-to-end delay in Figure 2-3, 2-4, 2-5, and 2-6.

Table 2-2 Delay and loss performance for bottleneck link of 128 Kbps

		Packetization (ms)				
		10	15	20	25	30
Average cross traffic load over bottleneck link (%)	60	540.93 ms 7.83%	655.65 ms 4.41%	715.65 ms 2.47%	763.15 ms 1.44%	803.15 ms 0.80%
	50	376.83 ms 0.63%	225.48 ms 0.05%	206.46 ms 0.03%	201.63 ms 0.02%	201.98 ms 0.03%
	40	214.44 ms 0.10%	180.71 ms 0.09%	173.58 ms 0.07%	173.73 ms 0.12%	176.66 ms 0.10%
	30	150.30 ms 0.10%	146.33 ms 0.05%	148.85 ms 0.12%	152.17 ms 0.03%	154.69 ms 0.07%
	20	130.1 ms 0.12%	133.81 ms 0.16%	133.65 ms 0.18%	136.00 ms 0.12%	139.07 ms 0.10%

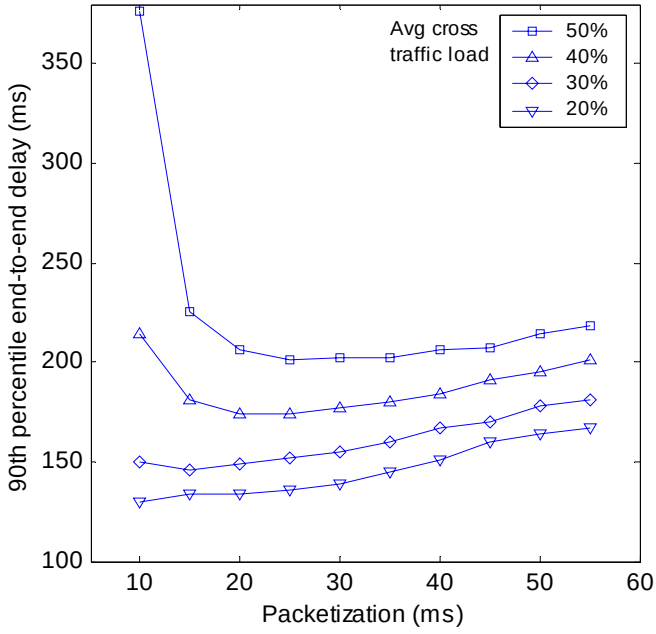

Figure 2-3 Plot of end-to-end delay for bottleneck link of 128 Kbps

Table 2-3 Delay and loss performance for bottleneck link of 256 Kbps

		Packetization (ms)				
		10	15	20	25	30
Average cross traffic load over bottleneck link (%)	75	384.49 ms 4.25%	452.28 ms 2.86%	477.28 ms 1.98%	500.27 ms 1.67%	520.51 ms 1.48%
	70	305.63 ms 1.44%	283.91 ms 0.56%	268.11 ms 0.40%	249.28 ms 0.33%	231.92 ms 0.38%
	65	268.19 ms 1.10%	232.13 ms 0.59%	217.46 ms 0.28%	216.04 ms 0.25%	214.02 ms 0.15%
	60	156.63 ms 0.42%	141.02 ms 0.10%	134.80 ms 0.08%	136.54 ms 0.06%	140.13 ms 0.12%
	50	135.38 ms 0.23%	125.29 ms 0.14%	126.41 ms 0.19%	127.84 ms 0.02%	131.91 ms 0.15%
	40	92.50 ms 0.20%	95.59 ms 0.25%	100.14 ms 0.23%	106.08 ms 0.17%	111.16 ms 0.33%

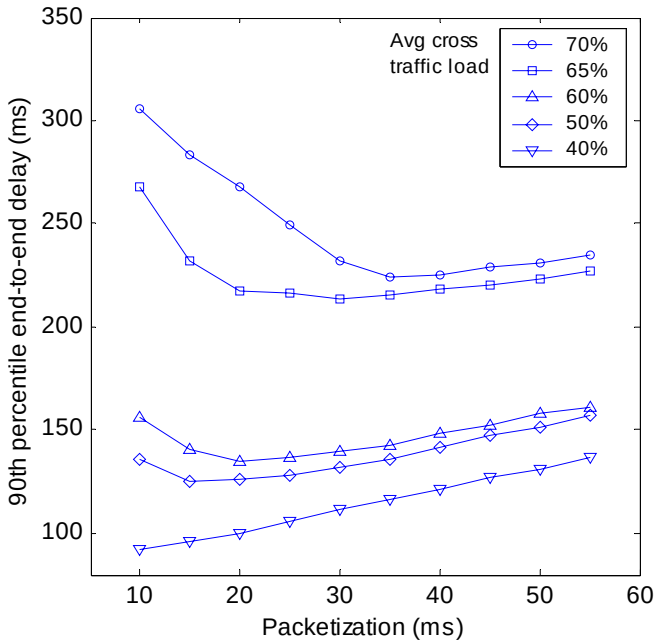

Figure 2-4 Plot of end-to-end delay for bottleneck link of 256 Kbps

Table 2-4 Delay and loss performance for bottleneck link of 512 Kbps

		Packetization (ms)				
		10	15	20	25	30
Average cross traffic load over bottleneck link (%)	80	266.30 ms 3.65%	283.17 ms 2.69%	300.67 ms 2.73%	308.65 ms 2.44%	327.38 ms 2.4%
	75	218.01 ms 1.11%	232.58 ms 0.91%	235.13 ms 0.48%	234.74 ms 0.77%	235.13 ms 0.53%
	70	150.23 ms 0.43%	143.11 ms 0.24%	141.54 ms 0.17%	145.09 ms 0.08%	147.16 ms 0.17%
	65	107.15 ms 0.14%	104.82 ms 0.16%	107.32 ms 0.12%	109.81 ms 0.15%	112.79 ms 0.07%
	60	81.82 ms 0.06%	85.00 ms 0.04%	89.26 ms 0.05%	93.86 ms 0.06%	99.04 ms 0.02%
	50	74.73 ms 0.07%	79.58 ms 0.04%	84.86 ms 0.05%	90.05 ms 0.02%	94.88 ms 0.07%

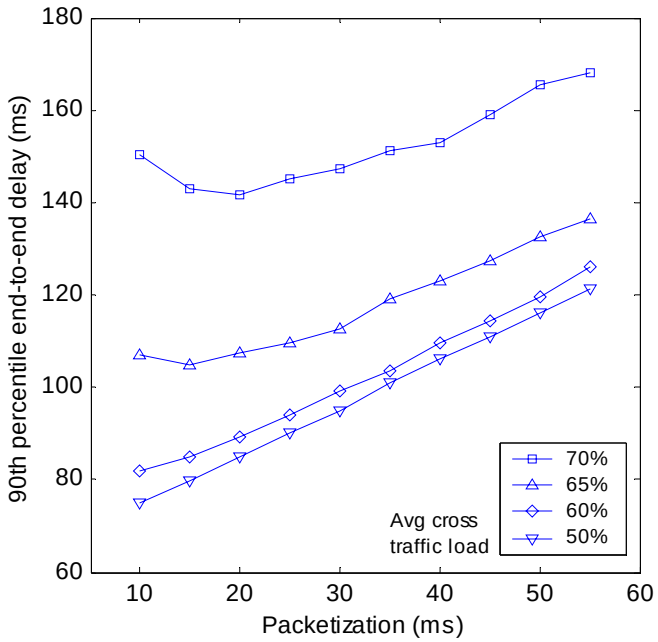

Figure 2-5 Plot of end-to-end delay for bottleneck link of 512 Kbps

Table 2-5 Delay and loss performance for bottleneck link of 768 Kbps

		Packetization (ms)				
		10	15	20	25	30
Average cross traffic load over bottleneck link (%)	85	230.23 ms 4.93%	248.33 ms 4.14%	259.03 ms 4.30%	269.96 ms 4.15%	279.69 ms 4.28%
	80	155.31 ms 1.52%	162.12 ms 1.23%	168.39 ms 1.28%	171.26 ms 1.38%	176.39 ms 1.15%
	75	121.67 ms 0.74%	121.41 ms 0.66%	122.81 ms 0.57%	126.47 ms 0.54%	129.80 ms 0.72%
	70	83.89 ms 0.31%	86.77 ms 0.28%	90.82 ms 0.17%	95.64 ms 0.31%	99.93 ms 0.15%
	60	69.37 ms 0.12%	74.00 ms 0.17%	79.02 ms 0.12%	83.92 ms 0.10%	88.88 ms 0.12%

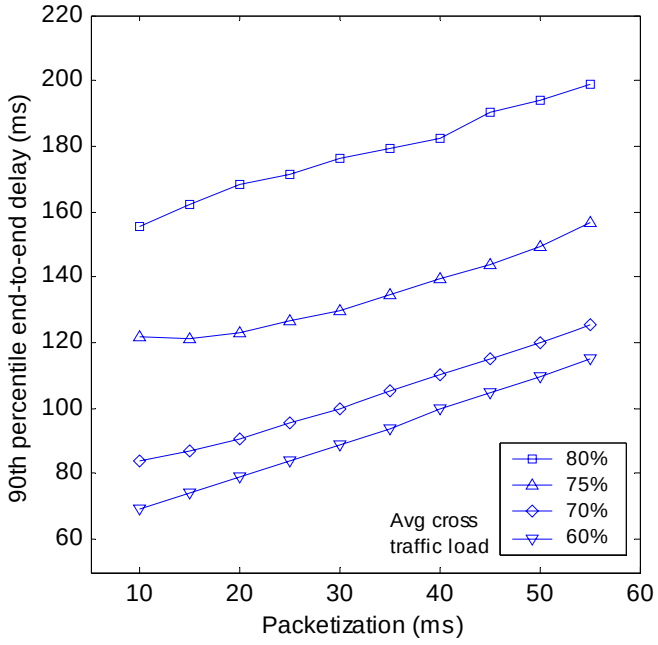

Figure 2-6 Plot of end-to-end delay for bottleneck link of 768 Kbps

The effect of packetization on the performance can be seen from the plot of end-to-end delay. Whereas the voice frame loss rate is relatively the same, the plot of the end-to-delay illustrates a concave curve; except in the case of the bottleneck link of 768 Kbps, which will be discussed later. The concave curve demonstrates that the important inherent trade-off of packetization. Recall that end-to-end delay consists of two delay components: packetization delay and network delay, more specifically queuing delay. Packetization appears to have an impact on both of the delay components, but in an opposite direction. While small packetization is desirable in order to minimize packetization delay, it results in a large bandwidth requirement. This causes no problem if sufficient network bandwidth is available, i.e. the network is lightly loaded. In this case, the plot of end-to-end appears as a straight line because it is mainly affected by packetization delay. Thus, it is best to use the smallest packetization. If the network load increases, such a large bandwidth requirement becomes a problem because it causes an increase in the network delay. Due to congestion at the bottleneck queue, the increasing network delay can be much higher than the saving in delay when using small packetization. As a consequence, the end-to-end delay remains high. This happens to the left side of the inflection point. On the other hand, large packetization may not be desirable because of the increased packetization delay, but it can reduce the bandwidth requirement. This, in turn, could help to avoid congestion and minimize network delay. A problem is that, if not used properly, too large packetization may unnecessarily contribute more delay to the end-to-end delay. This can be seen to the right of the inflection point, in which the end-to-end delay is directly affected by packetization delay. The plot of end-to-end delay suggests that optimizing packetization can help the VoIP flow to achieve the highest possible performance. The packetization is optimal when the network delay is minimized and there is no excess packetization delay. This result is consistent with the mathematical analysis and experiment study in [16].

Consider the network load factor. As the offered load to the bottleneck link increases, the overall end-to-end delay increases more dramatically. This is primarily due to the effect of queuing delay. As the offered load increases, network bandwidth becomes less available. It is necessary for the optimal packetization to be larger in order to reduce the bandwidth requirement even more, to match the decreasing available bandwidth. This shows that the

optimal packetization is not stationary, but instead varies depending on the current network condition.

Not shown in the plot of end-to-end delay, Table 2-2, 2-3, and 2-4 include the delay performance when the voice frame loss rate is more significant, at higher offered load. It can be seen that the end-to-end delay as a function of packetization does not exhibit the concave curve as mentioned earlier. Instead, smaller packetization gives a smaller delay, while larger packetization gives a higher delay. This happens because the effect of packet loss takes over the effect of queuing delay. With small packetization, the bandwidth requirement could be larger than the available network bandwidth. It would have resulted in a very high delay if the bottleneck link had an infinite queue size. In reality, routers and switches have a limited size buffer. As the bottleneck link is based on FIFO scheduling and drop-tail queuing, excessive loss as experienced by the VoIP flow means that the cross traffic encounters the same excessive loss as well. Therefore, as more packets are dropped at the bottleneck, it shortens the queue length and results in a lower delay. This shows that the effect of packetization that exhibits as the concave curve happens only when packet loss does not cause a significant impact. When the network becomes congested, excessive packet loss is another factor that can affect the network delay, and hence the end-to-end delay.

The factor of the significance of VoIP traffic plays an important role in determining the effectiveness of optimizing packetization. From Figure 2-3, 2-4, 2-5, and 2-6, the significance of VoIP traffic is 25%, 12.5%, 6.25%, and 4.16%, respectively. We can see from the figures that, when the significance of VoIP traffic is high, optimizing packetization can considerably help to reduce the end-to-end delay. As the significance of VoIP traffic gets smaller, the benefit of optimizing packetization decreases. In Figure 2-6, the plot of end-to-end delay does not exhibit the concave curve, regardless of how congested the network is. In such an environment with a large bottleneck link, optimizing packetization appears to give no benefit. This happens because the VoIP traffic is a small fraction of the bottleneck link. The change in the bandwidth requirement cannot make a significant impact on the overall traffic load, hence relatively causing no impact on the network delay. Accordingly, examine Table 2-5, given an offered load, the delay and loss performance appears to be the same, regardless of the packetization. Note that the table shows the end-to-end delay. The network delay can be found by which it is the end-to-end delay less the corresponding packetization delay. In

this situation, the performance is largely determined by the overall traffic load. Therefore, small packetization should be used in order to avoid an unnecessary extra delay caused by the packetization delay.

The findings above suggest that using pre-determined packetization cannot allow the highest possible performance. This is because the performance depends upon several factors, including packetization, network load, significance of VoIP traffic, and packet loss. Adaptive-rate VoIP can take advantages of the findings by attempting to optimize packetization based on observing the current network condition. In a situation where the VoIP flow does not make up a significant amount of traffic, the performance is dominated by the overall traffic load. Adaptive-rate VoIP can optimize the performance by operating at the smallest packetization. Although this may cause a large bandwidth requirement, it has little impact on the increasing network delay. Using small packetization ensures to eliminate unnecessary delay due to packetization.

Because the VoIP bandwidth requirement is relatively small, the significance of VoIP traffic becomes the factor that limits the effectiveness of optimizing packetization. Nonetheless, the significance of VoIP traffic can increase when multiple VoIP flows share the same bottleneck link. In such a case, we expect that optimizing packetization can yield the benefit to all the VoIP flows. To confirm this presumption, we extend our simulation study by using a broadband bottleneck link of 1.544 Mbps, with the offered load around 60 percent on average. We increase the number of VoIP flows across the network. In each case, we observe the delay and loss performance. Hence, the performance is studied under two factors: packetization and the number of VoIP flows. The result is shown in Table 2-6, and accordingly the plot of end-to-end delay is shown in Figure 2-7. Each cell in the table includes the average of the 90^{th} percentile of end-to-end delay from the corresponding flows (on the top), and the average of voice frame loss rate from the corresponding flows (on the bottom).

Table 2-6 Delay and loss performance in the case of multiple VoIP flows

		Packetization (ms)				
		10	15	20	25	30
Number of VoIP flows across the network	8	131.34 ms 1.67%	126.89 ms 0.74%	119.01 ms 0.49%	115.10 ms 0.43%	114.82 ms 0.42%
	7	121.43 ms 1.01%	107.80 ms 0.58%	102.39 ms 0.42%	101.61 ms 0.31%	102.92 ms 0.27%
	6	104.10 ms 0.58%	90.65 ms 0.33%	89.63 ms 0.24%	90.11 ms 0.22%	94.08 ms 0.23%
	5	84.37 ms 0.36%	79.23 ms 0.27%	81.75 ms 0.21%	85.02 ms 0.17%	89.40 ms 0.20%
	4	72.62 ms 0.26%	73.37 ms 0.23%	77.12 ms 0.18%	81.56 ms 0.13%	86.38 ms 0.18%

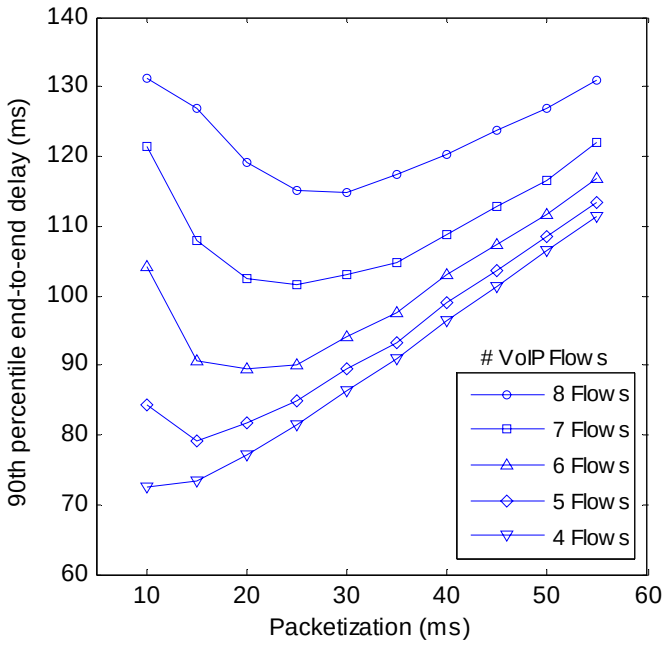

Figure 2-7 Plot of end-to-end delay in the case of multiple VoIP flows

As expected, Table 2-6 and Figure 2-7 show the same pattern of result as in the previous simulation. Due to the high speed bottleneck link, a couple of VoIP flows can be minority traffic in which the performance is dominated by the overall traffic load. The concave curve starts to appear when four VoIP flows share the same bottleneck. That is, the VoIP traffic becomes significant enough to have an impact on the network. The significance of VoIP traffic increases as the number of VoIP flows increases. With the increasing number of VoIP flows, the network also becomes more congested. Optimizing packetization helps to reduce the overall VoIP bandwidth requirement. This in turn minimizes congestion, and results in reducing the end-to-end delay. This simulation shows that, in a high speed network, we can still benefit from optimizing packetization, if multiple VoIP flows make up a significant level of traffic load. This is particularly true in a mostly VoIP network, but not the public Internet.

2.4 Summary

This chapter addresses a fundamental issue of adaptive-rate VoIP. VoIP typically has a fixed bandwidth requirement. The question is how to make VoIP to be able to vary the bandwidth requirement; in other words, to transmit packets at variable rates. A common approach to this question is to use variable rate speech coders. Here, we propose an alternative of using packetization as a means to vary the bandwidth requirement of VoIP. We begin by studying the relationship between packetization and bandwidth requirements. Whereas the effective voice bandwidth (or the output bitrate of the coder) is fixed, the needed overhead bandwidth is a decreasing function of packetization. Hence, the network bandwidth requirement, which is the sum of the voice and overhead bandwidth, is also a decreasing function of packetization. Therefore, by varying packetization, we can adjust the bandwidth requirement. The advantage of using packetization is that the output voice frames from the speech coder are not affected by, and are independent to, the rate adaptation by packetization. Thus, rate adaptation is likely to be transparent to the user.

The primary goal of this chapter is to study the effect of packetization on VoIP performance. This study answers the question whether or not adaptive-rate VoIP based on packetization can actually help to optimize the performance. The simulation result shows that

the VoIP performance is mainly affected by three factors: packetization, network load, and significance of VoIP traffic. The effect of packetization is illustrated in the concave curve of the plot of end-to-end. The concave curve also shows the inherent trade-off of packetization. Namely, small packetization is usually preferred for minimal incurred packetization delay. But, because of a large bandwidth requirement, it has a potential to cause congestion, which could result in increasing end-to-end delay. On the other hand, when the network becomes congested, large packetization can reduce the bandwidth requirement, which could help to minimize the congestion and reduce the end-to-end delay. However, using too large packetization may incur unnecessary extra delay due to packetization. Optimizing packetization allows us to vary the bandwidth requirement to match the available network bandwidth. When network congestion is minimized, the VoIP flow can achieve the highest possible performance.

The network load factor basically affects the overall delay and loss performance. When the network load increases, available bandwidth becomes less available. The optimal packetization needs to be larger in order to reduce the bandwidth requirement even more to match the decreasing network bandwidth. The factor of the significance of VoIP traffic plays a role in determining the effectiveness of optimizing packetization. The VoIP flow must make up a significant level of traffic load so that it can have an impact on the network. Otherwise, the performance would be dominated by the other traffic. In a situation where the VoIP flow is minority traffic, using the smallest packetization is suggested because it causes no excess packetization delay. Although this may cause a large bandwidth requirement, it has little impact on the increasing network delay.

Because VoIP performance depends upon several factors, it is clear that using pre-determined packetization cannot allow the highest possible performance. The findings from this study can be very useful and provide insights about when and how adaptive-rate VoIP can improve the performance. Adaptive-rate VoIP can take advantages of optimizing packetization by observing the current network condition and then respond properly. This study also demonstrates the feasibility of adaptive-rate VoIP based on packetization. However, in order to build an adaptive-rate VoIP system, observing the state of the network becomes the next challenge that must be accomplished. Because the VoIP sender and receiver are typically not time synchronized, it is almost impossible to observe the packet

delay; unless we use an external source that provides a synchronized timing. Observing the state of the network is thus limited to the commonly used packet loss and delay variation. In the next chapter, we focus on the issue of detecting the state of the network. We propose a measurement methodology that allows adaptive-rate VoIP to be able to observe the full spectrum of the network characteristics, including network delay, available network bandwidth, packet loss, and etc.

Chapter 3

Sync & Sense: Measurement Methodology for Network State Detection and Delay Assessment

Unreliable and unpredictable voice quality is a major problem for VoIP. The underlying cause of the problem is that packet-switched networks, including the Internet, only provide best-effort service, which offers no guarantees regarding delay, loss, and bandwidth. Under such an environment, VoIP needs to be flexible and adaptable in order to gain the most from whatever the available resources. For example, a VoIP system may observe alternate routes and choose the optimal one for transmitting the packets. A VoIP system may be equipped with a control mechanism that attempts to adjust the bandwidth requirement to match the available network bandwidth. Detecting the state of the network is an essential component that allows this adaptability. In other words, a VoIP system needs to be able to know the characteristics of the network before it can react upon them. Because VoIP is delay-sensitive, detecting the state of the network must serve the objective of acquiring the delay characteristics, particularly the one-way network delay. This, however, is a daunting task. The challenge lies in the fact that the Internet implicitly enforces the end-to-end principle. This means the endpoints are expected to operate independently, without support from the network. Because the VoIP sender and receiver have their own independent clock, without external synchronized timing, it is almost impossible to measure one-way network delay.

Here, we propose a novel measurement methodology for VoIP called Sync & Sense of periodic stream that can overcome such a challenge. Sync & Sense has the ability to virtually synchronize the transmission and reception timing of the VoIP session, which enables the measurement of one-way network delay, more specifically the queuing delay component. Because queuing delay is highly variable and typically unpredictable, being able

to measure the queuing delay is very crucial. It allows Sync & Sense to estimate the propagation delay and obtain the full spectrum of the delays of the VoIP session. In addition, Sync & Sense has the ability to estimate the available network bandwidth. Along with packet loss, Sync & Sense can provide a wider range of indications about the network condition. Our simulation study shows that the Sync & Sense measurement methodology is highly accurate and robust to packet loss. Therefore, Sync & Sense can offer tremendous benefits to VoIP. Adaptive-rate VoIP can significantly benefit from Sync & Sense by which it can have the clear picture of the network condition, and react properly to optimize the performance.

3.1 Background and Related Research

As is well-known, packet-switched networks, including the Internet, provide best-effort service that offers no guarantees for packet delivery. The transmitted packets could be delayed, lost, duplicated, corrupted, etc. It is the application's responsibility to ensure that the transmission of the packets is successful and meets the quality of service requirement. The Internet, in fact, implicitly enforces the end-to-end principle. That is, the endpoints are expected to operate independently, without assistance from the network. At the same time, the endpoints are expected to be smart enough to have mechanisms for error detection, flow control and congestion control. Detecting the state of the network is an essential component because it allows the control to be able to observe the network condition before making a control decision. For data applications, detecting the state of the network can be simple. Data traffic is loss-sensitive, but delay-insensitive. Thus, detecting packet loss is effective and sufficient as it serves the objective of avoiding excessive packet loss. This allows less retransmission and results in optimal throughput. The packets may experience considerable delay, but it is really not an interest of the control. The major concern is to ensure the reliability of the transmission and the integrity of the data.

VoIP typically uses UDP as its transport protocol, with no control mechanism. Voice packets are transmitted at a constant rate regardless of the state of the network. In dealing with the unpredictable behavior in the network characteristics, researchers aim to enhance VoIP with some kind of control. Adaptive-rate VoIP is expected to be smart enough to detect the state of the network, and adapt the transmission accordingly for optimal performance.

VoIP traffic is delay-sensitive and has a strict delay requirement. ITU-T Recommendation G.114 [17] specifies the one-way transmission time of 150 milliseconds or less for acceptable voice quality. In fact, both delay and loss are critical to VoIP. The perceived quality is generally determined by one-way end-to-end delay for echo and conversational interactivity, and packet loss for voice clarity [18]. Therefore, detecting the state of the network for VoIP must serve the objective of minimizing delay and avoiding excessive packet loss. That is, the control must observe and consider both one-way end-to-end delay and packet loss for its decision making.

Most of the previous works on adaptive-rate VoIP [6, 19, 20] primarily focus on using packet loss as a means for detecting the state of the network. A number of works take the next step by incorporating measurements into a perceived quality assessment scheme. In Qiao et al's work [5], speech quality is predicted from packet loss rate using a PESQ (Perceptual Evaluation of Speech Quality) based method. Mohamed et al [21] focus on developing a neural network based automaton that measures speech quality in real-time. The authors build a database comprised of a set of samples of distorted speech signals (based on packet loss rate and loss distribution) and their associated subjective quality scores (i.e. MOS). The database is used to train the neural network to assess speech quality.

What the previous works mentioned above have in common is that they seem to address only voice quality as indicated by packet loss, but not conversational interactivity as indicated by one-way end-to-end delay. Delay matters significantly, but is often ignored. In a sense, packet loss seems to be the only available implicit indication that allows the endpoint to detect the state of the network. Given the end-to-end principle, the endpoints are expected to operate without assistance from the network. Since the VoIP sender and receiver have their own independent clock, without an external synchronized timing, it is almost impossible to measure the one-way delay. This might be a reason why one-way delay is ignored in the previous works. Although round-trip delay can be obtained, one-way delay is often not well approximated by dividing the round-trip delay in half [22]. This is due to the fact that the network is asymmetric. While RTP (and its associated RTCP) is primarily designed to satisfy the needs of real-time applications including VoIP, one must be aware that RTP is a protocol framework that is deliberately not complete [8]. The RTP specification includes only functions expected to be common across all the applications for which RTP would be

appropriate. Relying on RTP and RTCP may not be sufficient in detecting the state of the network. Specifically, RTP does not provide a mechanism for measuring one-way network delay. Therefore, in order to serve the objective of detecting the state of the network, VoIP needs an additional mechanism that can observe the one-way network delay.

3.1.1 The Extent of RTP to Network State Detection

This section examines the RTP specification (RFC 1889) [8] with an aim to describe the extent of RTP to detecting the state of the network. As stated in the specification, RTP (Real-time Transport Protocol) provides end-to-end network transport functions suitable for applications transmitting real-time data such audio or video. However, RTP itself does not provide any mechanism to ensure timely delivery, or provide congestion control or other quality of service guarantees. RTP represents a new style of protocol following the two philosophies of the end-to-end principle and application level framing. The end-to-end principle [4] implies that the endpoints are intelligent and network-aware. This is well suited to the Internet as a dumb network. The concept of application level framing [23] states that, because different applications have different needs, the application has the best knowledge of its data to make a decision about how that data should be transported. It is clear from the specification that, whereas VoIP is based on RTP, additional mechanisms may still be required in order to meet the needs of the application. Below, we summarize the functions provided by RTP that allows the application to detect the state of the network. The observation of one-way network delay is not mentioned in the specification and is the problem that must be addressed by the application itself.

The primary functions provided by RTP include payload type identification, sequence number, and timestamp. The sequence number is used by the receiver to restore packet sequence as well as detect packet loss. It is important to note that the timestamp is not a reading of the system clock. It does not facilitate the receiver to measure one-way network delay of the RTP packet. The timestamp is used by the receiver to schedule the playout of the voice. The initial value of the timestamp is randomly chosen. The timestamp reflects the sampling instant of the first octet of payload in the RTP packet.

As RTP provides delivery of real-time data, it is augmented by its associated control protocol called RTCP. RTCP provides periodic reports along with the RTP session. While RTP packets are sent every few milliseconds, RTCP reports are sent on the scale of seconds. Five types of the RTCP packet are defined in the specification: receiver report (RR), sender report (SR), source description (SDES), membership management (BYE), and application-defined (APP). RR is of interest here as it provides reception quality feedback. The reception quality feedback can be useful for applications like adaptive-rate VoIP. The RR packet format includes the LSR and DLSR fields that allow the sender to calculate the round-trip time. LSR (last sender report) is the timestamp of the most recent SR packet received by the receiver. DLSR (delay since last sender report) is the delay between receiving the last SR packet and sending this RR packet. When receiving an RR packet, the sender subtracts the LSR field from the current time, which gives the delay between sending the SR packet and receiving this RR packet. To get the round-trip time, the sender subtracts the DLSR field to remove the offset introduced by the delay in the receiver. The process is shown in Figure 3-1, an example taken from the RTP specification.

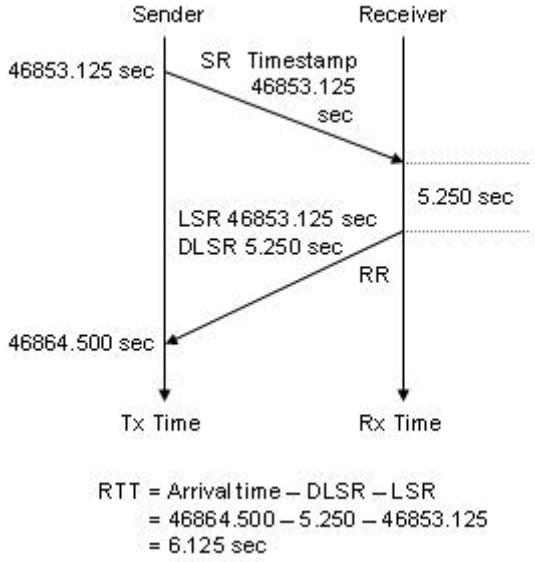

Figure 3-1 Round-trip time calculation as provided by RTCP

Besides the round-trip time, the RR packet format includes the following fields that provide information about the network condition. The cumulative number of packets lost field provides long-term packet loss measurements. The fraction lost field gives a short-term measurement from a single RR packet. While packet loss tracks persistent congestion, the inter-arrival jitter field can provide a measure of transient congestion. Since the inter-arrival jitter field is only a snapshot of the jitter at the time of a report, it is necessary to analyze a number of reports over time.

3.1.2 End-to-End Measurement Techniques

Besides using RTCP, end-to-end measurement techniques may be used to detect the state of the network. Without support from the network, these techniques utilize some kind of probe packets. The sender transmits probe packets of a certain pattern through the network to the receiver. Along the path, the probe packets experience variable delays incurred by the network. This causes a changing pattern of the probe packets, for instance, the interval gap between consecutive packets and packet loss. By observing the incoming probe packets, the receiver may infer the state of the network. End-to-end measurement techniques may be classified into three major categories: variable packet size probing, packet pair/train probing, and periodic stream probing. Below is a brief review of the techniques. A more comprehensive survey can be found in [25].

Variable packet size probing aims to measure the capacity of each hop along the path. The technique measures the round-trip time (RTT) from the sender to each hop of the path as a function of the probe packet size. The technique uses the Time-To-Live (TTL) field of the IP header to force the probe packets to expire at a particular hop. The router at that hop discards the probe packets, and returns an ICMP packet of time-exceeded error message back to the sender. The sender uses the received ICMP packet to measure the RTT to that hop. The RTT to each hop consists of three delay components in the forward and reverse paths: transmission, propagation, and queuing delays. Whereas multiple probe packets are transmitted, the technique assumes that at least one of those packets, together with the ICMP reply, will experience no queuing delay. Thus, the minimum RTT can be measured. Using variable packet sizes allow the technique to estimate the link capacity because the

transmission delay is a function of packet size and link capacity, whereas both the transmission delay and packet size are known. The following tools are based on the variable packet size probing technique: Patchar [49], Clink [50], and Pchar [51].

Packet pair/train probing is a technique to measure the end-to-end capacity of the path. Packet pair means that two packets of the same size are transmitted back-to-back. Assuming no cross traffic at the moment, when the packet pair goes through the link, they are separated (or dispersed) because of the limited link capacity. Along the end-to-end path, the dispersion is at most caused by the bottleneck link. Thus, the receiver can estimate the end-to-end path capacity by observing the dispersion of the packet pair. Packet pair techniques were originally discussed in the classic papers by Jacobson [31] and Bolot [26]. A problem of the packet pair technique is that, in reality, cross traffic can affect the dispersion of the packet pair and cause underestimation and overestimation of the path capacity. Several works propose to use statistical methods to filter out erroneous measurements due to the effect of cross traffic [38, 52, 53]. Packet train is a method that extends the packet pair technique by using multiple back-to-back packets. Available tools based on the packet pair/train probing technique are Bprobe [52], Nettimter [54], and Pathrate [55].

Periodic stream probing is a technique to estimate end-to-end available bandwidth. The sender transmits a periodic stream of equal-sized packets at a given rate. If the transmission rate is higher than the available bandwidth, the stream will induce congestion and cause an increasing trend in one-way network delay. At the same time, the receiver would observe the reception rate that is lower than the transmission rate. Based on the observations in the one-way delay or reception rate, the technique alters different transmission rates in order to search the given transmission rate that matches the available bandwidth. To observe one-way delays, the transmitted packets need to be time-stamped. A problem is that the endpoints are not clock-synchronized, which could affect the delay observations and the algorithm decision. Available tools based on the periodic stream probing technique are Pathload [56], TOPP [57], and PathChirp [58].

3.2 Sync & Sense Measurement Methodology

As discussed earlier, the RTP specification includes only functions expected to be common across all the applications for which RTP would be appropriate. The reception quality feedback of RTCP provides some information that can be useful for adapting the transmission rate. However, the reception quality feedback appears not to be able to serve the objective of detecting the state of the network for VoIP. Specifically, it lacks the measurement of one-way network delay, which is an important factor that determines the delivered quality. Thus, it is up to the application, such as adaptive-rate VoIP, to come up with a solution to measure the one-way network delay. Although there are several one-way delay measurement tools available, they rely on having an implementation of a synchronized clock between the endpoints, either using GPS (Global Positioning System) or NTP (Network Time Protocol). Examples of these tools are OWAMP (One-way Active Measurement Protocol) [59] and PingER (Ping End-to-end Reporting) [60]. These tools practically provide active measurement to study about the network delay and performance, rather than to be embedded in VoIP for the purpose of adaptability. Packets must be time-stamped both on the sender and receiver, thereby allowing accurate measurements to be made. The use of timestamp, however, is a drawback because it can affect the RTP packet format. The important drawback is that this approach requires support from the network. Since the Internet implicitly enforces the end-to-end principle, the endpoints are expected to be independent from the network. In reality, we could not assume that the endpoints are equipped with a synchronized timing source. Given such a constraint, measurement of one-way network delay becomes a real challenge.

Here, we propose a novel measurement methodology called Sync & Sense of periodic stream. Sync & Sense overcomes the challenge by which it can virtually synchronize the transmission and reception timing of the VoIP session. This allows Sync & Sense to be able to measure one-way network delay. Along with packet loss, Sync & Sense is a methodology that can truly achieve the objective of detecting the state of the network for VoIP. Note that Sync & Sense does not synchronize the clocks of the endpoints such as provided by NTP. The NTP protocol [24] specifies the message format, which includes timestamps. A few primary reference clocks are required by which the gateways use NTP to cross-check the clocks and synchronize them to local hosts. Sync & Sense is designed to have the desirable

properties. First, Sync & Sense follows the end-to-end principle, which is necessary for a system to work over the Internet. Sync & Sense performs on the receiving end by which it simply observes incoming RTP packets and makes measurements. Sync & Sense does not require any support from the network, making it independent of the network environment. Second, Sync & Sense is based on passive (or non-intrusive) measurement. Sync & Sense measures one-way network delay from the incoming RTP packets that can reflect a true quality-of-service of the VoIP session. In contrast to active measurement methods, Sync & Sense does not introduce extra traffic that could affect the VoIP session. Third, Sync & Sense detects the state of the network in a continuous manner, as opposed to a snapshot manner. This allows Sync & Sense to provide the clear picture of the network condition. Making measurements for every single incoming RTP packet is the finest grain available.

As indicated by its name, the two phases of the Sync & Sense methodology are synchronizing and sensing. The synchronizing phase is the ability to virtually synchronize the transmission and reception timing of the VoIP session, based solely on observing the incoming RTP packets. The sensing phase is the ability to measure dispersion times of the incoming RTP packets and further find the delay characteristics. Initially, Sync & Sense attempts to synchronize timing. When synchronized, the measurement can begin. The synchronizing phase also runs throughout the session, which constantly reaffirms the synchronization. This allows accurate measurements for each individual packet. Although the synchronizing phase leads to the sensing phase, it is more intuitive to convey the idea in the reverse order. Below, we describe the sensing phase, followed by the synchronizing phase.

3.2.1 Sensing Phase

Sync & Sense is considered in the same category of periodic stream probing because it utilizes the RTP packets themselves as the probe packets. Sync & Sense is different from existing works in several ways. As designed for VoIP, Sync & Sense performs passive measurement and does not introduce additional probing traffic to the network. Sync & Sense leaves the RTP frame format intact, and does not require time-stamping the packets in order to observe one-way network delays. Importantly, Sync & Sense can virtually synchronize the timing of the VoIP session, allowing measurement of one-way network delay. The RTP

packet format contains information that allows Sync & Sense to learn about the transmission pattern. In other words, Sync & Sense can have the probing pattern without requiring any initial setup. For example, an arriving RTP packet has payload size of 160 bytes. The RTP header indicates the G.711 PCM codec type. The characteristic of the PCM codec is that it outputs a sample frame of 1 byte every 0.125 milliseconds. Using this fact, Sync & Sense can calculate the packetization, which equals to 20 milliseconds (160 bytes * 0.125 ms). This means that the probing pattern is a periodic stream of 200-byte packets transmitted every 20 milliseconds. And, the transmission rate is 80 Kbps ((40 bytes of IP/UDP/RTP headers + 160 bytes of payload) / 20 ms). The ability to know the probing pattern upon receiving a RTP packet is an important advantage, particularly for adaptive-rate VoIP. This is because the transmission rate of adaptive-rate VoIP constantly changes. Since no initial setup is needed, Sync & Sense can continuously monitor the state of the network, even when the transmission (or the probing pattern) changes.

Figure 3-2 illustrates the dynamics of VoIP packets that traverse the network. When the network is lightly loaded, the packets could experience a very minimal queuing delay. Hence, the packets arrive at the receiver with the same interval gap as the inter-departure gap (or the packetization delay), as shown in Figure 3-2 (a). In this case, the network delay includes only propagation, transmission, and router processing delays. Assuming the end-to-end path does not change over the course of the session, every packet experiences the same network delay. Figure 3-2 (b) shows a condition where at least one link along the path is heavily loaded. The packets experience variable queuing delay in addition to the constant delays of propagation, transmission, and router processing. Given that the sender and receiver have no synchronized clock, the receiver is somewhat limited to obtain the network characteristics. Besides packet loss, the receiver may observe inter-arrival gaps between the incoming packets. Since there is no timing reference that the receiver can use, the receiver is restricted from making any measurements. For example, from Figure 3-2 (b), packet #3 experiences a queuing delay, but inter-arrival gap #3 appears smaller than the inter-departure gap. The receiver may infer a congestion condition, but would not be able to measure the queuing delay.

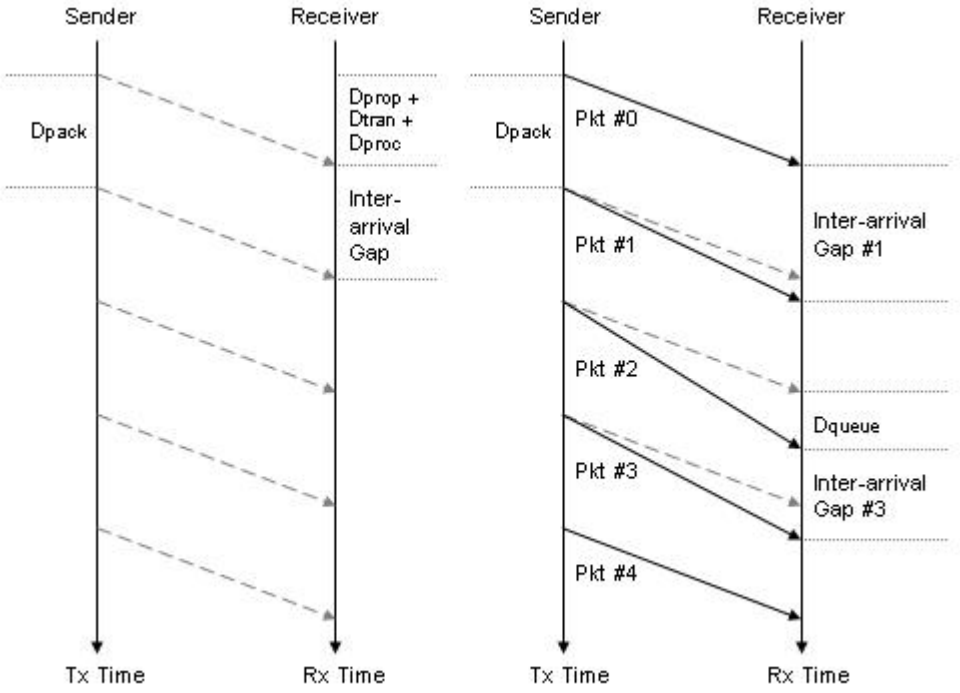

Figure 3-2 Dynamics of VoIP packets when the network is (a) lightly and (b) heavily loaded

Let's assume for now that the first arriving packet experiences no queuing delay. The sensing phase of Sync & Sense works as follows. Sync & Sense learns the packetization delay (or the inter-departure gap) upon receiving a RTP packet, as mentioned earlier. Sync & Sense uses the packetization delay to construct the transmission timing on the receiver. Figure 3-3 illustrates how Sync & Sense observes the incoming packets and makes measurements. The intervals between the dashed lines on the receiver represent the constructed transmission timing. Since the sender and receiver have no synchronized clock, Sync & Sense has no way to measure the constant delays of propagation, transmission, and router processing. The transmission and reception timing are virtually synchronized by which there is an unknown offset of such delays.

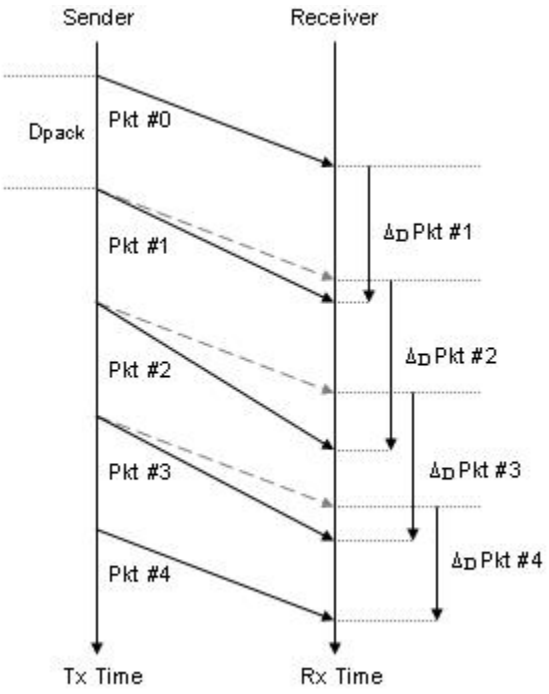

Figure 3-3 Dispersion gaps as measured by Sync & Sense

As shown in the figure, Sync & Sense 'senses' the state of the network by measuring what we call **dispersion gap**, instead of inter-arrival gap. The dispersion gap describes how the inter-departure gap (or packetization delay) has dispersed when the packet arrives at the receiver. The dispersion gap can be written as:

$$\Delta_D = D_{pack} + D_{queue} \qquad (3\text{-}1)$$

The above equation shows that the dispersion is caused by queuing delay. When the network is lightly loaded, the dispersion gap is relatively equal to the packetization delay. As mentioned earlier, once receiving a RTP packet, Sync & Sense knows the packetization delay. Sync & Sense can simply calculate the queuing delay from the above equation. Being able to measure the queuing delay is crucial. This allows Sync & Sense to determine other delay components, as well as to estimate the available network bandwidth. Note that the assumption made above that the first arriving packet has zero queuing delay is unrealistic. In the next section, we describe how Sync & Sense can synchronize the transmission and reception timing when the first arriving packet does experience a queuing delay.

3.2.2 Synchronizing Phase

The Internet traffic is known to have a characteristic of bursty behavior; more specifically, with a long-tail distribution. This can be seen in a large number of empirical and theoretical studies of the Internet traffic, for example in [27, 28, 48]. The implication of the bursty behavior is that, when packets travel across the network, some of those packets may experience a very minimal queuing delay. Based on this implication, Sync & Sense employs a unique technique that can virtually synchronize the transmission and reception timing of the VoIP session. We use Figure 3-4 to demonstrate the synchronizing phase of Sync & Sense. Figure 3-4 (a) shows a typical pattern of one-way network delay when packets travel across the network. Figure 3-4 (b) is the corresponding plot between packet sequence number and arrival time (in milliseconds) as the receiver monitors the arriving packets.

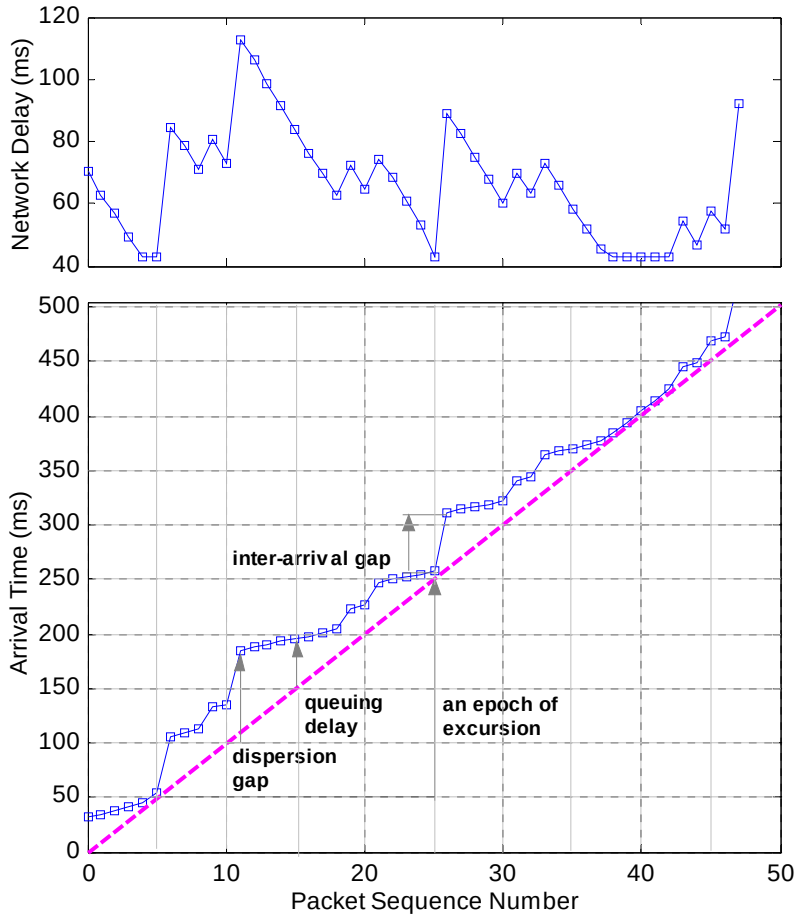

Figure 3-4 (a) Typical pattern of one-way network delay and (b) the corresponding plot as the receiver monitors the arriving packets

Assume the packets in the figure are based on RTP, and are transmitted with a packetization delay of 10 milliseconds. Also, we assume that the end-to-end routing path does not change over the course of the session. That is, every packet experiences the same constant delays of propagation, transmission, and router processing, plus the variable queuing delay. From Figure 3-4 (b), we can draw a diagonal baseline, given at least two lowest points in the arrival time. This diagonal baseline represents the transmission timing because the packets are transmitted as a periodic stream. For example, packet #5 and #10 are transmitted at time 50 and 100 milliseconds, respectively. As the observations of the packet arrival time represent the reception timing, Figure 3-4 (b) shows that both the transmission and reception timing are synchronized on the same time scale. It is, however, a virtual synchronization because it ignores the unknown constant latency of the forwarding path, which includes propagation, transmission, and router processing delays. This synchronization allows Sync & Sense to make measurements of the queuing delay. From the figure, the points (or packet sequence numbers) in which their arrival times are on the baseline indicate that the packets experience zero queuing delay. The points in which their arrival times are above the baseline indicate that the packets experience a queuing delay. For a given sequence number, the queuing delay can be measured, which is equal to the time difference between the arrival time and the baseline.

Although it may look easy to draw a baseline for the synchronization as shown in Figure 3-4 (b), algorithmically, it is not simple. This is because the algorithm has to look forward for the incoming packets, having no idea when it will be able to draw the baseline. Below, we describe how the Sync & Sense algorithm searches and constructs such a baseline. As Sync & Sense monitors the arriving packets, it observes the inter-arrival gaps and uses them to construct an epoch of excursion of queuing delay. Consider packets #5 to #25 in the figure, the epoch begins with a packet having a very minimal queuing delay, followed by several packets having variable queuing delays, and ends with a packet having a very minimal queuing delay. We denote here a valid complete **epoch of excursion of queuing delay** (E) is the time from an arriving packet with a very minimal queuing delay to the next arriving packet with a very minimal queuing delay. Let the first arriving packet be packet #0 and the following arriving packets be labeled packet #*i*, where $i = 1, 2, 3, \ldots, n$. Accordingly, *i*

also denotes the number of inter-arrival gaps. An ongoing epoch of excursion can be found as the sum of inter-arrival gaps (Δ_A), or

$$E = \sum_{i=1}^{n} \Delta_A^i \quad (3\text{-}2)$$

Assume, for now, that the first arriving packet experiences a very minimal queuing delay. The baseline for the transmission timing can be constructed when a valid complete epoch of excursion is found. That is, as illustrated in Figure 3-4 (b), given two arriving packets with a very minimal queuing delay, the baseline can be drawn. Once constructed, the baseline is also extendable for use with the future arriving packets. Algorithmically, an ongoing epoch of excursion is found complete when the following condition is satisfied:

$$nD_{pack} - \varepsilon \leq E \leq nD_{pack} \quad (3\text{-}3)$$

The above condition basically means that a complete epoch of excursion must be a multiple of packetization delays. This can also be seen in Figure 3-3, which shows a valid complete epoch. As a matter of fact, the probability of a packet having a very minimal queuing delay, when traversing the network, is very low. Rather, the arriving packets would at least have a minimal queuing delay. In order to compensate for such a minimal queuing delay, we need to allow an error margin (ε) in Equation 3-3. Specifically, a complete epoch of excursion is allowed to be a little bit smaller than a multiple of packetization delays.

Our simulation study of the Sync & Sense algorithm shows the error margin parameter, as small as one millisecond, can play an important role in the Sync & Sense performance. With the error margin, the baseline for the transmission timing gets a little bit larger, like a band. Thus, more of the arriving packets with a minimal queuing delay can be on the baseline. This makes it easier for Sync & Sense to find a complete epoch of excursion, which helps to speed up the synchronization. In addition, using the error margin causes epochs of excursion to be shortened. As a result, Sync & Sense can find more of the complete epochs. This allows Sync & Sense to reaffirm the synchronization more often, which helps to ensure accurate measurements. Note that we could also use the error margin

such that a complete epoch of excursion is allowed to be a little bit larger than a multiple of packetization delays (or $E \leq nD_{pack} + \varepsilon$). Although this helps to catch more of the arriving packets with a minimal queuing delay as well, it may allow the measurement error to accumulate. When we only allow a complete epoch to be a little bit smaller than a multiple of packetization delays (or $nD_{pack} - \varepsilon \leq E$), we restrict measurement error accumulation. If a complete epoch is found that is smaller than a multiple of packetization delays, Sync & Sense adjust the transmission timing to a new lower baseline. Over time, such an adjustment can lower the baseline even more, which helps to reduce the measurement error to the minimal.

As mentioned above, the condition for a complete epoch of excursion (Equation 3-3) allows Sync & Sense to construct the synchronized timing. However, a complete epoch may be found to be invalid, specifically if the first arriving packet does not actually have a very minimal queuing delay. For example, from Figure 3-4 (b), consider if the first arriving packet is packet #10. When packet #16 arrives, a complete epoch can be observed, but it is invalid. This results in a false baseline of the transmission timing, and false measurements for the arriving packets that follow. This kind of situation can be prevented. Besides Equation 3-3, extra conditions are needed to ensure that a complete epoch of excursion is found valid. Sync & Sense actually declares the synchronization when the following conditions are satisfied:

- A complete epoch of excursion must comprise at least three inter-arrival gaps.
- Sync & Sense finds two consecutive epochs of excursion.

When the network is loaded, packets usually arrive at the receiver unequally spaced. It is possible that, by chance, a couple of arriving packets may have their inter-arrival gaps equal to the packetization delay. This may be observed as a complete epoch of excursion, but it is not valid. The former condition ensures that a complete epoch is long enough, which can eliminate the probability of encountering such a problem. Similarly, it is possible that a complete epoch of excursion, which is not valid, may happen accidentally. By looking for two consecutive epochs, the latter condition dramatically reduces the probability of observing invalid epochs of excursion. Since network delay is highly variable, it is considerably less likely that two consecutive complete epochs of excursion may happen by chance; unless such complete epochs are valid. Our simulation study shows that the above strategy works

remarkably well. When the synchronization is declared, the following observed epochs allow Sync & Sense to reaffirm the synchronization over the course of the session.

Note that the synchronizing process is not affected by packet loss. From Figure 3-3, if packet #2 is lost, upon receiving packet #3, Sync & Sense can observe the inter-arrival gap that begins from the arrival of packet #1. Hence, this observed inter-arrival gap accounts for both packet #2 and #3. By checking the packet sequence number, Sync & Sense knows that it must advance the transmission timing accordingly by two packetization delays. This can also be seen in Equation 3-2. Note that i is counted based on the packet sequence number. In the event of packet loss, the observed inter-arrival gap (Δ_A) includes more than one of the individual inter-arrival gaps. Thus, i must advance in accordance to the number of lost packets. Therefore, once the synchronization is declared, the measurements can go on without interruption from the event of packet loss.

A practical issue for VoIP is silence suppression, which is a process of not transmitting the actual voice packets when one of the parties on a call is not speaking. Silence suppression allows bandwidth savings because, typically, one party in a conversation speaks at any one time. Silence suppression works by which the sender transmits comfort noise packets when silence periods are detected. RFC 3389 [29] describes RTP payload format for comfort noise. The comfort noise payload, also known as a Silence Insertion Descriptor (SID) frame, includes a description of the noise level. The receiver uses this SID frame to drive the Comfort Noise Generator (CNG) and produce an appropriate amount of comfort noise. As mentioned earlier, the synchronizing phase of Sync & Sense relies on observing the periodic stream of incoming packets. Enabling silence suppression on the VoIP session could affect Sync & Sense if the comfort noise packets are not transmitted in a periodic manner. Sync & Sense would need to resynchronize the timing every time the voice packets are resumed, which could significantly degrade the synchronization performance. However, the transmission rate of the comfort noise packets is implementation specific. To ensure continuous synchronization for Sync & Sense, the comfort noise packets must be transmitted periodically in the same rate as the voice packets.

We assume earlier that the first arriving packet has a very minimal queuing delay and, when a valid complete epoch of excursion is found, the baseline for the transmission timing can be constructed. But, we really cannot assume this. The remaining issue is how Sync &

Sense determines whether or not the first arriving packet has a very minimal queuing delay, as for the beginning of the epoch. Initially, Sync & Sense assumes the first arriving packet has a very minimal queuing delay, even it actually does not. When the following packets arrive, Sync & Sense starts monitoring the ongoing epoch of excursion, using Equation 3-2. For each arriving packet, Sync & Sense computes the dispersion gap, which can be written as:

$$\Delta_D = E - (n-1)D_{pack} \quad (3\text{-}4)$$

$$= \sum_{i=1}^{n} \Delta_A^i - (n-1)D_{pack} \quad (3\text{-}5)$$

As we can see from Equation 3-1, when the arriving packet has zero queuing delay, the dispersion gap is equal to the packetization delay. In other words, the dispersion gap must be equal or larger than the packetization delay. Sync & Sense uses this fact to prove whether or not the assumption, that the first arriving packet has a very minimal queuing delay, is correct. If it is true, we will never find a dispersion gap smaller than the packetization delay. That is, in Figure 3-4 (b), the first arriving packet is on the diagonal baseline. Specifically, Sync & Sense uses the following condition to test each arriving packet:

$$\Delta_D \geq D_{pack} - \varepsilon \quad (3\text{-}6)$$

The above condition must hold for all arriving packets. When an arriving packet's dispersion gap violates Equation 3-6, it means that the assumption is wrong: the first arriving packet does not have a very minimal queuing delay. Sync & Sense abandons the ongoing epoch and starts the synchronizing process over again. The last arriving packet is assumed to have the minimal queuing delay and the new epoch begins. If the assumption is correct, Equation 3-6 will never be violated, the ongoing epoch will be found complete eventually. For the same reason as in Equation 3-3, the error margin (ε) is allowed in Equation 3-6.

Figure 3-5 shows the flow chart of the Sync & Sense algorithm. Table 3-1 demonstrates the synchronization process of Sync & Sense. The error margin used in this example is one millisecond. The data in the table is in accordance with Figure 3-4 (b).

Assume the first arriving packet is packet #10, which does not experience a very minimal queuing delay. The alternating background colors in the table represent the observed epochs of excursion.

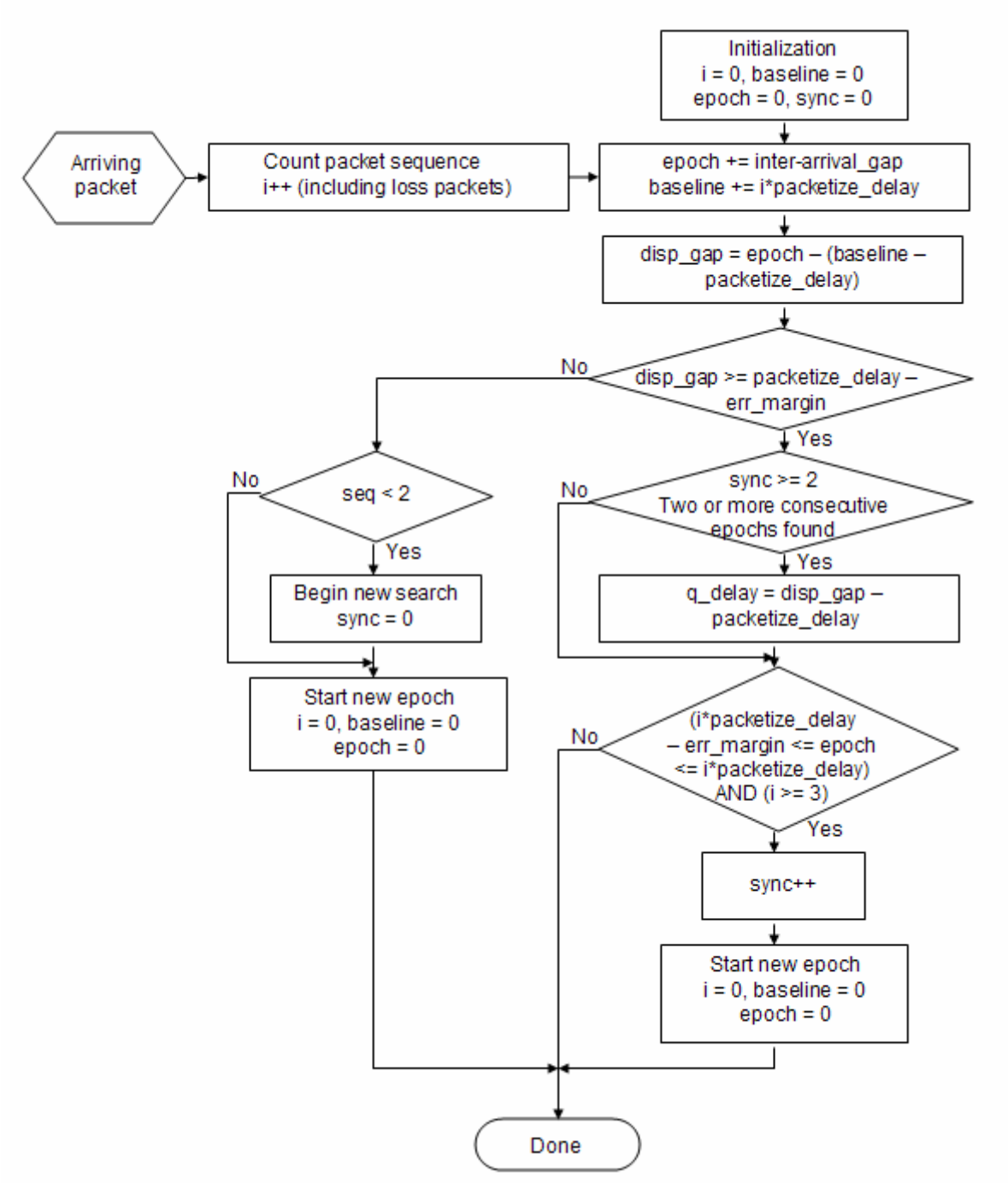

Figure 3-5 Flow chart of the Sync & Sense algorithm

Table 3-1 Demonstration of the synchronization process of Sync & Sense

Packet #	Inter-arrival Gap (Δ_A)	Epoch (E)	Dispersion Gap (Δ_D)	Notes
10	0	0	0	Start new epoch
11	49.375	49.375	49.375	
12	3.750	53.125	43.125	
13	2.500	55.625	35.625	
14	2.500	58.125	28.125	
15	2.500	60.625	20.625	
16	2.500	63.125	13.125	
17	3.750	66.875	6.875	Eq.3-6 violated; Start new epoch
18	2.500	2.500	2.500	Eq.3-6 violated; Start new epoch
19	19.687	19.687	19.687	
20	2.500	22.187	12.187	
21	19.688	41.875	21.875	
22	3.750	45.625	15.625	
23	2.500	48.125	8.125	Eq.3-6 violated; Start new epoch
24	2.500	2.500	2.500	Eq.3-6 violated; Start new epoch
25	2.500	2.500	2.500	Eq.3-6 violated; Start new epoch
26	53.178	53.178	53.178	
27	3.750	56.928	46.928	
28	2.500	59.428	39.428	
29	2.500	61.928	31.928	
30	2.500	64.428	24.428	
31	19.688	84.116	34.116	
32	3.750	87.866	27.866	
33	19.687	107.553	37.553	
34	2.500	110.053	30.053	
35	2.500	112.553	22.553	
36	3.750	116.303	16.303	
37	3.750	120.053	10.053	Eq.3-3 satisfied; Valid complete epoch
38	9.8051	9.8051	9.8051	
39	10.000	19.8051	9.8051	
40	10.000	29.8051	9.8051	Eq.3-3 satisfied; Valid complete epoch; Synchronization declared

3.3 Performance Study

Performance is the most important issue for Sync & Sense. Because Sync & Sense needs to spend some time to construct the synchronized timing before making measurements, the question needed to be answered is how long it takes for Sync & Sense to get synchronized. Another performance question is how accurate the measurements made by Sync & Sense are. In the following, we conduct a simulation study that demonstrates and evaluates the performance of Sync & Sense. The error margin is an important parameter in the Sync & Sense algorithm. The study also attempts to determine the optimal value of the error margin that gives the best performance. In addition, we study the factors that can affect the performance of Sync & Sense.

3.3.1 Simulation Setup

The performance study is conducted using the Network Simulator 2 or ns-2 [12]. The simulations that follow are based on the network topology in Figure 3-6. The number of nodes varies as this is a factor under study. All nodes implement FIFO scheduling and drop-tail queuing. All links, except the last hop link, have capacity of 10 Mbps. The last hop link is set to have a limited capacity of 1.544 Mbps, so as to create the bottleneck. The propagation delay from the sender to the receiver is fixed at 40 milliseconds, regardless of the number of hops. In other words, the sum of propagation delays from all the links is always equal to 40 milliseconds. To make the simulation more realistic, each link has a fair amount of cross traffic, which accounts for 60 percent load on average. This level of cross traffic load also introduces a decent amount of packet loss. The cross traffic is generated as Long Range Dependent (LRD) by using nine Pareto sources with the shape parameter (α) set to 1.5. Packet sizes of the cross traffic are distributed following the studies in [14, 15]; 60% of the packets are 40 bytes, 25% are 550 bytes, and 15% are 1500 bytes. Note that, in terms of load distribution, about 7% of the load is 40-byte packets, 35% is 550-byte packets, and 58% is 1500-byte packets. The Sync & Sense algorithm is implemented at the VoIP receiver. The VoIP flow is based on the ADPCM codec, with the packetization delay of 10 milliseconds.

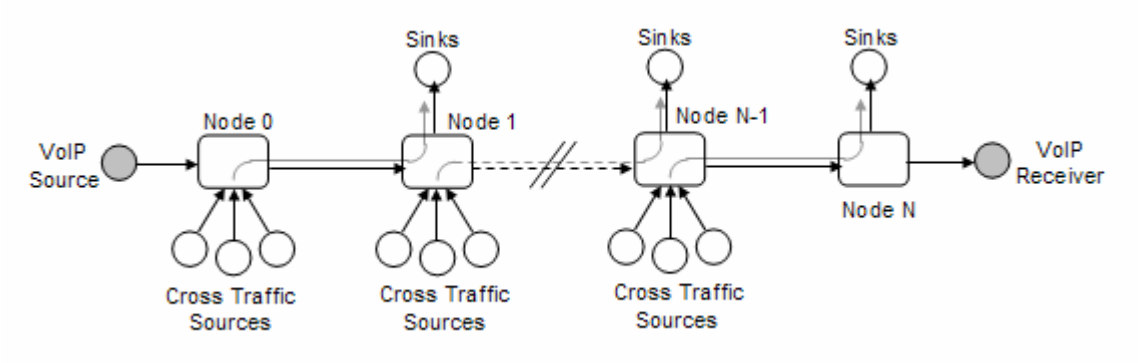

Figure 3-6 Simulation network topology

3.3.2 Sync & Sense Demonstration

The objective of this section is to demonstrate the performance of Sync & Sense. We closely monitor the cross traffic on the bottleneck link, and accordingly examine how well Sync & Sense makes measurements of the queuing delay. Figure 3-7 shows the results from an instance of the simulations, where the network topology consists of 6 hops. In this simulation, the error margin parameter of Sync & Sense is set to one millisecond. Figure 3-7 (c) shows the actual one-way network delay of the VoIP packets that arrive at the receiver. Accordingly, Figure 3-7 (b) shows the packets lost in the network. This simulation environment allows us to easily obtain the actual queuing delay of each individual packet since the measurements are made within the same time domain. In order to evaluate the accuracy of Sync & Sense, we present the result in the form of measurement error, as shown in Figure 3-7 (a), which is the difference between the queuing delay measured by Sync & Sense and the actual queuing delay.

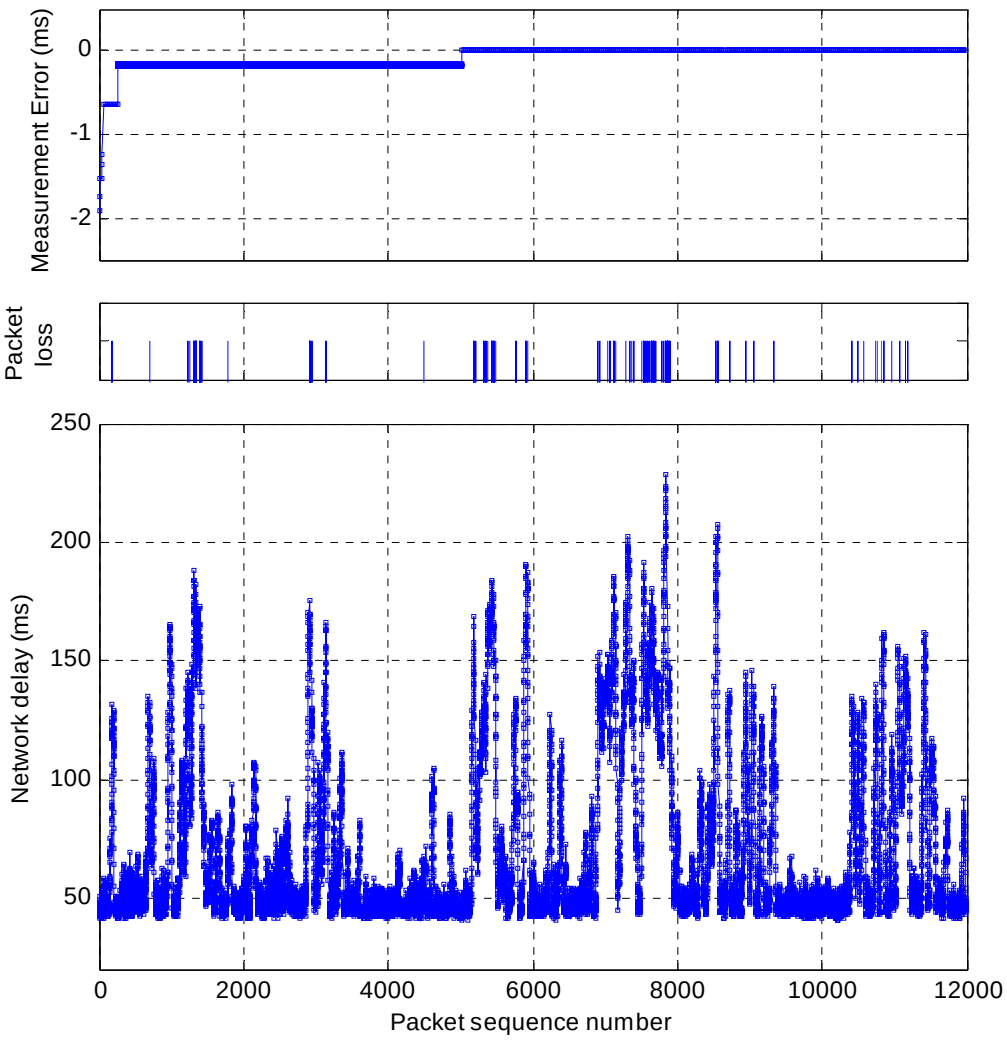

Figure 3-7 Sync & Sense demonstration

The figure demonstrates that Sync & Sense can provide impressive measurement accuracy. The measurement error is, in overall, less than 0.25 milliseconds, given that packet loss is persistent and the network delay is highly variable as high as hundreds of milliseconds. Negative error means that Sync & Sense underestimates the measurements. This is because the baseline for the transmission timing is constructed using the arriving packets with a minimal, but not actually zero, queuing delay. For the very first arriving packets, the measurements have error up to 2 milliseconds. Our examination shows that this is a direct result of the error margin parameter. Using a large error margin allows a wider error in the measurements, but Sync & Sense can get synchronized quicker, within tens of the arriving

packets. As seen from the figure, the measurement errors get narrower as time goes by. This is because, when Sync & Sense observes more arriving packets, it can find new baselines for the transmission timing with smaller and smaller error. Specifically, when a dispersion gap is found less than the packetization delay (violating Equation 3-6), such a dispersion gap constitutes a better and more accurate baseline. Eventually, no dispersion gap ever violates Equation 3-6 again. That is, the baseline has reached the bottom that reflects the actual transmission timing.

3.3.3 Contributing Factors to the Performance

The synchronization process of Sync & Sense relies on the assumption that some arriving packets may experience a very minimal queuing delay when traveling from the sender to the receiver. This allows Sync & Sense to construct the synchronized timing and make measurements. Factors that could impact this assumption include the number of hops between the sender and receiver and the underlying cross traffic. As the number of hops increases, the chance that the arriving packets experience a very minimal queuing delay decreases. In other words, Sync & Sense would observe fewer arriving packets with a very minimal queuing delay. This could potentially affect the performance of Sync & Sense. Besides the number of hops, the error margin parameter is another factor that has a direct impact on the performance. Using a large error margin helps Sync & Sense to get synchronized easier and faster, but incurs a higher error in the measurements. On the contrary, using a small error margin ensures minimal measurement error, but it could take a long time to get synchronized. In the following, we conduct an extensive simulation study to examine these contributing factors to the performance of Sync & Sense. We also attempt to determine the optimal value of the error margin parameter that could overall yield the best performance.

The simulation setup is based on the network topology in Figure 3-6. The number of hops between the sender and receiver is the factor under study. We repeat the simulation by modifying the network topology with increasing number of hops from 2, to 4, 6, 8, and 10, respectively. Each hop link has an offered load of 60 percent on average, which introduces an observable amount of packet loss. This ensures that each link is fairly congested. In order to study the error margin factor, the Sync & Sense algorithm is modified to produce multiple

copies of the measurements, based on different error margin values. The value of the error margin ranges from zero to 3 milliseconds, with the increment step of 0.1 milliseconds. Each simulation runs for 120 seconds and is repeated 100 times with different random cross traffic.

Two performance metrics are used to evaluate Sync & Sense: synchronization and accuracy. In the synchronization metric, we consider the number of packets leading to synchronization, which determines how fast Sync & Sense can get synchronized and start to provide the measurements. Besides the initial synchronization, it is possible that Sync & Sense may lose its synchronization and needs to reconstruct the synchronized timing. During the course of the session, some of the arriving packets may be in the synchronizing phase. Hence, we also consider synchronization rate, which refers to the percentage of the arriving packets in which the measurements can be made. The accuracy metric refers to the measurements of queuing delay made by Sync & Sense, in comparison to the actual queuing delay obtained by the simulation timestamps. In other words, we use measurement error to characterize the accuracy. Care is especially taken when considering measurement error. The conventional method of using the mean of measurement errors to assess the accuracy could be misleading. A considerable number of small errors could weigh down or 'hide' some large errors in the measurements. Instead, we use the percentile of the *x*-ms accuracy. For example, a reading of 99^{th} percentile of the 1-ms accuracy means that 99 percent of all the measurements are accurate with possible errors of 1 millisecond or less. This method is more effective and can truly assess the accuracy of Sync & Sense. In addition, we also consider the maximum error, which reveals the worst case scenario of error in the measurements.

Figure 3-8, 3-9, 3-10, 3-11, and 3-12 are the simulation results when the number of hops between sender and receiver are 2, 4, 6, 8, and 10, respectively. Each figure includes the plots of performance metrics versus the error margin parameter: (a) synchronization rate, (b) number of packets leading to synchronization, (c) maximum error, and (d) percentile of the *x*-ms accuracy. The plot of the number of packets leading to synchronization is displayed with the 0.5 (median) and 0.8 percentile, so as to reflect its long-tail distribution. As we have seen from Figure 3-7, the measurement error is caused by underestimation. The plot of maximum error and percentile of the *x*-ms accuracy is displayed with absolute measurement error, with no negative sign. The plot of the percentile of the *x*-ms accuracy includes 1-ms and 0.1-ms benchmarks, with the addition of 10-μs and 1-μs for higher precision.

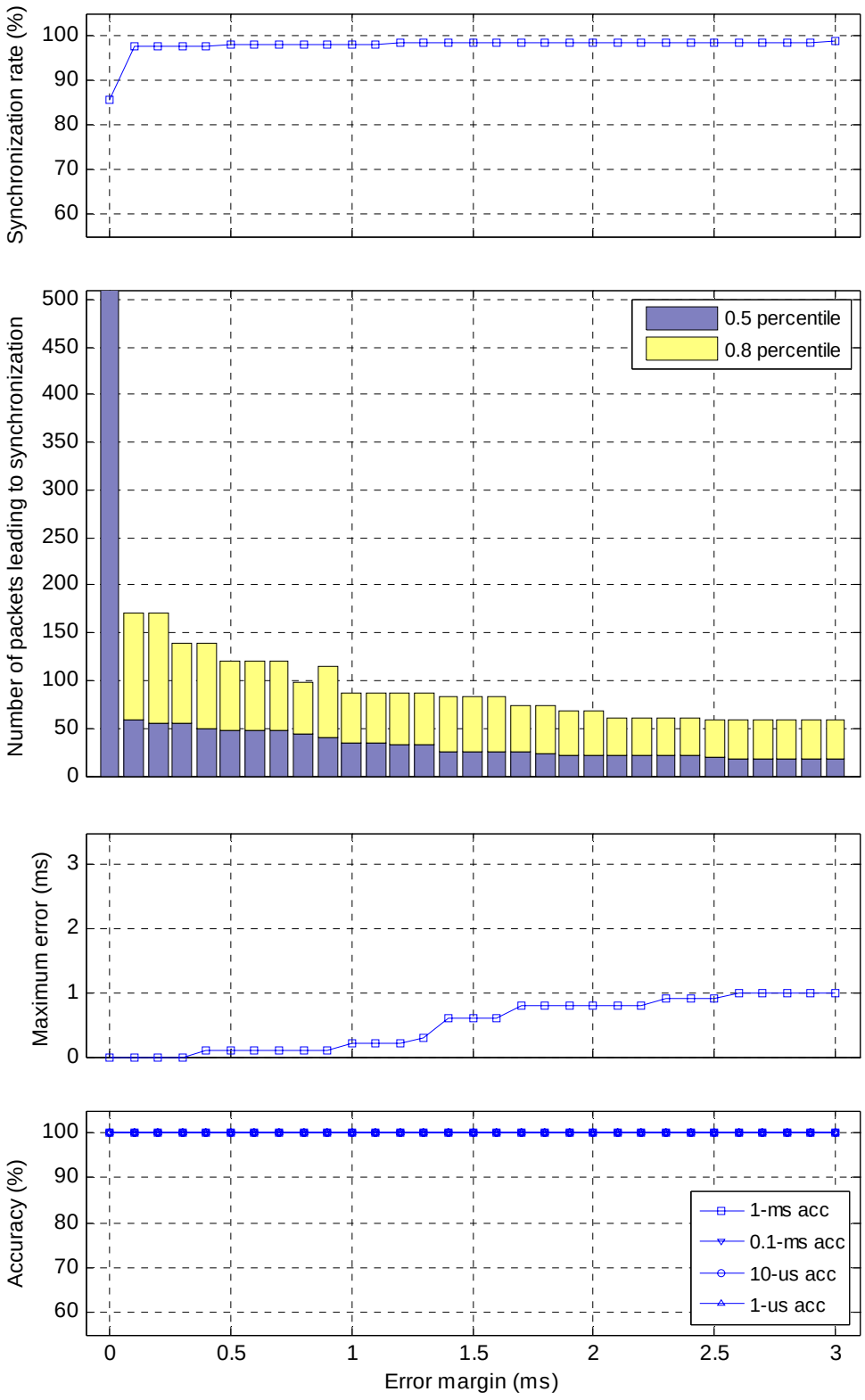

Figure 3-8 Performance of Sync & Sense when the number of hops is 2

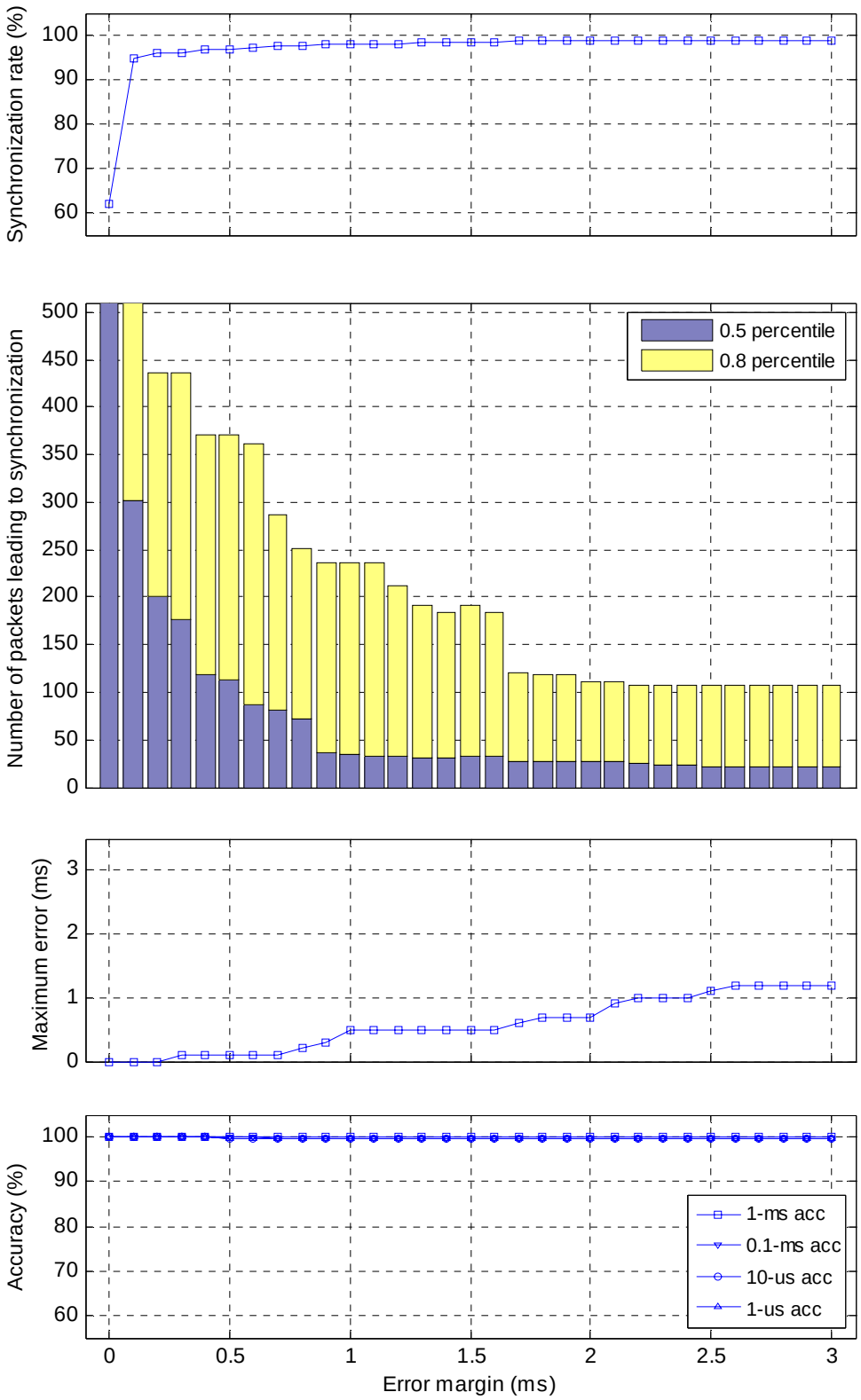

Figure 3-9 Performance of Sync & Sense when the number of hops is 4

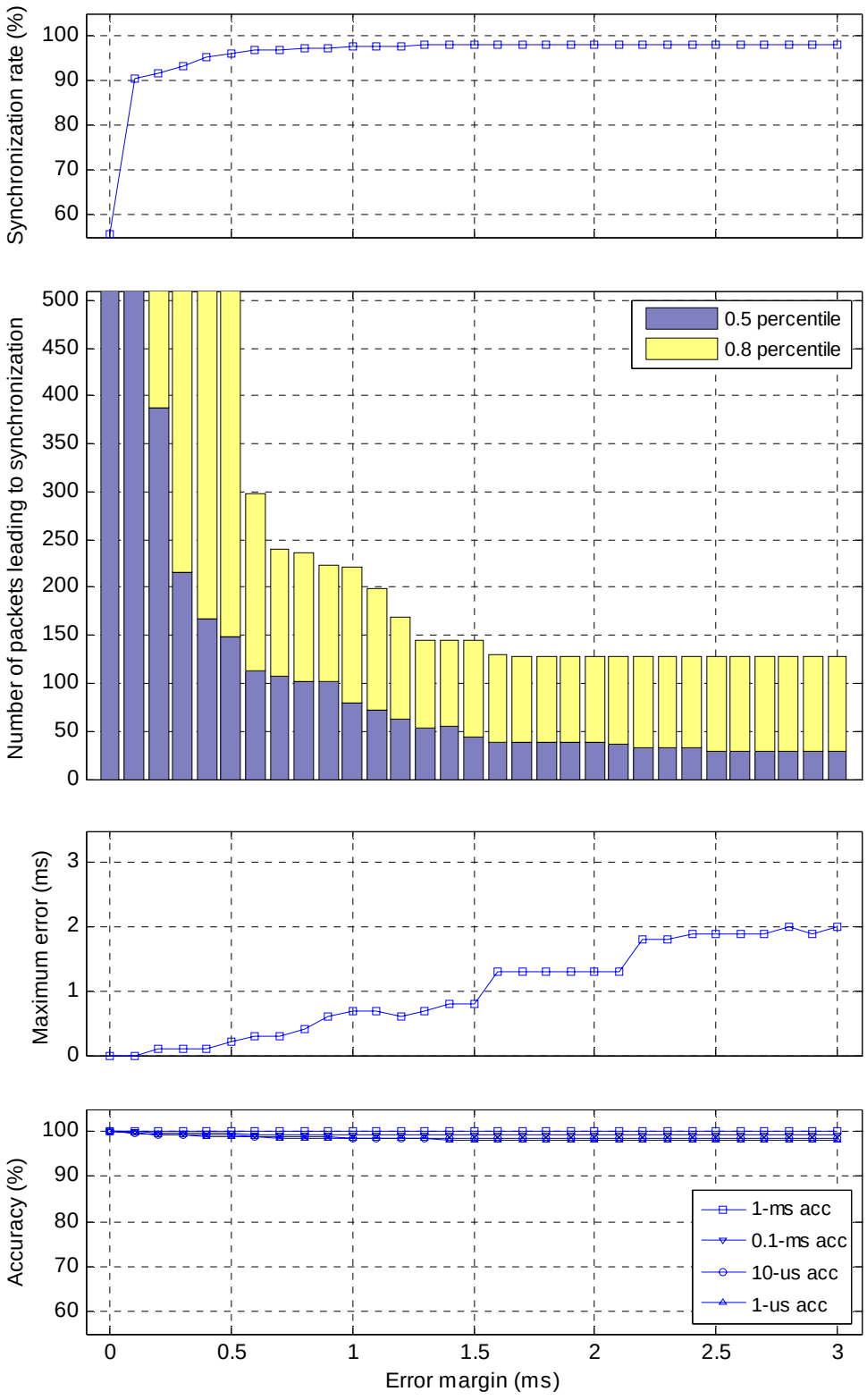

Figure 3-10 Performance of Sync & Sense when the number of hops is 6

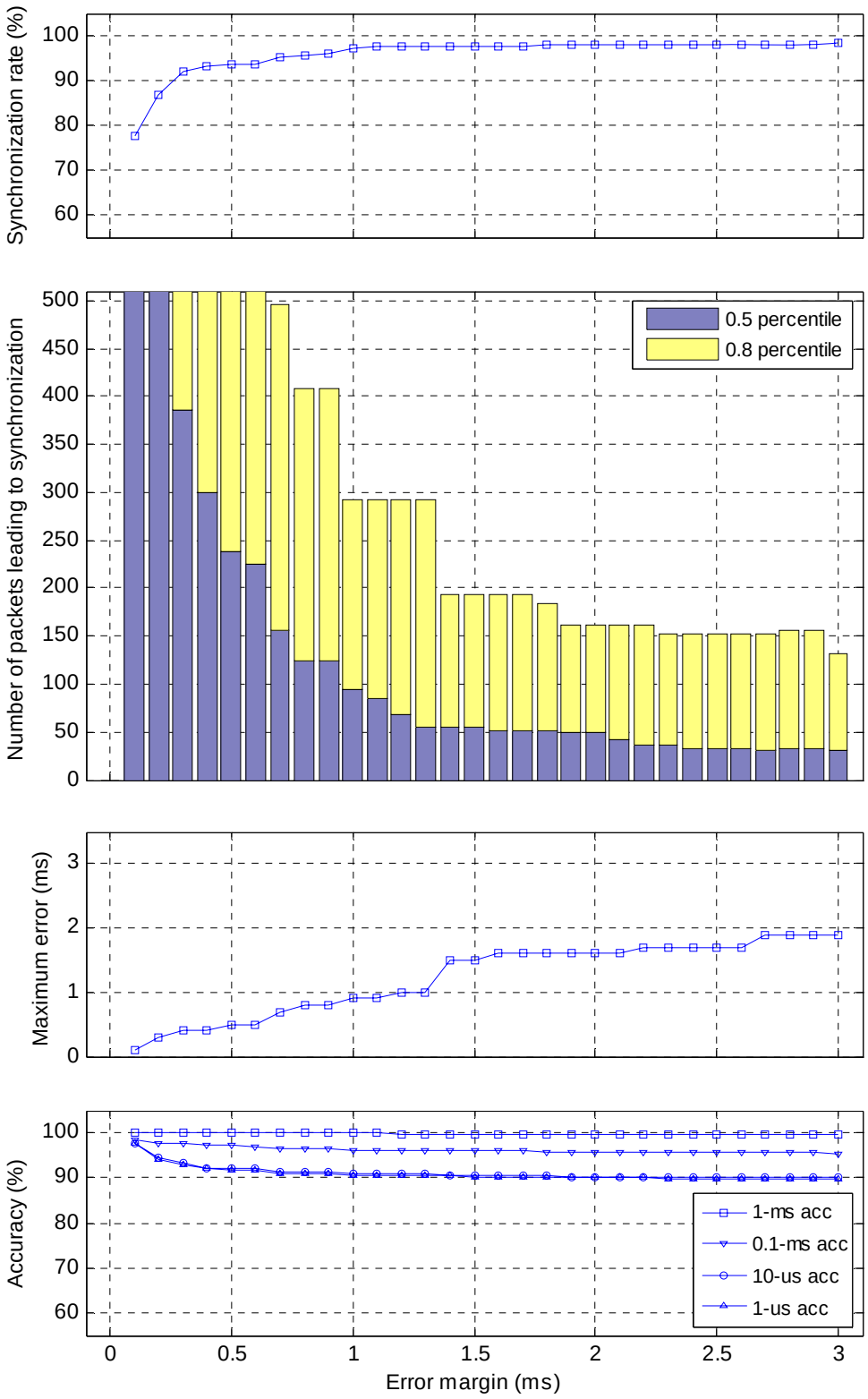

Figure 3-11 Performance of Sync & Sense when the number of hops is 8

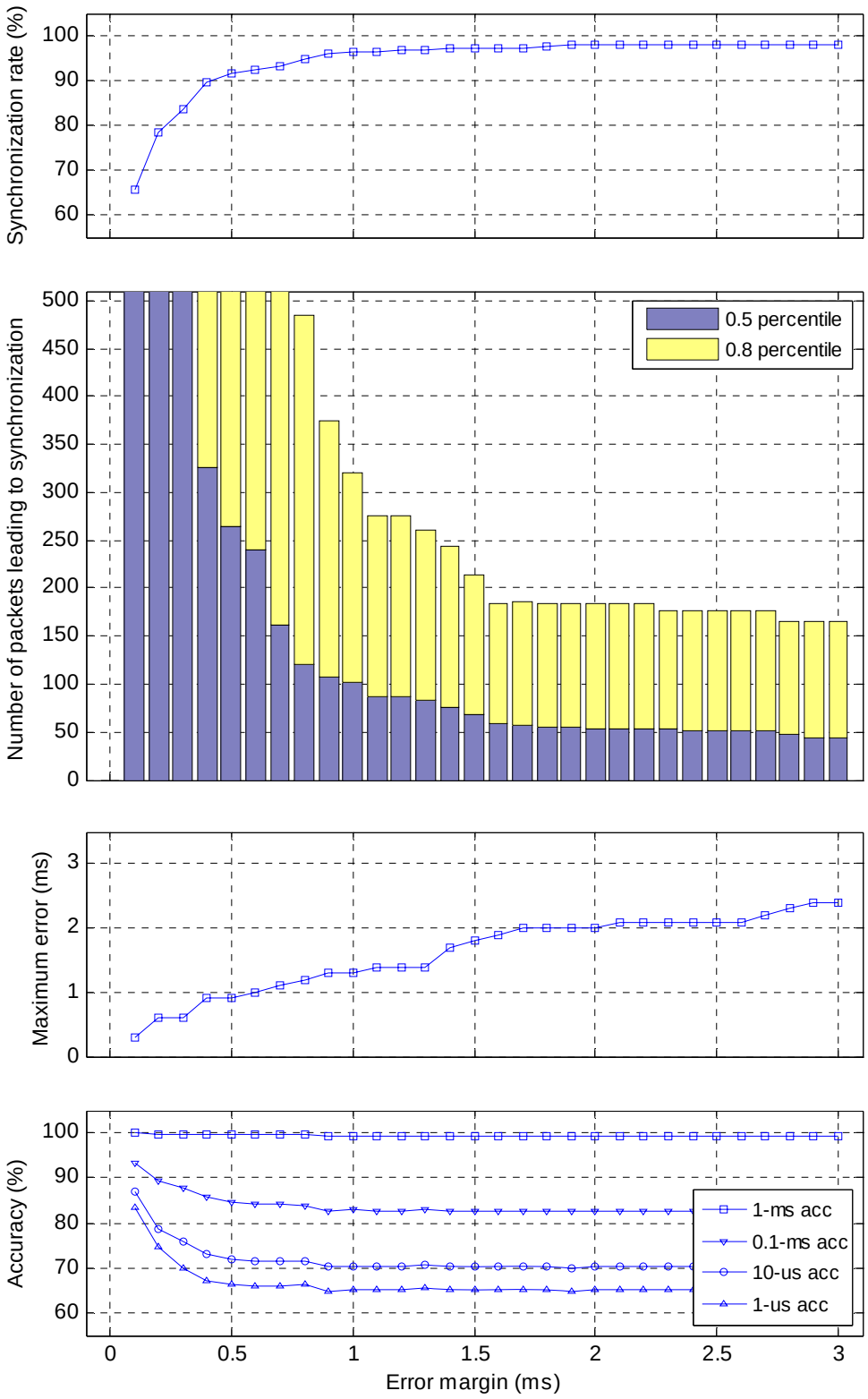

Figure 3-12 Performance of Sync & Sense when the number of hops is 10

From the figures, the number of hops between sender and receiver appears to have little impact on the synchronization rate as long as the error margin parameter is not too small. Sync & Sense can achieve a synchronization rate as high as 98 percent, even when we increase the number of hops up to 10 hops. The impact of the number of hops is more evident when we use a small value of error margin. It can be seen from the figures that, when the error margin is less than 0.5 milliseconds, the synchronization rate drops significantly when the number of hops increases. This corresponds to the plot of the number of packets leading to synchronization, which shows that more of the arriving packets are needed for Sync & Sense to get synchronized. Note that, in order to provide a clear view of the plots of the number of packets leading to synchronization, we limit the display of 500 packets at maximum. Not shown in the plots, when using a small error margin, the median of the number of packets leading to synchronization could be as high as a few thousand packets, particularly when the number of hops is large.

The results show that the error margin parameter plays an important role in determining the synchronization performance. As seen from the figures, using a zero value of error margin is highly ineffective. Ideally, the Sync & Sense algorithm is based on the assumption that some of the arriving packets experience zero queuing delay when traveling from the sender to the receiver. But, in practice when the number of hops increases, the chance that the arriving packets experience zero queuing delay would be extremely small. Using a small value of the error margin helps the synchronization performance considerably. This is because, not only the arriving packets with zero queuing delay, but also the arriving packets with a minimal queuing delay can be used to construct the synchronized timing. The plot of the number of packets leading to synchronization reflects the time period in which Sync & Sense needs to wait for an adequate number of the arriving packets with a minimal queuing delay for the synchronization. As seen from the figures, the number of packets leading to packetization drops significantly when we increase the value of the error margin.

In terms of the accuracy performance, the simulation results show that Sync & Sense can provide impressive measurement accuracy, even when we increase the number of hops up to 10 hops. As seen from the figures, 99.9 percent of the measurements are accurate within a possible error of 1 millisecond or less. The impact of the number of hops can be seen at the maximum error and a higher precision of accuracy (e.g. less than 1 millisecond).

It can be seen from the figures that, when the number of hops increases, the maximum error increases, as well as the accuracy of the higher precision decreases.

The error margin parameter also plays a key role in the accuracy performance. By using a small value of the error margin, we choose to use only the arriving packets with a minimal queuing delay to construct the synchronized timing. Thus, this limits the error to the minimal. As we have learned earlier, using a relatively larger value of the error margin is necessary to achieve higher synchronization performance. This allows Sync & Sense to use the arriving packets, not only with a minimal queuing delay, but also a larger minimal queuing delay, to construct the synchronized timing. As a consequence, the disadvantage is accepting more error in the measurements. As seen from the figures, an increasing error margin has a direct impact on the maximum error.

A conclusion is that the error margin parameter is the controlling factor to the performance of Sync & Sense. An increasing number of hops between sender and receiver affects the synchronization performance such that it causes more of the packets to arrive with a larger minimal queuing delay. Using a relatively larger value of the error margin allows Sync & Sense to use those arriving packets with a larger minimal queuing delay to construct the synchronized timing. This, as a result, helps to increase the synchronization performance. The maximum synchronization performance can be achieved when we use an appropriate error margin value that matches those arriving packets with a minimal queuing delay. However, using a larger error margin value accepts more error in the measurements. Because the error margin parameter has both positive impact on the synchronization performance and negative impact on the accuracy performance, a compromise value of error margin should be chosen in order to optimize the performance between synchronization and accuracy. In this simulation study, given up to 10 hops, the error margin parameter of 1.5 milliseconds is the smallest value that allows minimal measurement error, while ensuring the highest possible synchronization. In reality, the actual number of hops is usually unknown. Besides, the underlying cross traffic is unknown and variable. It would be safer to use a relatively large value of the error margin. This study also shows that the negative impact to the accuracy is somewhat contained. As we can see in the case of 10 hops, even with the error margin of 3 milliseconds, 99.9 percent at 1-ms accuracy is still achievable, with limited maximum error. Hence, a value of the error margin within the range 1.5 to 3 milliseconds is recommended.

3.4 Network State Detection

Given no support from the network, protocols and applications rely on the end-to-end approach to detect the state of the network. Because endpoints are not typically time synchronized, the only observations that are commonly available to infer the state of the network are limited to packet loss and round-trip delay. Such observations work well with data applications because delay is not a significant issue. On the contrary, VoIP is highly sensitive to delay, particularly one-way network delay. The problem is that one-way network delay cannot be measured if the endpoints are not time synchronized. The Sync & Sense methodology solves this problem. Sync & Sense has the ability to measure one-way network delay, more specifically the variable queuing delay component, of the packets traveling from the sender to the receiver. Knowing the queuing delay component is very crucial because it allows us to connect the dots of the other delay components. The result is that Sync & Sense can offer the full spectrum of metrics to detect the state of the network. Namely, besides the commonly used packet loss, Sync & Sense can estimate the available network bandwidth and assess all the delay components in VoIP. Sync & Sense offers tremendous benefits to VoIP applications. The extended information about the state of the network allows adaptive-rate VoIP to make a better control decision to optimize performance. A VoIP agent, including routers and gateways, could use this extended information to compare alternate routes and choose the optimal one for transmitting packets. Because the transmitted RTP packets themselves are used as probe packets, the measurements are in-band, which can reflect the quality of the VoIP session. The extended information can also be used for management purposes, for example, to monitor and collect statistics of the routes of the VoIP sessions. In the following, we describe how the Sync & Sense methodology can be applied to obtain a variety of information about the state of the network.

3.4.1 Available Bandwidth Estimation

The Sync & Sense methodology utilizes the RTP packets, transmitted periodically, as the probe packets. Whereas Sync & Sense is implemented on the receiver, it can learn about the pattern of the probe packets without requiring any initial setup. The codec type identified in the RTP packet header and the packet size simply allow Sync & Sense to calculate the

transmission rate. Specifically, transmission rate is equal to the packet size divided by the packetization delay. At the receiver, Sync & Sense monitors the incoming packets and makes measurements of the dispersion gap. While the packetization delay (or inter-departure gap) is the interval in which the packets are transmitted; the dispersion gap describes how the inter-departure gap has dispersed when the packets arrive at the receiver. The dispersion is caused by the effect of the queue and the underlying cross traffic. Hence, the dispersion gap can reflect the characteristic of the network. Similar to calculating the transmission rate, Sync & Sense can calculate the reception rate, which is equal to the packet size divided by the dispersion gap.

As the dispersion gap reflects the network condition, so does the reception rate. When the network is lightly loaded, the packets experience a small delay. The dispersion gap of the arriving packets would be slightly larger than the packetization delay, and the reception rate would be relatively equal to the transmission rate. On the other hand, when the network is congested, the packets experience a large and variable delay. The dispersion gap of the arriving packets would be much larger than the packetization delay, and the reception rate would be much lower than the transmission rate. We can see that, the reception rate provides an estimate of available bandwidth of the network. The arriving packets are the probe packets that sample the network bandwidth. As network congestion increases, the available bandwidth decreases. As a matter of fact, the reception rate can never exceed the transmission rate. Sync & Sense is limited to estimate the available bandwidth relative to the transmission rate.

As mentioned above, the measurement of the reception rate is an effective tool to sense the state of the network. The difference between the reception rate and the transmission rate can be used as a measure of congestion. The lower the reception rate, compared to the transmission rate, the more the congestion. While the reception rate can give an idea of mild to moderate congestion, the observation of packet loss provides an indication of more serious congestion. With both the reception rate and packet loss, Sync & Sense can offer the whole range of detecting the state of the network.

Because the network delay is highly variable, it is usually difficult to make use of raw observations of the reception rate. For a meaningful interpretation, it is necessary to get rid of the spikes and find the average of the reception rate. The exponential smoothing method is a

common and widely used technique that averages and predicts the next value on the basis of a time series of past values. With this method, the moving average of the reception rate can be calculated from the following equation:

$$R^i = \alpha R^{i-1} + (1 - \alpha) \frac{P}{\Delta_D^i} \quad (3\text{-}7)$$

where P is the packet size, including IP/UDP/RTP headers and the payload, Δ_D is the dispersion gap and α is a smoothing factor ($0 < \alpha < 1$). The exponential smoothing method takes into account the past values, but the more distant ones have less weight. The α parameter determines how the weight is distributed. The smaller the value of α, the greater the weight is given to the more recent observations. A small value of α allows the average to quickly reflect rapid changes in the reception rate, but it may be vulnerable to the spikes. On the other hand, a large value of α is better to get rid of the spikes, but the average may be slow to reflect the instantaneous reception rate. The exponential smoothing method is also used in TCP for estimating the average round-trip time. A value for α between 0.8 and 0.9 was suggested in the TCP specification [30], which has proven to spread out the weight efficiently. In the following simulation, we use the value $\alpha = 0.875$, as suggested in Jacobson's classic congestion paper [31], which examines this issue extensively.

Below, we present a simulation result that demonstrates the ability of Sync & Sense to estimate the available bandwidth, or the reception rate. We use the same network topology, with 10 hops, as well as parameters as described in section 3.4.1. The VoIP flow is based on the ADPCM codec, with the packetization delay of 10 milliseconds. So, the transmission rate is 64 Kbps. The Sync & Sense algorithm uses the error margin parameter of 1.5 milliseconds. Figure 3-13 (a) shows the cross traffic on the bottleneck link. The envelope line is the average cross traffic on a larger time scale of one second. The shaded area within the envelope line is the cross traffic on a smaller time scale, which is sometimes bursty beyond the bottleneck link capacity of 1.544 Mbps. Figure 3-13 (c) and (d) shows the packets lost in the network and the actual one-way network delay, respectively. Figure 3-13 (b) shows the moving average of the reception rate as measured by Sync & Sense.

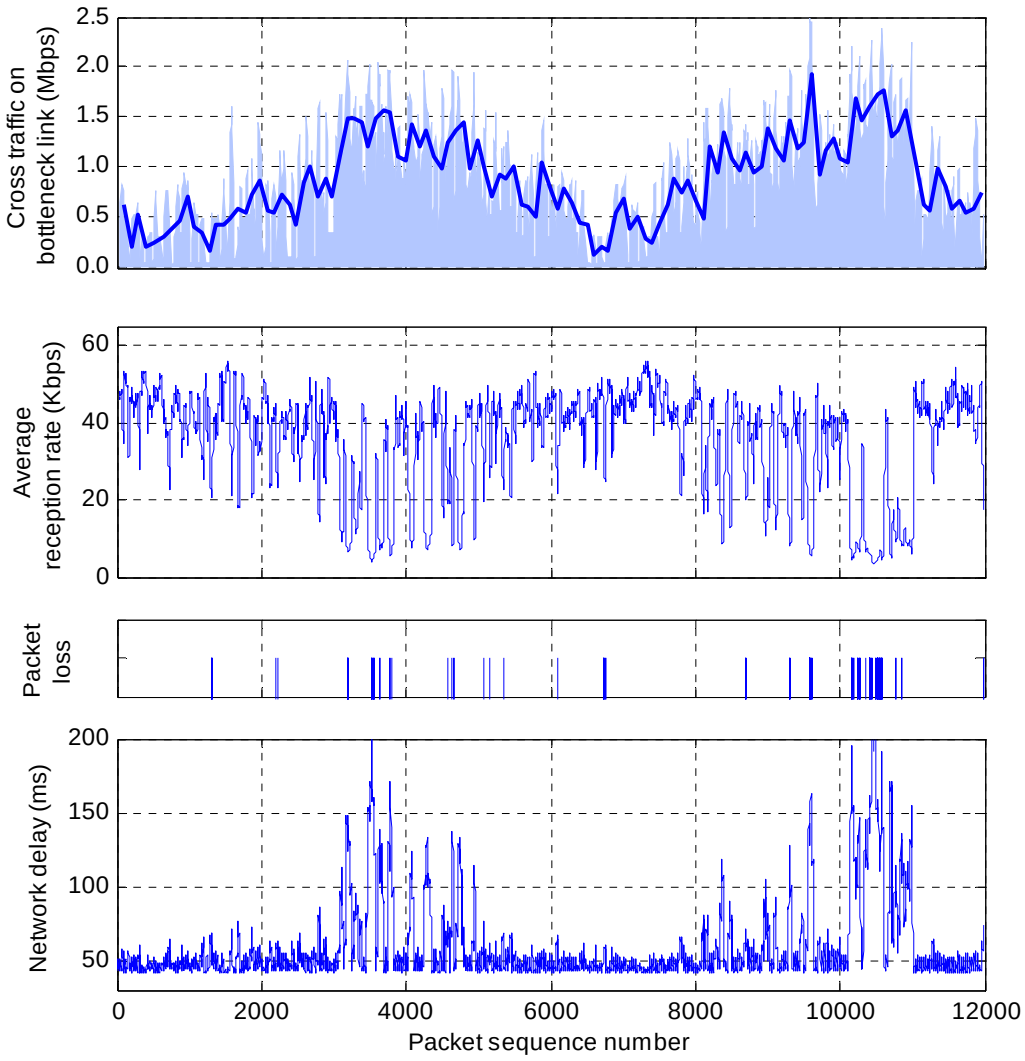

Figure 3-13 Available bandwidth estimation

The figures show that the moving average reception rate is largely a mirror of the cross traffic. In other words, the reception rate reflects the available network bandwidth, as well as the changes in the underlying traffic. When the link utilization is low, the bottleneck link can accommodate the bandwidth required by the VoIP flow. Thus, Sync & Sense observes the reception rate that is fairly close to the transmission rate. Note that the reception rate remains below the transmission rate because of the cross traffic imposed on the other hop links as well. Accordingly, when the cross traffic load increases and the link utilization gets close to its full capacity, Sync & Sense observes that the reception rate drops dramatically. It can be seen from the figures that the average reception rate is a mirror of the

network delay as well, because the increased network delay is basically caused by limited available bandwidth. Consider Figure 3-13 (c), packet losses mostly occur when the bottleneck link is congested; whereas some packet losses may be caused by other hop links. Because Sync & Sense observes both the reception rate and packet loss, it is possible to distinguish the packet losses incurred on the bottleneck link. These packet losses happen when the reception rate drops dramatically. This information can be beneficial to adaptive-rate VoIP, for example, to respond to different packet losses appropriately and efficiently. This simulation demonstrates that the reception rate does not only provide an estimate of the available bandwidth, but is also an effective tool to detect the state of the network.

3.4.2 Propagation Delay Estimation and Delay Assessment

The main objective of this section is to demonstrate how the Sync & Sense methodology can be applied to estimate the propagation delay, without requiring synchronization between the sender and receiver. Besides, we also go through different types of delay. This allows us to demonstrate that Sync & Sense can enable an agent or application to be able to assess all the delay components in VoIP. We begin with network delay, which consists of four components: transmission, router processing, queuing and propagation delay. Router processing delay is the time it takes for the router to prepare the packet for delivery. Router processing delay is normally insignificant because it is very small relative to the other delays. So, router processing delay can be negligible in the calculation. Transmission or serialization delay is the time required to put all the bits in the packet for transmission. It is equal to the packet size divided by the transmission link capacity. In a high speed network, transmission delay can be very small and negligible. Transmission delay on a limited bandwidth access link may be relatively large. But, because the access link is usually close or directly connected to the endpoints, the link capacity is often known by the sender. Thus, transmission delay can be estimated.

Queuing delay is the time that packet waits in a buffer to be transmitted. It depends on several factors, including the egress link capacity, the size of the buffer, the router scheduling policy, and the amount of congestion. Because the underlying cross traffic can change rapidly and dramatically, queuing delay is typically highly variable and unpredictable.

We have demonstrated in the previous section that Sync & Sense can overcome the difficulties in obtaining the queuing delay. With Sync & Sense, queuing delay of each arriving packet can be measured. Propagation delay is the time it takes for the packet to travel along the physical path. It is equal to the path distance between the sender and receiver divided by the speed of the electronic signal. Depending on the type of medium, the electronic signal typically travels at 2/3 of the speed of light. Since the endpoints normally do not know the distance, propagation delay cannot be calculated. In a synchronized system, we can obtain the network delay by using the timestamps when the packet leaves the sender and arrives at the receiver. Such a measurement includes all the four delay components. To identify the propagation delay, further analysis is usually needed. Below, we present a novel mechanism based on Sync & Sense that can estimate propagation delay without requiring synchronization.

Recall that Sync & Sense can virtually synchronize the transmission and reception timing by which there is an unknown offset that includes propagation, transmission and router processing delay. Since router processing delay is negligible, the unknown offset is reduced to the sum of propagation and transmission delays. The mechanism requires that the sender logs the time when the packet is transmitted. Upon receiving the packet, the receiver is required to return an acknowledgement immediately to the sender. The acknowledgement can be as simple as a UDP packet. The sender then logs the time when receiving the acknowledgement packet. This allows the sender to obtain the round-trip time of the transmitted packets. The observed round-trip time (RTT) can be broken down into the delay components as follows:

$$RTT = (D_{tran} + D_{prop} + D_{queue}) + (D_{tran}^{ack} + D_{prop}^{ack} + D_{queue}^{ack}) \quad (3\text{-}8)$$

That is, the observed RTT consists of two sets of transmission, propagation, and queuing delays, the former for the RTP packet on the forwarding path and the latter for the acknowledgement packet on the reverse path.

Several delay components in the Equation 3-8 are known. The sender can calculate D_{tran}. Similarly, on the receiver, D_{tran}^{ack} can be calculated and D_{queue} can be measured as provided by Sync & Sense. The acknowledgement functions to relay both D_{tran}^{ack} and D_{queue}

back to the sender. Due to asymmetric routing, the RTP and acknowledgement packet may not travel on the same physical path. This analysis assumes that the difference in the distance would not make a significant difference between D_{prop} and D_{prop}^{ack}. This is because, in calculating propagation delay, the denominator of the speed of light is extremely large. Thus, we assume that the sum of D_{prop} and D_{prop}^{ack} is equal to $2D_{prop}$. By rearranging the terms, we define RTT' as the observed RTT, excluding the known delay components, or:

$$RTT' = RTT - (D_{tran} + D_{queue} + D_{tran}^{ack}) \quad (3\text{-}9)$$

$$RTT' = 2D_{prop} + D_{queue}^{ack} \quad (3\text{-}10)$$

Thus, Equation 3-10 is a reduced form of Equation 3-8, where D_{prop} and D_{queue}^{ack} are still unknown.

The same assumption used by Sync & Sense can be applied here. Whereas the underlying traffic has bursty behavior, when the acknowledgement packets travel across the network, some of those packets may experience a very minimal queuing delay. As the sender monitors the incoming acknowledgement packets, it can estimate the propagation delay by looking for the minimum of the observations of RTT', or:

$$\text{Min}\{RTT'_1, RTT'_2, RTT'_3, \ldots, RTT'_n\} \approx 2D_{prop} \quad (3\text{-}11)$$

As soon as the sender receives an acknowledgement packet with a minimal queuing delay, D_{queue}^{ack} in Equation 3-10 is eliminated. Then, the propagation delay can be estimated, as the minimum RTT' divided by two. A question with this scheme is that the sender would not know whether the observed RTT' has the minimum value. As a consequence, it is possible that D_{queue}^{ack} may be incorrectly perceived as an integral part of the propagation delay. Such a problem can be eliminated if the sender observes sufficient number of the incoming acknowledgement packets. Due to the fact that the Internet traffic typically has a long-tail distribution, a large majority of the incoming acknowledgement packets would only experience a small queuing delay. This ensures that the problem, if it happens, would be very limited. In our simulation study in section 3.4.3, we see that a few hundred arriving packets

are needed for Sync & Sense to get synchronized. Recall that the synchronizing phase of Sync & Sense requires that two epochs of excursion must be found. Looking for the minimum RTT′ is even easier because only one minimum RTT′ is needed. Therefore, it can be expected that the minimum RTT′ would appear within tens of the arriving packets.

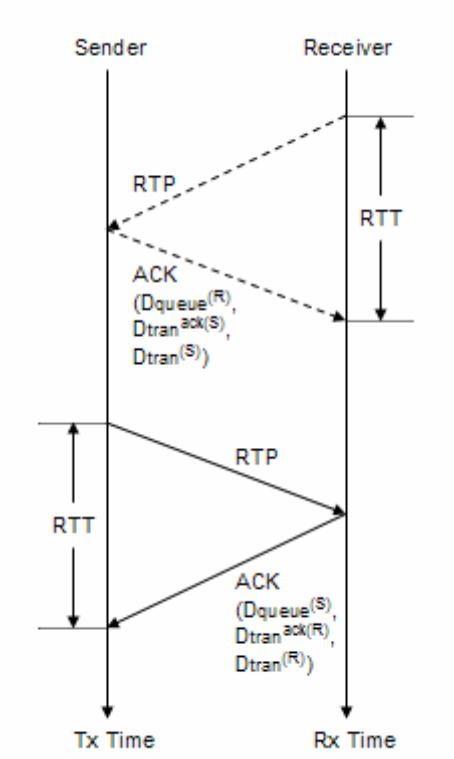

Figure 3-14 Mechanism for estimating propagation delay

Note that, with the above mechanism, the propagation delay is estimated on the sender side. Whereas a VoIP session consists of two separate RTP flows, the receiver also performs the same mechanism when receiving the acknowledgement packet from the sender. It is also necessary that the acknowledgement packet from the sender includes D_{tran}. This allows the receiver to calculate the network delay, which is the sum of the three delay components: queuing delay as provided by Sync & Sense, propagation delay as calculated from the above mechanism, and transmission delay as relayed by the sender's acknowledgement. Figure 3-14 illustrates the mechanism for estimating propagation delay and the exchange of information between the sender and receiver. The figure shows two

separate flows of the VoIP session: the solid lines for the sender's flow and the dashed lines for the receiver's flow. We use superscript (S) and (R) to distinguish the delay components between the sender flow and the receiver flow, respectively.

The calculated network delay is associated with the packet level. In VoIP, it is necessary to assess the delay in the voice frame level. This is because sample voice frames produced from the codec are not transmitted immediately. They spend time waiting to be loaded into the same packet before leaving the sender. The delay incurred in this process is called packetization delay. To the receiver, packetization delay can be simply calculated by identifying the codec type in the RTP packet header. The calculation was discussed in section 3.3.1. Here, we define one-way end-to-end delay of a voice frame as the sum of the network delay and the packetization delay. It is the time from which the voice frame is generated by the codec to which it arrives at the receiver and is ready to be decoded.

Being able to measure the network delay and the end-to-end delay provides a substantial benefit to VoIP. A VoIP agent, including routers and gateways, could use this information to evaluate alternate routes and choose the optimal one for the transmission. Whereas the variation in network delay can also be used as an indication of congestion, the measured network delay also helps to improve the calculation of the delay variation, called jitter. The RTP specification [8] is limited to use inter-arrival jitter, because it is assumed that the sender and receiver are not time synchronized. Inter-arrival jitter is defined as the mean deviation (smoothed absolute value) of the difference in the inter-arrival gap compared to the inter-departure gap for a pair of packets. The measured network delay allows the calculation of absolute jitter, which is the mean deviation of the queuing delay for an arriving packet.

In dealing with delay variation, the receiver uses a jitter buffer (also called playout buffer). A jitter buffer holds early arriving packets so that late arriving packets can still be in time; and thus all arriving packets can be played out smoothly. Because the current network condition is not known a priori, an adaptive jitter buffer is preferred over a fixed jitter buffer. An adaptive jitter buffer attempts to match the jitter delay to the delay characteristic of the network. There is no standard algorithm for calculating the jitter delay; examples can be found in [32, 33, 34]. The objective is to keep the jitter delay as small as possible, while at the same time minimizing packets being dropped by the jitter buffer itself. A problem with adaptive jitter buffers is that the algorithm tries to reach what it sees as an optimal jitter delay

without being aware of the actual end-to-end delay. The consequence is that packets could be unnecessarily dropped by the buffer, even if the actual delay is still below the delay budget. As the end-to-end delay can be measured, Sync & Sense allows the receiver to combine the unique benefits of both adaptive and fixed jitter buffers. When the observed delay never exceeds the delay budget, it is better to use a fixed jitter buffer. This can prevent any packet being dropped by the jitter buffer. Also, a fixed jitter buffer does not cause distortion on the playout as is the case with an adaptive jitter buffer. The receiver may use an adaptive jitter buffer when the observed delay appears higher than the acceptable level. In this case, it is necessary to compromise some packet loss in order to reduce the end-to-end delay.

Below, we summarize all delay components of a VoIP session in a grand equation of mouth-to-ear delay:

$$D_{m2e} = D_{codec} + (D_{pack} + (D_{tran} + D_{prop} + D_{queue})) + D_{jit} \qquad (3\text{-}12)$$

where network delay $D_{net} = D_{tran} + D_{prop} + D_{queue}$, and end-to-end delay $D_{e2e} = D_{pack} + D_{net}$. Mouth-to-ear delay accounts for the total latency of a voice sample from the speaker's mouth on one end to the listener's ear on the other end. Codec delay includes algorithmic delays for compression (on the sender) and decompression (on the receiver). Codec, packetization, and transmission delays are constant. Propagation delay is also constant, and can be estimated by using the Sync & Sense based mechanism. Queuing delay is variable, which can be measured by Sync & Sense. Depending on the type of jitter buffer, jitter delay can be constant or variable. As we have discussed so far, Sync & Sense enables the receiver to be able to assess all the delay components in Equation 3-12, without requiring synchronization between the sender and receiver. Note that processing delays in the endpoints and routers are omitted in the equation. These delays are typically negligible, but they could vary depending on the utilization of the processor. It is usually difficult to measure these delays. In the above equation, these delays would be perceived as part of the propagation delay.

3.5 Dealing with Network Pathologies

Network pathologies are unusual or unexpected behaviors of the network, for example, out-of-order delivery, packet replication, and packet corruption. As shown previously in the performance study, Sync & Sense can provide accurate measurements of queuing delay, under a common network assumption. Namely, a network is a series of FIFO queue servers in which packets get queued behind each other while in transit. Under this assumption, the network delay has its integrity intact. Therefore, the observed network delay can be useful, as it allows Sync & Sense (and any methods) to infer the network condition. Unfortunately, several network pathologies violate this assumption. This restricts the use of the observed network delay because it may be misleading. Below, we identify network pathologies that could be potential problems for Sync & Sense.

Packet reordering can occur any time when a route changes, if the new route has a lower propagation delay than the old one. Sync & Sense can easily detect this problem by observing out-of-sequence packets. Since a decrease in propagation delay due to the route change would be perceived by Sync & Sense as falling queuing delay, it is necessary for Sync & Sense to abandon the existing synchronized timing. Then, Sync & Sense can start the synchronizing phase again and get synchronized based on the new route. Although packet reordering may allow Sync & Sense to infer as a route change, the problem of a route change itself can be undetectable in many cases. If the new route has a lower propagation delay than the old one, packet reordering may not happen. This is because cross traffic also has an impact on the delay of the packets. If the new route has a higher propagation delay than the old one, packet reordering certainly does not occur. This is undetectable and Sync & Sense may perceive the increased delay as additional queuing delay. Another potential problem is multi-channel links, which may operate in parallel. A router may balance its outgoing load across two different links. In this case, a packet may go out over the first channel and the next packet may go out over the other channel. They don’t queue behind each other. Multi-channel links violate the assumption that there is a single end-to-end forwarding path.

As the network becomes more heterogeneous, wireless networks are often part of the forwarding path. Due to relatively low quality of the wireless link, several mechanisms are employed in order to improve the performance. In many wireless networks, the data link layer may perform error recovery, such as retransmission, to provide reliability of the

wireless link to the upper layer. This could alter the integrity of the network delay assumption. As a matter of fact, since voice packets are based on UDP, they should be discarded in the event of packet loss, even in the link level. Furthermore, in cellular networks, handoff typically results in more delay. The additional delay is introduced since it takes time to forward packets to the new base station and to perform the handover procedure.

The solution to the above mentioned problems lies in detecting whether the integrity of the network delay assumption is violated. This is not an easy task, given that Sync & Sense operates in an end-to-end fashion, with no explicit support from the network. In fact, any end-to-end control mechanism is vulnerable to these network pathologies. For instance, TCP relies on the round-trip time observations in its timeout and retransmission strategies. Misleading delay observations could easily affect TCP performance. In case of Sync & Sense, if the problem is momentary, it can be resolved upon detecting it. When a route change is detected, Sync & Sense can simply abandon the existing synchronized timing and get synchronized again based on the new route. In other cases, such as in wireless environments, the underlying assumption may be completely broken. Upon detecting this, it would be safer to disable any measurement by Sync & Sense. Nevertheless, we believe that there are solutions to deal with the network pathologies. Further research is needed to study each individual of the network pathologies and explore the options to deal with them.

3.6 Detecting Route Change

In this section, we present a solution that allows Sync & Sense to deal with the problem of a route change. A route change, in which the new route has a lower propagation delay, than the old one, is not a problem for Sync & Sense. Such a route change usually results in packet ordering. Thus, Sync & Sense can simply detect it by observing out-of-sequence packets. Nonetheless, packet reordering may not occur, due to the effect of cross traffic. The Sync & Sense algorithm itself has the ability to detect and adjust if the propagation delay has decreased. Referring to Figure 3-4 (b), epochs of excursion are always on or above the baseline for the transmission timing. That is, the propagation delay is considered equal to zero. A decrease in propagation delay, due to a route change, would cause Sync & Sense to observe the epoch of excursion running relatively below the baseline.

As soon as the route changes, the dispersion gap of the arriving packets would violate Equation 3-6. This causes Sync & Sense to abandon the ongoing epoch of excursion. Sync & Sense then restarts the synchronization process with the arriving packet that violates Equation 3-6. When Sync & Sense observes two valid complete epochs of excursion, the synchronization is declared. This results in a new lower baseline that reflects the decrease in propagation delay. On the contrary, an increase in propagation delay due to a route change is a problem for Sync & Sense. It is undetectable because epochs of excursion are still above the baseline for the transmission timing and would never violate Equation 3-6. In the following, we present an extension to the Sync & Sense algorithm, which is aimed to solve the problem when the new route has a higher propagation delay.

3.6.1 Extended Sync & Sense Algorithm

When the propagation delay increases, the ongoing epoch of excursion will never end. In other words, it will never come back down to the baseline for the transmission timing. The basic idea of the solution is that, when the propagation delay increases, there will be a new higher baseline that can complete the ongoing epoch of excursion. The goal of the algorithm is to find such a new higher baseline, which only exists if the route has changed. We denote the baseline for the transmission timing as the primary. This primary baseline is based on Equation 3-6 in which the dispersion gap of an arriving packet must be equal or greater than the packetization delay. This condition allows Sync & Sense to run a search to find the first arriving packet with zero queuing delay. Once Sync & Sense declares the synchronization, it runs another search for the secondary baseline. This secondary baseline is used to find complete epochs of excursion that are above the primary baseline, if the propagation delay has increased. Since the synchronization has been declared, the queuing delay for each arriving packet can be calculated. Hence, the secondary baseline can run based on the raised queuing delay. For illustration purpose, we assume the first arriving packet as seen by the secondary baseline has the queuing delay of 10 milliseconds, which is assumed to be due totally to an increase in propagation delay. In other words, this arriving packet has zero queuing delay based on the new route, but it is perceived as having 10-millisecond queuing delay based on the old route (using the primary baseline). If the mentioned assumption is

correct, the arriving packets that follow must have the queuing delay equal or greater than 10 milliseconds. Therefore, similar to Equation 3-6, the secondary baseline uses the following condition to test each arriving packet:

$$D_{queue} \geq \hat{D}_{queue} - \varepsilon \qquad (3\text{-}13)$$

where $\hat{D}_{queue}$ is the raised queuing delay that is assumed to be due totally to the increase in propagation delay. The above condition must hold for all arriving packets. The secondary baseline runs exactly the same as the primary baseline. When Equation 3-13 is violated, it means that the assumption, which $\hat{D}_{queue}$ is not totally due to the increase in propagation delay, is wrong. The ongoing epoch of excursion is abandoned. D_{queue}, which has lower queuing delay than $\hat{D}_{queue}$, becomes the next $\hat{D}_{queue}$. In other words, the secondary baseline moves lower, and the new epoch of excursion begins. Eventually, $\hat{D}_{queue}$ reaches the lowest point that reflects the raised queuing delay, due to the increase in propagation delay. Then, Equation 3-13 will never be violated. When two valid complete epochs of excursions are found, it proves that the propagation delay has indeed increased. Therefore, Sync & Sense abandons the existing synchronization. When Sync & Sense restarts the synchronizing phase, it will be automatically based on the new route.

As described above, the secondary baseline uses Equation 3-13 to test $\hat{D}_{queue}$ from top to bottom. Whenever the queuing delay shows an increasing trend, it is possible that this may be caused by an increase in propagation delay. It is necessary for the secondary baseline to reset $\hat{D}_{queue}$ to the highest queuing delay, when an increasing trend is detected. This allows the secondary baseline to work its way down to find the bottom, which reflects the raised queuing delay due to the increase in propagation delay. To detect an increasing trend in the queuing delay, Sync & Sense maintains a set of the queuing delays $\{D^{i}_{queue}, i = 1,\ldots,n\}$ of the most recent arriving packets, where n = 12 (empirically selected). For each arriving packet, Sync & Sense uses the following equation to test the queuing delay trend:

$$QT = \frac{\sum_{i=2}^{n} I\left(D_{queue}^{i} > D_{queue}^{i-1}\right)}{n} \quad (3\text{-}14)$$

where I(X) is one if X holds, and zero otherwise. This method measures the fraction of consecutive queuing delay pairs that are increasing; thus, $0 \leq QT \leq 1$. If the queuing delays are independent, the value of QT is 0.5. If there is a strong increasing trend, QT approaches one.

3.6.2 Simulation Result

Below, we demonstrate how the extended Sync & Sense algorithm deals with the event of route change. Unless specified otherwise, we conduct a simulation using the same network topology, with 10 hops, as well as parameters as described in section 3.4.1. We impose route changes by increasing/decreasing the propagation delay on one of the links. The extended Sync & Sense algorithm is implemented at the VoIP receiver. The error margin parameter is set to 1.5 milliseconds. The simulation result is shown in Figure 3-15. Figure 3-15 (c) shows the actual one-way network delay of the VoIP packets. Route changes can be seen from the figure since minimum network delay reflects the propagation delay. Accordingly, Figure 3-15 (b) shows the packets lost in the network. Figure 3-15 (a) shows the accuracy performance in the form of measurement error.

From the figures, we can see that Sync & Sense can quickly detect the decrease in propagation delay, and resynchronize the timing based on the new route. Large measurement errors can be seen for a short period of time while Sync & Sense adjusts its baseline for transmission timing to the new lower propagation-delay route. Overall, Sync & Sense performs very well in the event of a decrease in propagation delay. In the case of an increase in propagation delay, the simulation result shows that the extended Sync & Sense algorithm works as expected. The figures show that, as soon as the route changes, the measurement error surges and is equal to the increased amount in propagation delay. This is because the measurements are still made based on the primary baseline, which is based on the old route. It takes a while before the secondary baseline gets synchronized and detects the increase in

propagation delay. Once detected, Sync & Sense can quickly resynchronize the primary baseline to be based on the new higher propagation-delay route.

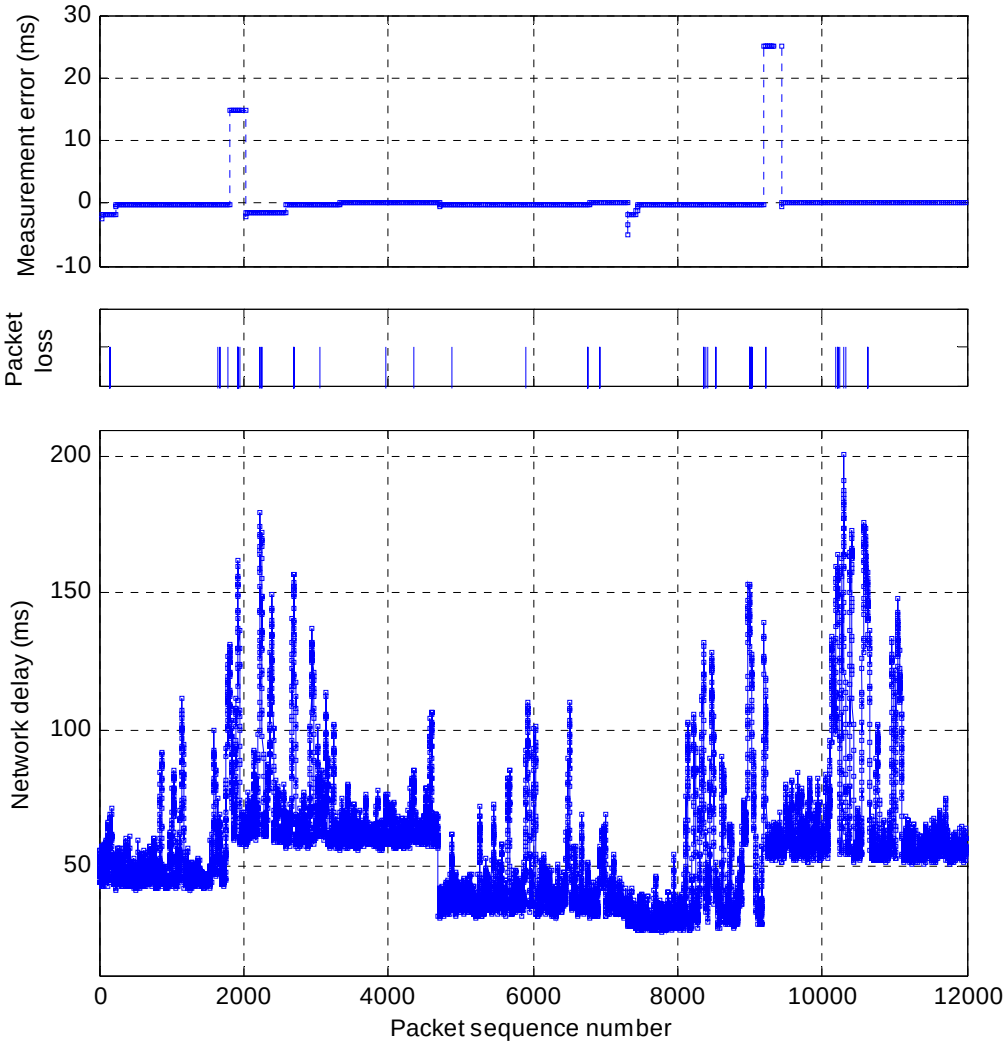

Figure 3-15 Detecting route change

The drawback of the extended Sync & Sense algorithm is that such a surge in the measurement error may stay for a longer period of time. This drawback seems inevitable due to the nature of the problem. The duration of the surge in the measurement error is essentially due to the underlying traffic. If the network is lightly loaded, it is likely that the secondary baseline can get synchronized very soon. On the other hand, if the network is congested, it would take longer to detect the route change.

3.7 Summary

This chapter focuses on network state detection, which is an essential component for a VoIP system enhanced with the ability to adapt in response to the network condition; for example, adaptive-rate VoIP. The objective of detecting the state of the network is to acquire the characteristics of the network that are critical to the control decision. For VoIP, both delay and loss are critical to voice quality. One-way end-to-end delay affects echo and conversational interactivity, while packet loss affects voice clarity. Detecting packet loss is very simple as it can be done by observing out-of-sequence packets. However, measuring one-way end-to-end delay is a real challenge. This is due to the reality that the endpoints are normally not time synchronized. Without external synchronized timing, it is almost impossible to measure one-way network delay. Here, we propose a novel measurement methodology for VoIP called Sync & Sense, which can overcome the challenge and truly serve the objective of detecting the state of the network for VoIP.

As indicated by its name, the Sync & Sense methodology consists of two phases. The synchronizing phase is the ability to virtually synchronize the transmission and reception timing of the VoIP session, based solely on observing the incoming RTP packets. The synchronization relies on the assumption that some arriving packets may experience a very minimal queuing delay when traveling from the sender to the receiver. Epoch of excursion of queuing delay is defined as the time from an arriving packet with a very minimal queuing delay to the next arriving packet with a very minimal queuing delay. The Sync & Sense algorithm tracks the transmission timing by using the known packetization delay. At the same time, it tracks the reception timing by observing inter-arriving gaps of the arriving packets, and searches for epochs of excursion. The synchronization can be constructed based on the fact that the elapsed time of a valid complete epoch of excursion must be equal to the elapsed time of the transmission timing. Sync & Sense ensures that this condition, along with additional conditions, is satisfied before declaring the synchronization. Once synchronized, the sensing phase operates by measuring dispersion gaps of the arriving packets. The dispersion gap is the interval in which the arriving packet disperses from the packetization delay. The dispersion is caused by queuing delays in the network. In other words, the dispersion gap includes both the queuing delay and the packetization delay. Since the dispersion gap can be measured and the packetization delay is known, Sync & Sense can find

the queuing delay. Being able to measure the queuing delay allows Sync & Sense to estimate the propagation delay and obtain the full spectrum of the delays of the VoIP session. In addition, Sync & Sense has the ability to estimate the available network bandwidth. Along with packet loss, Sync & Sense can provide a wider range of indications about the network condition.

Our simulation study shows that Sync & Sense can provide impressive performance, in terms of synchronization and measurement accuracy. Sync & Sense is robust to packet loss. A potential factor that could affect the performance is the number of hops between sender and receiver. This is because, as the number of hops increases, Sync & Sense would observe fewer arriving packets with a very minimal queuing delay. However, this problem is manageable by using an error margin in the synchronizing process. The simulation result shows that the error margin parameter is the controlling factor over the performance. Using a relatively large value of the error margin allows Sync & Sense to use, not only the arriving packets with a very minimal queuing delay, but also the arriving packets with a larger minimal queuing delay to construct the synchronized timing. Thus, this can help to increase the synchronization performance. A drawback of using a larger error margin is accepting more error to the measurements. Nonetheless, the simulation result shows that the impact is very small and limited. Given the drawback, a compromise value should be chosen in order to optimize the performance between synchronization and accuracy.

Like other probing schemes, Sync & Sense operates under the assumption that a network is a series of FIFO queue servers in which packets get queued behind each other while in transit. Under this assumption, the network delay has its integrity intact. Therefore, the observed network delay can be useful to infer the state of network. Unfortunately, several network pathologies violate this assumption and restrict the use of the observed network delay; for example, route change, multi-channel links. The solution to this problem lies in detecting whether the integrity of the network delay assumption is violated. However, this is not an easy task, given that Sync & Sense operates in an end-to-end fashion. Here, we also present an extended Sync & Sense algorithm that aims to deal with the problem of route change. Future research is needed to study each individual of the network pathologies and explore the options to deal with them.

Chapter 4

Sync & Sense Enabled Adaptive Packetization VoIP

Requiring fixed bandwidth is an inherent problem with VoIP. This is because packet-switched networks, including the Internet, provide no guarantees about bandwidth, as well as delay and loss. Because the available network bandwidth can vary at any time, it often results in unreliable and unpredictable voice quality. A solution to this problem is to enhance VoIP with the ability to adapt its transmission to match the available bandwidth, called adaptive-rate VoIP. Whereas adaptive-rate VoIP attempts to minimize network congestion, it helps to lower network delay and reduce excessive packet loss. An adaptive-rate VoIP system is composed of three fundamental components: rate adaptation, network state detection, and control mechanism. In the previous chapters, we have studied the first two components. We demonstrate that packetization is an effective means that allows VoIP to perform rate adaptation. And, we propose a novel measurement methodology for VoIP called Sync & Sense of periodic stream. Without requiring external support for synchronization, Sync & Sense has the ability to virtually synchronize the transmission and reception timing that allows measurement of one-way network delay. In addition, Sync & Sense can estimate the full spectrum of the delays of the VoIP session and estimate the available network bandwidth.

Here, we design an adaptive-rate control that makes use of our previously proposed components of rate adaptation and network state detection. The integration of the three components is a novel and unique adaptive-rate VoIP called Sync & Sense Enabled Adaptive Packetization VoIP. Unlike other adaptive-rate VoIP systems that are based on a variable bit rate speech coder, our adaptive VoIP has the advantage that it works with any constant bitrate coder. As rate adaptation is independent of the coder, it is likely to be more transparent to the user. Unlike other adaptive-rate VoIP, in which the control decision is primarily based on

packet loss, the Sync & Sense methodology allows our adaptive VoIP to be able to make the control decision based on both delay and loss. While responding to packet loss is reactive, observing of the trend of one-way network delay allows our adaptive VoIP to implement a proactive strategy that reduces the transmission rate in anticipation of potential packet loss. In addition, the delay trend allows our adaptive VoIP to gain more insight about the network state and to better respond in order to optimize the performance.

To test our proposed adaptive VoIP, we conduct a simulation study in which the performance is compared against traditional VoIP. In a high statistical multiplexing environment, the result shows that our adaptive VoIP can optimize the performance under any given network load. On the other hand, traditional VoIP cannot, because of its inability to adapt. In a low statistical multiplexing environment, we examine the interaction between our adaptive VoIP flow and other types of traffic flow that compete for the shared network bandwidth. The result shows that our adaptive VoIP has the needed aggressiveness to compete with TCP for its share of bandwidth; at the same time, it is responsive to network congestion. In a homogeneous adaptive VoIP network, the result shows the benefit that more flows can be accepted with minimal impact on the performance. The adaptive VoIP flows automatically choose to operate at a lower rate when the network gets more congested.

4.1 Background and Related Research

Adaptive-rate control is a critical component of any adaptive-rate VoIP. The control aims at optimizing the delay and loss performance, while at the same time it attempts to minimize network congestion. Thus, adaptive-rate control can also be viewed as congestion control. Congestion control is a necessity, especially in packet-switched networks, because the network generally admits all traffic from its users. Congestion control on each endpoint as a whole helps to prevent the overall network from over-utilization. This in turn allows the endpoints to optimize the performance. AIMD (Additive Increase Multiplicative Decrease) is the most common and well-known algorithm for congestion control, as it is used in the TCP protocol. For data applications, AIMD has proven to be an effective scheme, not only for minimizing congestion, but also optimizing the throughput. AIMD drastically reduces the transmission when it observes an indication of congestion, usually packet loss. This is a

quick and effective means to clear congestion. As packet loss is minimized, decreased retransmission allows increased throughput. Without the indication of congestion, AIMD assumes more bandwidth is available, and increases the transmission conservatively.

For VoIP, the performance refers to the delivered quality, rather than the throughput. Thus, the main objective of adaptive-rate control can be different from AIMD, although they both attempt to minimize network congestion. As VoIP is delay-sensitive, designing adaptive-rate control must aim at minimizing the delay as the top priority, and minimizing packet loss as well. Several other issues should also be considered. Namely, rate adaptation should be smooth so it does not cause distraction. The interaction of the traffic flows should be examined; whereas the adaptive-rate VoIP flow should demonstrate that it can compete with the AIMD-based TCP flow in order to acquire its share of bandwidth. The adaptive-rate control should provide a fair bandwidth allocation when the adaptive flows share the same link. Additionally, we need to determine the position of adaptive-rate VoIP towards TCP-friendliness because attempting to achieve fairness would not allow optimize the performance.

Our survey indicates that research in this area is still in its infancy. Only a limited number of studies related to adaptive-rate VoIP are available [5, 6, 20, 21, 22, 35]. Although all studies aim at improving the performance in comparison to traditional VoIP, their focus is specific and does not cover the desirable properties that adaptive-rate VoIP should possess. For example, little attention has been paid to minimizing one-way end-to-end delay, which affects echo and conversational interactivity of the perceived quality. In most of the studies, the adaptive-rate VoIP makes a decision based on packet loss, which only affects voice clarity of the perceived quality. Some studies adopt the AIMD algorithm as part of the control mechanism. Some studies aim for adaptive-rate VoIP to be TCP-friendly. In the following, we briefly summarize some of the existing papers on adaptive-rate VoIP.

Abreu-Sernandez et al [20] propose an adaptive multi-rate speech coder for VoIP. The coder incorporates a control mechanism that selects one of five possible operating modes, depending on network congestion. The transmission rate ranges from 8.8 to 10.6 Kbps. Rate adaptation is based on the following loss levels: less than 15%, 15 – 20%, 25 – 30%, 30 – 35%, and more than 35%. The sender obtains the average loss rate from RTCP receiver reports sent from the receiver. Qiao et al [5] propose the use of an objective measure of

perceived speech quality (i.e. objective MOS score) with the control mechanism. The adaptive-rate VoIP is based on the AMR (Adaptive Multi-Rate) speech coder [7]. On the receiver, speech quality is predicted from packet loss rate using a PESQ (Perceptual Evaluation of Speech Quality) based method. The predicted speech quality serves as a feedback that is sent to the sender via RTCP reports, every five seconds. The control mechanism adopts the AIMD algorithm. When the predicted MOS increases by 0.2, the bitrate of the coder is set to the next higher step. When the predicted MOS decreases by 0.5, the bitrate of the coder is reduced by half (to the closest step). In between those two thresholds, the bitrate of the coder is set to the next lower step.

In Barberis et al's work [21], the control makes a decision based on delay and loss measurements as provided by RTCP receiver reports. Rate adaptation is assumed using a hypothetical speech coder in which the transmission rate varies between 8 and 64 Kbps in steps of 8 Kbps. The AIMD algorithm is adopted. When packet loss rate exceeds the loss threshold, the transmission rate is reduced by half. If the measured delay increases by 10 percent of the mean delay, the transmission rate decreases by a step. The transmission rate conservatively increases when delay and loss are below their thresholds. Beritelli et al [6] study adaptive-rate VoIP that is based on TCP-friendly algorithms, including RAP and TFRC. Whereas TCP-friendly algorithms were designed to achieve fairness with TCP flows, the authors aim to study the performance aspect when they are applied to VoIP. In addition, both RAP and TFRC algorithms are modified in which the control mechanism uses delay variation as part of the rate adjustment decision. Following the RAP and TFRC algorithms, the delay variation is calculated using RTT (Round-Trip Time). Rate adaptation is achieved through the AMR speech coder.

Mohamed et al [22] primarily focus on developing a neural network based automaton that measures speech quality in real-time. The authors develop a database comprised of a set of samples of distorted speech signals and their associated subjective quality scores (i.e. MOS). Distorted speech signals are based on packet loss rate and loss distribution (i.e. the number of consecutively lost packets). The database is used to train the neural network to assess speech quality. The control mechanism is a hybrid of using the TFRC scheme [10] and the trained neural network. The TFRC scheme periodically calculates a suggested transmission rate. The neural network measures the MOS score based on the network

condition. Rate adaptation is done by changing different speech coders. RTCP reports are used to transport feedback from the receiver to the sender.

4.2 Design of Adaptive-Rate Control

In this section, we discuss design choices for adaptive-rate control, which is a key component of adaptive-rate VoIP. An adaptive-rate control takes into account the observations provided by the network state detection component before making a decision on rate adaptation. The object is to optimize the delay and loss performance; at the same, also minimizing network congestion. We identify three key issues that should be considered when designing adaptive-rate control, as follows: placement of the control, decision metrics, and increase/decrease algorithm. Below, we discuss each issue in details.

4.2.1 Placement of the Control

Whereas following the end-to-end principle [4] is the key requirement in our design of adaptive-rate control, both endpoints of the VoIP session are expected to be able to operate intelligently and independently, without support from the network. Under this architecture and because a VoIP session is actually two separate streams of RTP packets, monitoring the network condition can only be done at the receiver by observing the incoming packets. However, rate adaptation must be done at the sender. The choice is where the control should be placed, either at the sender or the receiver.

In sender-based control, observations about the network must be reported back to the sender. Utilizing RTCP is a common scenario. The observations (such as packet loss and inter-arrival jitter) are summed up into a reception feedback report. The report is sent out periodically, usually on the time scale of seconds. Although this helps to reduce the feedback traffic, it limits the decision frequency of the control. The major drawback is that the control decision is restricted with the pace of receiving the feedback. As a consequence, the control may be too slow to respond to the network. If the control needs to make the decision more frequently, it is would be necessary to use alternate feedback schemes rather than RTCP. But, because all the network observations must be fed back to the sender, this could introduce additional traffic on the reverse path.

On the other hand, receiver-based control seems to be a better choice. Since observations about the network are obtained by the receiver, they can be processed immediately and continuously. In this scenario, the feedback conveys the control decision about the rate change. The feedback may be sent only upon a rate change. This can significantly help to reduce the feedback traffic to minimal. Inevitably, since feedback lost in the network is a potential problem, to ensure reliability, the feedback may be transported using the TCP protocol.

4.2.2 Decision Metrics

Decision metrics refer to the observations, as provided by the network state detection component, which the control uses to infer the network condition and make a decision. Packet loss is the most common decision metric in any control mechanism. This is because it can be simply obtained by observing out-of-sequence packets. In packet switched networks, packets get lost for two reasons. First, random loss occurs independently due to bit errors, hardware errors, etc. Second, network congestion can cause overflows of any insufficient buffer capacity somewhere on the path. On most network paths, loss due to damage is rare. So it is probably that a packet loss is due to congestion in the network [31]. This has been a fundamental assumption in the design of congestion control in TCP as well as numerous proposed control schemes. Nowadays, the network becomes more and more heterogeneous. A network is not necessarily entirely wired. A wireless segment, for example, could well contribute to a higher loss probability. End-to-end based control would not able to distinguish a packet loss from any causes, and would have to take into account all lost packets no matter what the causes are. Nonetheless, packet loss still seems to be the most feasible implicit indication of network congestion. This is primarily due to the presence of competing TCP traffic. Depending on its use, packet loss as a decision metric can be classified into loss event and loss rate. In loss event, the control responds to any events of packet loss, which is a conservative strategy. In loss rate, the control monitors the loss rate and makes a decision whenever the loss rate reaches a threshold. Loss rate can be simply obtained as the loss count divided by the total number of packets received. Loss rate is generally a decision metric that is less responsive than loss event.

Besides packet loss, the receiver may observe packet inter-arrival gaps and use them as a decision metric. Constant packet rate traffic is a good property of VoIP because the inter-departure gap is fixed and known by the receiver. As the packets traverse the network, they arrive at the receiver with variable delays and inter-arrival gaps. The delay variation can suggest congestion in the network. When the inter-arrival gap equals, or is close to, the inter-departure gap, it suggests the network is under no congestion. High expansion/contraction of the inter-arrival gap shows the degree of congestion. Due to the bursty nature of Internet traffic, however, observing the delay variation can be less meaningful and only provides transient congestion. High delay variation may appear at times, but not necessarily because the network as a whole is over-utilized. As the matter of fact, packet loss is still a better and more effective decision metric to detect persistent or threatening congestion in which a response is needed.

Another means to obtain decision metrics is to use the reception feedback report provided by RTCP. The report is intended to serve as feedback to the sender about reception statistics, which include packet loss count and inter-arrival jitter (or delay variation). It is transmitted periodically, usually on the time scale of seconds. The limitation is that the control on the sender can only view packet loss in the form of cumulative loss count and fraction loss between the report intervals. Similarly, the inter-arrival jitter is only a snapshot of the average at the time of a report. The control may calculate packet loss rate for a decision metric. In addition, RTCP also provides a mechanism in which the sender can compute the round-trip time between the sender and receiver. This, however, is a very coarse estimate because most network paths have asymmetrical delays.

Mentioned above are feasible decision metrics that are available under standard RTP /RTCP implementation. The Sync & Sense measurement methodology proposed in the previous chapter allows more options for the decision metric. Sync & Sense has the ability to measure the reception rate of the incoming packets, which also indicates the available network bandwidth. When sufficient network bandwidth is available, the reception rate would relatively equal to the transmission rate. When the network becomes more congested, the available bandwidth decreases. The reception rate would be lower than the transmission rate. The difference between the reception rate and the transmission rate can be an effective decision metric that reflects the network condition. The lower the reception rate is from the

transmission rate, the more the congestion. While the reception rate can give an idea of mild to moderate congestion, packet loss is an indication of severe congestion. The reception rate together with packet loss, hence, can provide the full range of detecting the state of the network.

In addition, Sync & Sense has the ability to measure the one-way network delay of the arriving packets. The trend of one-way network delay can be very useful to sense the state of the network. An increasing delay trend usually indicates that network congestion is building up. As network bandwidth becomes less available, packets arrive at the receiver with progressively higher delays. A number of researchers have confirmed that there is a correlation between delay and loss on the Internet [36, 37, 38, 39]. Specifically, there is a range of one-way network delay in which packet loss is likely, which we denote as the delay threshold. This is simply due to the fact that, when the bottleneck link queue is full, packet loss begins to occur. In reality, the correlation can be affected by several factors, including the number of hops along the path. Nonetheless, the delay threshold is bounded. Assuming all the links are bottlenecks, the upper bound of the delay threshold is the sum of the queuing time from every link. This could happen when a packet arrives at the end of the queue on every link, but it is extremely unlikely. As a matter of fact, a few bottleneck links with limited capacity would dominate to the one-way network delay. Therefore, the delay threshold can be used as a proactive decision metric that allows the control to respond sooner to prevent potential packet losses.

4.2.3 Increase/Decrease Algorithm

The increase/decrease algorithm determines the rate change in response to network condition as indicated by the decision metrics. The choices of the increase/decrease algorithm rely on the capability of rate adaptation, which is essentially based on the speech coder being used. Waveform speech coders (such as PCM, ADPCM) produce tiny sample voice frames at the basic sampling rate. The G.711 PCM coder, for instance, generates a one-byte voice frame every 0.125 milliseconds. Packetization thus can be as small as one byte up to hundreds of bytes. This allows fine-grain rate adaptation. The control can have great flexibility in adjusting the transmission rate. It may choose to increase the rate aggressively,

in order to quickly gain the available bandwidth. Or, it may choose to increase the rate conservatively, in order to avoid network congestion. Because the step of the rate change is small, multiple adjustments are usually needed. Accordingly, the control must make decisions more frequently. The decision frequency can be as often as one round-trip time, at the maximum rate. This allows the control to observe the impact of the rate change before making the next decision. In sender-based control, it takes at least one round-trip time from when the sender switches to a new rate to when the sender receives a feedback of such a rate change. Similarly, in receiver-based control, it takes at least one round-trip time from when the receiver sends a rate change command (to the sender) to when the receiver observes an arriving packet with such a rate change. As the rate change is made over short time intervals, the control is responsive and can react to rapid changes in the network. However, the downside is additional traffic load on the network, due to the increased amount of feedback exchanged between the sender and receiver.

Unlike waveform coders, voice source speech coders (such as G.729, G.723.1) produce sample voice frames that are parameters of the model of voice source signals. A voice frame is generated every compression cycle, which is typically in the few tens of milliseconds range. Accordingly, the size of a voice frame is in the few tens of bytes range. Thus, packetization is limited to a couple of voice frames, without causing too much packetization delay. This results in coarse-grain rate adaptation, which allows only a certain number of transmission rates. Similarly, variable rate speech coders (such as AMR [7]) are a type of voice source coders that can produce sample voice frames at different bitrates. They can be used for coarse-grain rate adaptation as well. Because the step of the rate change is large, the rate change can be less frequent. That means less feedback is exchanged between the sender and receiver. The advantage is very little additional traffic needed for the feedback. However, the control may not benefit quickly from rapid changes in the network. The goal of coarse-grain rate adaptation is to adjust the transmission rate to the average available bandwidth, rather than the instantaneous. Since the VoIP bandwidth requirement is relatively small, coarse-grain rate adaptation should be suitable for VoIP. Fine-grain rate adaptation may not actually benefit much from instantaneous increases in the available bandwidth.

In the steady state, the control may reach equilibrium in which the transmission rate matches the available network bandwidth. However, oscillation around the optimal rate is

inevitable. This is due to the property of a control that it always attempts to perform optimization. Given a steady state, the situation can be illustrated as follows. When no sign of congestion is seen, the control would increase the transmission rate. But moments later, such a rate increase would cause the control to observe congestion. The control would then need to reduce the transmission rate back to where it was. Rate adaptation in a smooth manner is a desirable property for adaptive-rate VoIP, because it can minimize distraction to the user that may be caused by the rate change. With small steps of the rate change, fine-grain rate adaptation has a major advantage that it allows smooth transmission rate and minimal rate oscillation. In contrast, coarse-grain rate adaptation could suffer from rate oscillation. The degree of the problem tends to be higher when the adaptive-rate VoIP is based on a variable rate speech coder. This is because the bitrate is associated with the audio quality of the output voice frames. The oscillation in the voice quality could noticeably reduce the perceived quality.

Issues to be considered when designing the increase/decrease algorithm for real-time applications include *aggressiveness*, *responsiveness*, *convergence*, *smoothness*, and *fairness* [40, 41]. Aggressiveness and responsiveness refer to the transient state where the control attempts to adapt the transmission rate to reach equilibrium. An increase algorithm or aggressiveness determines the ability of the control to acquire the available bandwidth; while, a decrease algorithm or responsiveness determines its ability to avoid network congestion. Convergence measures how fast the algorithm converges to the optimal transmission rate. The most common increase/decrease algorithm is AIMD (Additive Increase/Multiplicative Decrease), which is used in the TCP protocol. AIMD possesses most of the mentioned desirable properties and has proven for decades to be successful and practical over the Internet. Adopting the AIMD algorithm is an option for adaptive-rate control. However, one property that AIMD lacks is smoothness. This is due to the fact that AIMD was originally designed for data communications, in which smoothness is not a concern. AIMD halves the transmission rate in response to a single congestion indication. This is unnecessarily severe, particularly for VoIP, which could degrade the perceived quality. Therefore, the smoothness issue should be addressed when adopting the AIMD algorithm. Adjusting the aggressiveness and responsiveness can improve the smoothness of AIMD. For fine-grain rate adaptation, any plausible increase/decrease algorithm can be supported, including adopting the AIMD and

similar algorithms. Because coarse-grain rate adaptation only allows a certain number of transmission rates, it leaves very few options for the increase/decrease algorithm. A stepwise algorithm is a possibility. Adopting the AIMD and similar algorithms is possible, but would not be complete.

Fairness (also called TCP friendliness or TCP compatibility) is an emerging issue, especially for VoIP. Because real-time applications rarely implement congestion control, they do not share the available bandwidth fairly with competing TCP traffic. While the network today has an increasing amount of real-time traffic, the concern is that the current situation could evolve to the starvation of TCP traffic in the future. Non-TCP applications are currently encouraged to implement a congestion control mechanism so that they behave fairly to competing TCP traffic. As suggested in [42], a TCP-friendly flow should not have its long-term throughput exceed a conforming TCP flow under the same condition. Several protocols for real-time applications have been developed to achieve fairness and, at the same time, ensure smooth transmission. GAIMD [43], SIMD [41], and IIAD [44] are some of the proposed protocols that are based on the AIMD algorithm. Non-AIMD algorithms that offer TCP-friendliness and smooth transmission include TFRC [10] and LDA [45].

Most TCP-friendly protocols, however, are studied in the context of video streaming applications. This is because video traffic consumes considerable amount of bandwidth and could threaten unfairness to competing TCP traffic. The issue of fairness for adaptive-rate VoIP is somewhat unclear because it has not been studied extensively. Compared to other types of traffic, adaptive-rate VoIP requires a relatively small amount of bandwidth; usually within tens of Kbps range. Adaptive-rate VoIP traffic is less likely to be a threat to other traffic. Instead, adaptive-rate VoIP traffic could be affected by other traffic. In TCP-dominant networks, the AIMD algorithm creates a race condition in which TCP flows compete to get their share of bandwidth. One key design element for adaptive-rate control is that the increase/decrease algorithm must, likewise, be equipped with adequate aggressiveness in order to compete with TCP traffic. If the algorithm is less aggressive than the AIMD algorithm, the adaptive-rate VoIP flow might not be able to get its needed bandwidth. Since VoIP is delay-sensitive and has a strict delay requirement, we believe that the increase/decrease algorithm should give a higher priority to optimizing the performance.

This is because we could not optimize both the performance and fairness at the same time. Nevertheless, this question remains open to future research.

4.3 Sync & Sense Enabled Adaptive Packetization VoIP

We now present our novel adaptive-rate VoIP called Sync & Sense enabled adaptive packetization VoIP. This work incorporates the knowledge we have gained about optimizing packetization, in Chapter 2, which can help to minimize delay and loss. And, it incorporates the Sync & Sense measurement methodology, proposed in Chapter 3, which provides a full range of observations about the state of the network. We design a new adaptive-rate control that (1) uses packetization as a means for rate adaptation and (2) makes use of the Sync & Sense methodology. We thoroughly consider design choices and desirable properties, as mentioned in the previous section. We then use simulations to refine the design and test validity of the assumptions. In the following, we present Sync & Sense enabled adaptive-rate control. We then discuss important issues of implementing Sync & Sense in an adaptive packetization environment. Finally, we discuss jitter buffer management in which the jitter buffer needs to adapt in accordance with varying packetization.

4.3.1 Sync & Sense Enabled Adaptive-rate Control

Let us begin with the choice of rate adaptation because the design of an adaptive-rate control relies on the speech coder being used. We choose coarse-grain rate adaptation because it is supported by any speech coder. This means, our adaptive-rate control can work with any speech coder. With Sync & Sense, the receiver can monitor the arriving packets for observations about the state of the network. It is clear that the control should be placed on the receiver. This allows the control to process the observations immediately and continuously. In this case, whenever there is a need to adjust the transmission rate, the control immediately sends a feedback command to the sender. This allows a prompt reaction to the changes in the network. This also helps to limit the feedback traffic to the minimal. We certainly cannot rely on the RTCP feedback report, which is sent out periodically in the time scale of seconds. In our design, we assume that the sender must have a client-server UDP port opened for accepting the command feedback packets, in addition to the standard ports for RTP and

RTCP. We may also use a TCP port instead, in order to provide reliable transport of the command feedback.

As for the decision metrics, our design assumes the traditional wired packet-based network. We believe that our adaptive VoIP should first and foremost be developed and proven to work under this assumption. In future research, we can focus on improving its capability toward heterogeneous environments, including wireless networks. The assumption of the traditional network is that random loss due to bit errors is very rare. Therefore, packet loss is an indication of network congestion, and it can be an effective decision metric. Given that Sync & Sense provides measurements of one-way network delay, one may be tempted to have the control make a decision based on the delay budget. Specifically, ITU-T Recommendation G.114 [17] specifies the one-way transmission time of 150 milliseconds or less for acceptable voice quality. However, because a VoIP flow typically has a less significant traffic impact, the network delay is often and largely determined by the underlying network traffic. It is more possible for the VoIP flow to optimize variations of the delay, rather than reducing the delay. We use the trend of one-way network delay as another decision metric because it allows the control to infer network conditions beyond congestion as indicated by packet loss. In addition, we can take advantage of the studies that there is a correlation between one-way network delay and packet loss [36, 37, 38, 39]. Thus, observing the correlation can give an idea of the upper bound delay in which packet loss is likely, where we denote as the delay threshold. While packet loss is a reactive metric, the delay trend and delay threshold provides a proactive metric in which the control can respond sooner to prevent potential packet losses.

The delay threshold can be estimated from the history of the observations of network delay and packet loss event. A packet loss event could be a single packet loss or a burst of consecutive packet losses. Initially, when the session starts, the control has no delay threshold. When packet loss events occur, the delay observations of the previous arriving packet are recorded. These delay observations are used to calculate the delay threshold. We use a common technique called the exponential smoothing method to average the delay observations. The moving average of the delay threshold can be calculated as follows:

$$DThres^{(i)} = \alpha\, DThres^{(i-1)} + (1 - \alpha)\, DLoss \qquad (4\text{-}1)$$

where DLoss is the delay observation of the previous arriving packet before the packet loss event and α is a smoothing factor ($0 < \alpha < 1$). This equation is updated every time a packet loss event is observed. The α parameter determines how the weight is distributed. We use the α value of 0.875 so that the greater weight is given to the average. With the lesser weight given to the more recent DLoss observations, it allows us to filter out the noise or variations of the DLoss observations. The delay threshold usually stays within a certain range of delays. This is due to the fact the bottleneck link contributes a significant and relative large impact to the DLoss observations. In addition, most of the packet loss events happen on the bottleneck link. The noise can be caused by traffic mix on the bottleneck link. When the majority of traffic is elephants, or large-size packets, the DLoss observations tend to be higher. On the other hand, when the majority of traffic is mice, or small-size packets, the DLoss observations tend to be lower. Also, increasing the number of hops in the path tends to add more variation to the DLoss observations. For example, packet loss events on non-bottleneck links would result in unusually low DLoss observations. As a matter of fact, the routing path may change during the session. As new packet loss events are observed, the moving average of the delay threshold can reflect the changing characteristics of the path.

As mentioned earlier, our adaptive VoIP is based on coarse-grain rate adaptation. Since only a certain number of transmission rates are available, the increase/decrease algorithm is limited to a stepwise fashion. The algorithm always starts off with the highest transmission rate, in order to test whether the network can support the best transmission mode. With the decision metrics mentioned above, the algorithm uses two strategies, running in parallel, in the decision making. The packet loss strategy runs by default. The delay threshold strategy relies on the Sync & Sense methodology. It is only available when Sync & Sense can get its synchronization running. In addition, the delay threshold strategy can start after a number of packet loss events have been observed, at which point the delay threshold is available. In terms of decision frequency, it is best to make a decision as often as possible so that the algorithm is responsive to rapid changes in the network. The nominal decision frequency is set to every one second, which is a highest possible frequency. This ensures that the decision interval is always larger than any round-trip time, which limits the maximum decision frequency.

Upon the decision interval, the algorithm makes a rate adjustment and immediately sends a command feedback to the sender. Note that, although the decision has been made, the algorithm is required by at least a round-trip time to be able to observe the effect of the rate change. That is, a half round-trip time is for the feedback command to get to the sender; and, the other half round-trip time is for the first packet with the new rate to arrive at the receiver. During this round-trip time, the algorithm continues to observe the decision metrics based on the arriving packets with the previous rate. It will be wrong to take into account any observations during this period of time. Therefore, out of one second of the decision interval, the algorithm disregards any observations for such a period of round-trip time, until receiving the first arriving packet with the new rate. We call this round-trip time the **silence period**. The algorithm has the remainder of the decision interval to evaluate the effect of the rate change.

The stepwise increase/decrease algorithm reduces its rate by one step when congestion is observed, and increases its rate by one step in the absence of congestion. The decision is made at each time interval. In coarse-grain rate adaptation, a step of rate change is usually large. A step increase in the transmission rate could be too high and overshoot the available bandwidth. In this case, it would be too slow to respond at the next decision interval. As Sync & Sense provides continuous observations for every arriving packet, we can add the following rule, called **fast response**, so that the algorithm is quick to react to the overshoot. Following a step increase in the transmission rate (after the silence period), if a negative indication (from packet loss or the delay threshold) is observed, the algorithm makes a decision immediately to lower the rate by one step, instead of waiting until the next interval. Fast response is a conservative strategy to increase the transmission rate, which is like dipping a toe into unknown water. It tries just a little bit to see if the water is too hot or too cold, and continues if the water seems to be nice and warm.

In the packet loss strategy, an observation of packet loss is considered a sign of network congestion. Given that packet loss is due to congestion, it can be classified into self-induced and independent. Self-induced packet loss means that the rate change by the algorithm can make an impact on the network, particularly the bottleneck link. Directly, an increase in the transmission may induce congestion and cause packet loss. Indirectly, the transmission rate may stay unchanged, but the increasing cross traffic induces congestion. In

this case, responding to packet loss would help to reduce congestion. On the other hand, independent packet loss means that the rate change by the algorithm has no impact on the network. This kind of packet loss may happen on a large capacity bottleneck link where the VoIP flow contributes little significant traffic, or may happen on non-bottleneck links. In this case, responding to packet loss would have no impact on congestion. For the VoIP flow, it would be better to stay with the highest transmission rate to retain the highest possible performance. Using the trend of one-way network delay and the delay threshold, it is possible for the algorithm to identify a packet loss whether or not it is self-induced. This is because self-induced packet loss is usually correlated with an increasing delay trend and a high delay threshold. This intelligence would allow the algorithm to better optimize the performance. However, we choose to use a conservative packet loss strategy. That is, the algorithm responds to every packet loss event. Another reason is that responding to packet loss is a factor that determines fairness to competing TCP flows. Since this packet loss strategy is in line with that being used by TCP, our increase/decrease algorithm would be able to demonstrate a degree of fairness to competing TCP flows. This issue, however, is not our main focus in the design because we give a higher priority to optimizing the performance. Future research would be needed to study the fairness issue.

In the delay threshold strategy, the algorithm monitors the trend of one-way network delay of the arriving packets. Because of variations in the network delay, it is necessary to smooth the trend and get rid of the spikes. We use the same technique of exponential smoothing average, as in the calculation of the delay threshold. With every arriving packet, the algorithm updates the delay trend using the following equation:

$$\mathrm{DTrend}^{(i)} = \alpha\, \mathrm{DTrend}^{(i-1)} + (1 - \alpha)\, \mathrm{DNet} \qquad (4\text{-}2)$$

where DNet is the delay observation of the arriving packet. The smoothing factor α is set to 0.875. This smoothing method has also been used in TCP to calculate the smooth round-trip time estimate. Setting the α parameter between 0.8 and 0.9 is suggested in the TCP specification [30] to spread out the weight efficiently. Upon the decision interval, the algorithm checks to see if the delay trend has crossed the delay threshold. Specifically, there are four possible scenarios: (1) the delay trend stays below the delay threshold, (2) the delay

trend has crossed above the delay threshold, (3) the delay trend has crossed below the delay threshold, and (4) the delay trend stays above the delay threshold.

In the first scenario, the delay trend stays below the delay threshold: $DTrend^{(i-1)} < DThres$ and $DTrend^{(i)} < DThres$. This indicates that the network has sufficient available bandwidth to accommodate the current transmission rate and there is no congestion, particularly on the bottleneck link. The algorithm thus increases the transmission rate by one step, with an attempt to achieve greater performance.

In the second scenario, the delay trend has crossed above the delay threshold: $DTrend^{(i-1)} < DThres$ and $DTrend^{(i)} \geq DThres$. This indicates that congestion is building up and packet loss is imminent. The algorithm thus reduces the transmission rate by one step, in order to combat congestion. However, the rate reduction may or may not help to reduce congestion. It would help if the congestion is self-induced. Namely, the VoIP flow makes a significant traffic on the bottleneck link. In this case, the increasing delay trend could be caused by the recent rate increase or an increase in the cross traffic. It would not help if the congestion is independent. That is, the bottleneck link has a large capacity in which the rate reduction is relatively too little to make any impact on the congestion.

In the third scenario, the delay trend has crossed below the delay threshold: $DTrend^{(i-1)} \geq DThres$ and $DTrend^{(i)} < DThres$. This indicates that network congestion has subsided. If the congestion is self-induced, this would be an effect of the recent reduction in the transmission rate. Otherwise, the amount of cross traffic might have decreased. Thus, the algorithm increases the transmission rate by one step, with an attempt to achieve greater performance.

In the last scenario, the delay trend stays above the delay threshold: $DTrend^{(i-1)} \geq DThres$ and $DTrend^{(i)} \geq DThres$. This indicates an ongoing congestion. The conventional wisdom would have been to reduce the transmission rate to combat congestion. But, this strategy has a problem and has proven to be ineffective in our experiments. This is because the interaction among competing flows also has an impact on the behavior of the flows, particularly in a low statistical multiplexing environment. TCP flows typically create a race condition in the network in which they compete for their shares of bandwidth. The AIMD algorithm used in TCP has some degree of aggressiveness in acquiring the available bandwidth. It gradually increases the window size (or the transmission rate) until a packet

loss is observed; then, reduces the window size by half in response to the packet loss. In this scenario, reducing the transmission rate could help to reduce congestion, but it is in a wrong direction. This is because it is releasing available bandwidth to the competing TCP flows. TCP data packets normally have a large packet size, which can cause large queuing delay. As the competing TCP flows gain greater shares of bandwidth, the algorithm would observe higher increasing delay trend. When the algorithm responds by reducing the rate, it would observe even higher increasing delay trend. This vicious cycle would eventually cause the algorithm to operate at the minimum transmission rate. Therefore, while the objective is to reduce congestion, at the same time the algorithm must maintain its aggressiveness to compete for its share of bandwidth. In this scenario, the algorithm, instead, increases the transmission rate by one step. By doing so, the competing TCP flows would not be able to advance the window size much further because packet loss happens sooner. This balance of power allows all competing flows to get their needed shares of bandwidth. The strategy to increase the transmission rate also has a benefit to allow a more accurate delay threshold. The noise, such as unusually low DLoss observations, caused by packet losses on non-bottleneck links could cause the delay threshold to be lower than it is supposed to be. Increasing the transmission rate allows the algorithm to push until packet losses on the bottleneck link are observed. The delay threshold can then become more accurate in providing a guideline for potential packet loss.

In summary, the delay threshold strategy allows the algorithm to test the condition of self-induced congestion. The delay trend crossing above the delay threshold is an indication of a congestion buildup. The algorithm thus responds by reducing the transmission rate. If the congestion is indeed self-induced, the rate reduction would cause the delay trend to fall below the delay threshold. Hence, this strategy can proactively prevent potential self-induced packet loss. Nonetheless, following the rate reduction, the algorithm may also observe the delay trend remains above the delay threshold. There are two possibilities. First, the cross traffic are TCP flows. The recent rate reduction allows the competing TCP flows to gain more available bandwidth, causing the delay trend to move even higher. Second, the congestion is independent. The recent rate reduction makes no impact. The delay trend remains high because it is determined by the cross traffic. The algorithm cannot distinguish between the two cases. However, the same response by increasing the transmission rate

works in both cases. It allows the algorithm to maintain its aggressiveness to compete with TCP flows. In the case of independent congestion, it allows the algorithm to operate at a better transmission mode to optimize the performance.

Figure 4-1 shows the flow chart of the adaptive-rate control algorithm.

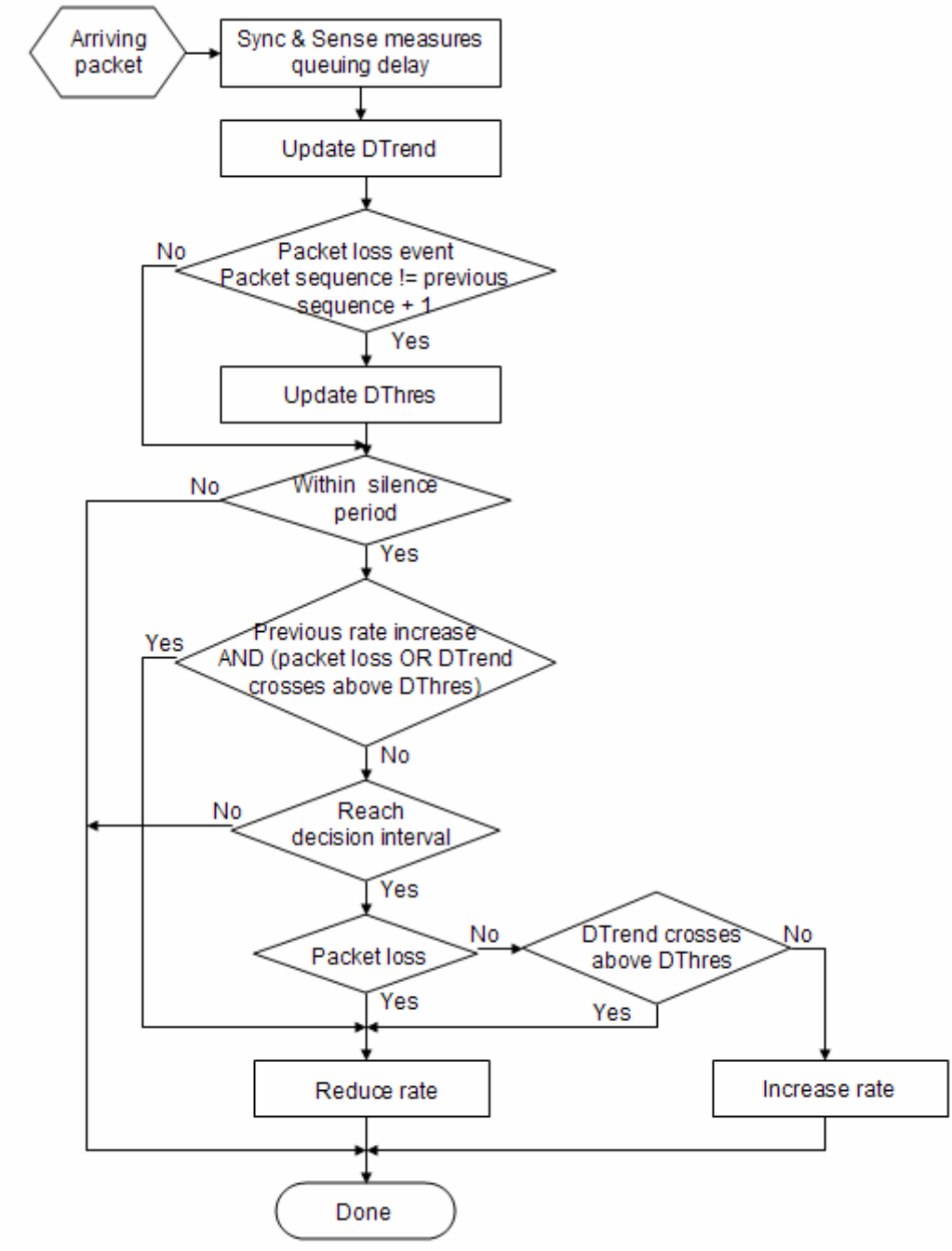

Figure 4-1 Flow chart of the adaptive-rate control algorithm

4.3.2 Sync & Sense Implementation

The Sync & Sense measurement methodology presented in the previous chapter is based on observing a periodic stream of incoming packets. However, in adaptive packetization VoIP, the periodic stream constantly changes its rate following the control decision. Here, we take a closer look at the issues that may affect Sync & Sense and how to implement Sync & Sense in an environment of adaptive packetization VoIP. Referring to Equation 3-3, in the previous chapter, Sync & Sense finds a complete epoch of excursion of queuing delays when the ongoing epoch is equal (within the error margin) to a multiple of packetization delays. In adaptive packetization VoIP, rate adaptation is done by varying packetization delay. This can affect the condition of a complete epoch of excursion. When Sync & Sense is yet to observe a complete epoch of excursion, it has no reference for the transmission timing. If the transmission rate changes (due to the packet loss strategy), Sync & Sense needs to restart the synchronizing phase and look for a new epoch. Once a complete epoch is found, the key issue is what happens when the transmission rate changes. If Sync & Sense needs to abandon the ongoing epoch and start over, this would dramatically affect and reduce the synchronization rate performance.

Figure 4-2 is a revisit of Figure 3-3, in the previous chapter, illustrating how Sync & Sense makes measurements in adaptive packetization VoIP. Starting from packet #2, the transmission rate is reduced. In other words, the packetization delay gets larger. When Sync & Sense observes an arriving packet with a changing payload size, packet #2, it can follow this rate change by advancing the transmission timing by the corresponding packetization delay. The intervals between the dashed lines on the receiver are the constructed transmission timing. From the figure, we can see that the inter-arrival gap includes the change in packetization delay. Therefore, at the end of an epoch, the transmission and reception timing advance by the same amount of time. The figure demonstrates that, in adaptive packetization VoIP, Sync and Sense can continuously provide delay measurements, unaffected by changes in the transmission rate. For Sync & Sense to work with adaptive packetization VoIP, Equation 3-3 must be updated to:

$$\sum_{i=1}^{n} D_{pack}^{i} - \varepsilon \;\leq\; E \;\leq\; \sum_{i=1}^{n} D_{pack}^{i} \qquad (4\text{-}3)$$

The above equation basically means that Sync & Sense finds a complete epoch of excursion when the ongoing epoch is equal (with an error margin) to the sum of packetization delays.

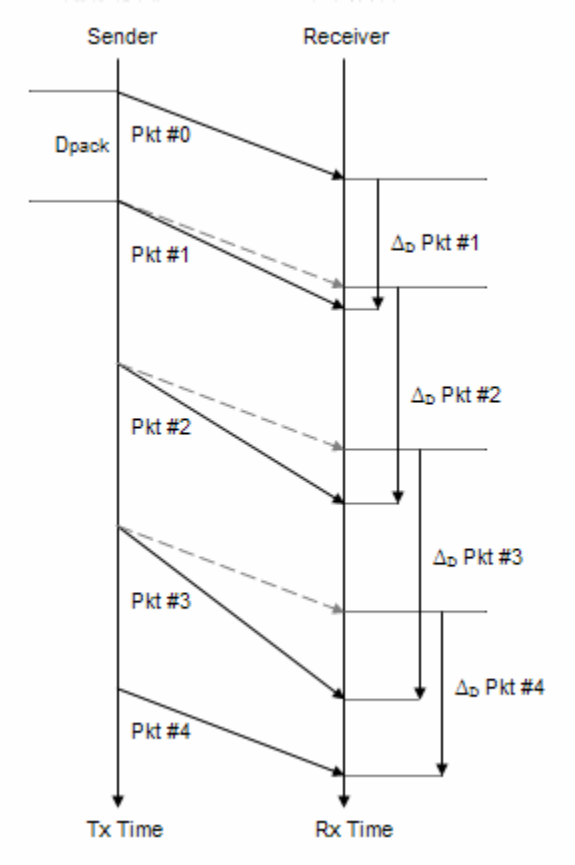

Figure 4-2 Sync & Sense measurements in adaptive packetization VoIP

In the previous chapter, Sync & Sense has proven that it is not affected by packet loss. From Figure 4-2, if packet #3 is lost, Sync & Sense can still continue to observe the inter-arrival gap. Upon receiving packet #4, Sync & Sense observes the inter-arrival gap beginning from the arrival of packet #2. This already includes the inter-arrival gap that would have been from the lost packet #3. By checking the packet sequence number, Sync & Sense knows to advance the transmission timing accordingly by adding two packetization delays. Therefore, both the ongoing epoch and the transmission timing can advance correctly in the event of packet loss. Nonetheless, in adaptive packetization VoIP, a scenario of packet loss can have an impact on Sync & Sense. From Figure 4-2, if packet #2 is lost, a problem arises. Upon receiving packet #3, Sync & Sense has no way to know whether packet #2 is transmitted

based on the old or new packetization. This problem could happen during the silence period. After a command feedback is sent to the sender, Sync & Sense is expecting incoming packets with a different packetization. It is the packet loss at the end of the silence period that causes the problem. The lost packet could be the last of the previous rate stream or the first of the new rate stream. Advancing the transmission timing with the incorrect packetization delay can affect the measurement accuracy. A solution is to abandon the ongoing epoch and restart the synchronizing phase. This, however, comes at a heavy price because it typically takes some time to re-synchronize. This reduces the synchronization rate performance. As a result, Sync & Sense would not be able to continuously provide delay measurements to the adaptive-rate control.

The answer to this problem comes down to the choice between the measurement accuracy and the synchronization rate. For adaptive packetization VoIP, the latter is more important because it allows the control to fully use the delay threshold strategy. To maintain the synchronization, when Sync & Sense observes an arriving packet with a different packetization, it assumes this packet is the first of the new rate stream. This assumption could also be wrong if this packet is actually the last of the previous rate stream. Although the result of this strategy is an error in the delay measurements, it causes relatively little impact on how the adaptive-rate control makes a decision. Since the control adjusts the transmission rate in a stepwise fashion, a step of the rate change is really a small change in packetization, for example 5 milliseconds. The size of the error is equal to the change in packetization delay, which is relatively small compared to the network delay. It is possible that the measurement error may accumulate following several consecutive wrong assumptions in this situation. But, the probability would be very low. Besides, Sync & Sense can automatically catch the error and correct it when the condition of Equation 3-6, the dispersion gap must be equal or larger than the packetization delay, is found. Lastly, the delay trend and delay threshold are compared with respect to each other. The measurement error incurred in both the delay trend and delay threshold thus does not affect the control decision.

4.3.3 Jitter Buffer Management

In packet-switched networks, packets typically arrive at the receiver with variable network delays. The variation in the packet inter-arrival times is called jitter. For VoIP, jitter must be removed so that voice frames can be decoded in a constant rate and played out smoothly. A jitter buffer does this job by storing a number of voice packets to normalize the delay variations. If a packet arrives too quickly, it will be held for a small duration of time so that it and late arriving packets can be played out at a constant rate. A typical jitter buffer deals with voice packets with fixed packetization. In adaptive packetization VoIP, additional jitter buffer management is needed to prevent buffer underflow and overflow. Note that buffer underflow and overflow can happen in any jitter buffer when the buffer is too small to accommodate the delay variations. In an underflow, the buffer is empty when the speech coder needs to play out a sample. In an overflow, the buffer is already full and another packet arrives. Such a packet thus needs to be discarded. Both cases cause gaps in the speech and degrade the voice quality. Below, we discuss a special case of buffer underflow and overflow that could happen due to the change of packetization, and a simple solution that can fix this problem.

Figure 4-3 illustrates most of the processes in VoIP, along with the time line. Line A is the time in which sample voice frames are produced by the speech coder, at a constant rate, for example, every 5 milliseconds. Each dot represents a sample voice frame. Several voice frames may be loaded into the same packet, called packetization. For example, voice frames #5 and #6 are loaded together and transmitted as a single packet. The dashed line represents transmission of the packet from the sender (line A) to the receiver (starting from line B). Line B is the minimum delay incurred on every packet traveling across the network, which includes, for the most part, propagation delay, and delays in routers and switches. The jitter buffer is represented by the time from line B to line D. Packets may arrive at any time between line B and line D due to the variation in network delay. The packets arriving on line B are early. Thus, they are held in the jitter buffer. All voice frames leave the buffer at a constant rate on line D (ignoring line C, for now), to be decoded and played out. The time from line A to line D is the playout time.

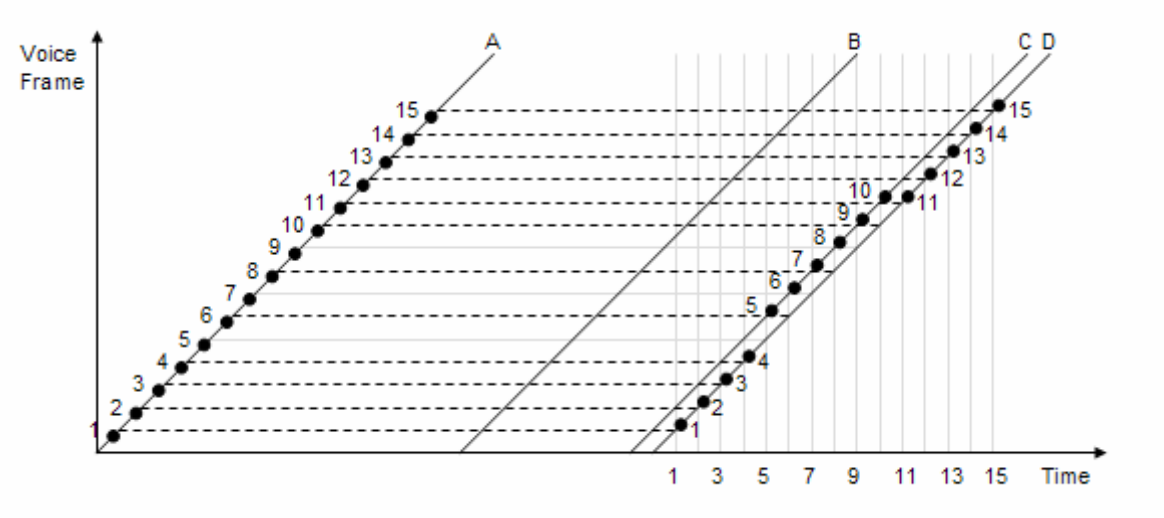

Figure 4-3 Jitter buffer management in adaptive packetization VoIP

From Figure 4-3, when packetization includes one voice frame, the generated voice frame is loaded into a packet and transmitted right away. Assume it takes 5 milliseconds for the speech coder to collect enough voice samples and produce a voice frame. Upon line D, the voice frame is unloaded from the packet and leaves the jitter buffer immediately. Thus, the packetization delay is 5 milliseconds. When packetization includes more than one voice frame, a delay artifact can be seen in the figure. For example, voice frames #5 and #6 are loaded into the same packet. It takes 5 milliseconds to generate voice frame #5. The generated voice frame #5 must wait for another 5 milliseconds for voice frame #6 to be generated. Then, both voice frames can be transmitted together as a packet. Note that, at this time, voice frame #6 incurs 5-millisecond packetization delay. Ignoring line C, for now, when the packet arrives at the receiver and reaches line D, voice frame #5 is unloaded first and leaves the buffer right away. Voice frame #6 must wait for 5 milliseconds to be unloaded. Therefore, this process of packetization (and de-packetization) incurs 10 milliseconds to both voice frames #5 and #6.

Figure 4-3 also illustrates how, during the time of packetization change, the packetization delay can cause buffer underflow and overflow. Starting on voice frame #5, the control increases packetization to 10 milliseconds. Ignoring line C, for now, if the packet containing voice frames #5 and #6 arrives too late at time 6, the packetization delay causes a buffer underflow. At time 5, there is no voice frame to be played out. Similarly, starting on

voice frame #11, the control decreases packetization back to 5 milliseconds. If the packet containing voice frames #9 and #10 arrives too late at time 10, the packetization delay causes a buffer overflow. At time 11, voice frame #10 is ready to be played out. But, at the same time, voice frame #11 arrives and needs to be played out as well. Either voice frames #10 or #11 must be discarded. The probability of encountering the above buffer underflow and overflow scenarios is very low. This is because it only happens during the time of packetization change and the packets arrive so late, just before overflow on line D. Besides, a typical network delay distribution has a long tail toward high delays. Thus, most of the packets would arrive well before line D. A rather common scenario is that the packets arrive before line D. Line C shows a jitter buffer management that must be coordinated with the control. From the figure, upon receiving the packet containing voice frames #5 and #6, the control observes the increase in packetization by 5 milliseconds. The control must reduce the jitter buffer accordingly by the size of the packetization delay; that is, from line D to line C. Voice frame #5 thus can be played out at time 5, and there is no buffer underflow. Similarly, upon receiving the packet containing voice frame #11, the control observes the decrease in packetization by 5 milliseconds. The control must increase the jitter buffer accordingly by the size of the packetization delay; that is, from line C to line D. Voice frame #10 thus can be played out at time 10, followed by voice frame #11 at time 11, and there is no buffer overflow.

4.4 Performance Study in High Statistical Multiplexing Environment

Characterizing aggregate network traffic is difficult because it is a mix of traffic flows from a variety of applications. It depends upon several factors, including the application itself, transport protocol, user behavior, etc. Whereas most traffic is carried by the TCP protocol, researchers have recently adopted a flow based approach. That is, TCP traffic can generally be categorized into two types: so-called mice and elephants [46, 47]. Mice are short-lived TCP flows in which data is sent in its entirety within the slow start phase. Without congestion control, mice are not sensitive to the bandwidth sharing imposed by TCP. Mice can start when other flows are reducing their transmission. Web-like traffic is an example of mice. On the contrary, elephants are long-lived TCP flows that extend pass the

slow start phase. Thus, elephants share the bandwidth according to the congestion control mechanism of TCP. FTP traffic is an example of elephants. As a consequence, mice and elephants, as well as UDP traffic, have a totally different behavior from a modeling point of view. This is an important issue that needs to be considered in the performance study of our adaptive packetization VoIP. Therefore, we classify our simulation into 2 cases: high and low statistical multiplexing environments.

Because of high statistical multiplexing, the aggregate competing traffic is relatively insensitive to the adaptive VoIP flow under study. This can also be a model for different circumstances. For instance, the competing traffic is composed of a large number of UDP flows. Or, the majority of the competing traffic is mice. Under this environment, the change in the transmission rate of adaptive VoIP has relatively little impact on the behavior of the competing traffic. The adaptive VoIP flow thus must be aware of network congestion and react promptly in order to optimize the delay and loss performance. In the following, we present the performance study in a high statistical multiplexing environment to evaluate our adaptive packetization VoIP in comparison to traditional VoIP. The simulation result also demonstrates how the adaptive VoIP flow behaves in this environment.

4.4.1 Simulation Setup

The performance study is conducted using the Network Simulator 2 or ns-2 [12]. The network topology for this study is shown in Figure 4-4. All nodes implement FIFO scheduling and drop-tail queuing. All links have capacity of 10 Mbps, except the link between node 9 and 10 that is the bottleneck with a limited capacity of 512 Kbps. The propagation delay along the path is set to 40 milliseconds. The cross traffic on each link creates load around 60 percent on average. For a high statistical multiplexing environment, the cross traffic can be generated using ON/OFF sources with the shape parameter of the Pareto distribution set to 1.5 [40]. Specifically, we use nine sources on each node to create the cross traffic. Packet sizes of the cross traffic are distributed following the studies in [14, 15]; 60% of the packets are 40 bytes, 25% are 550 bytes, and 15% are 1500 bytes. Note that, in terms of load distribution, about 7% of the load is 40-byte packets, 35% is 550-byte packets, and 58% is 1500-byte packets. The adaptive VoIP flow is assumed using the

ADPCM codec, in which the packetization is variable among 10, 15, 20, 25, and 30 milliseconds. The corresponding transmission rates are 64, 53.3, 48, 44.8, and 42.6 Kbps, respectively.

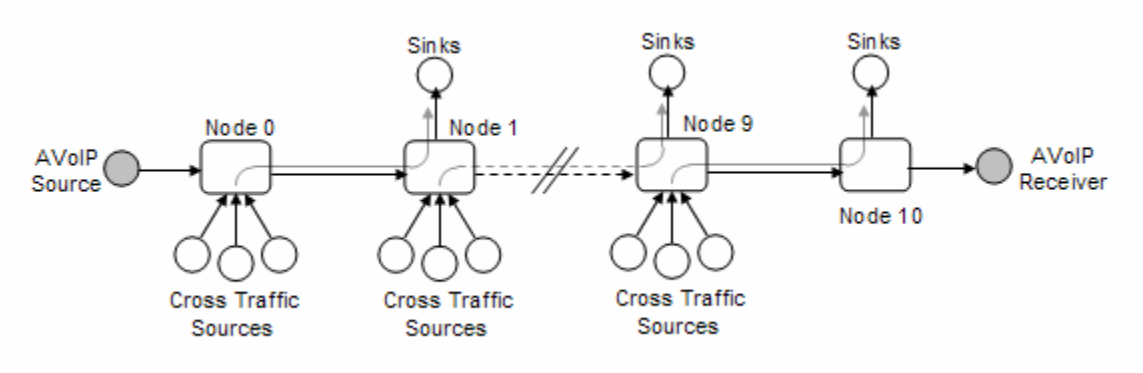

Figure 4-4 Simulation network topology for high statistical multiplexing

One-way network delay and packet loss are measured in each simulation, which runs for 120 seconds. Note that the network delay is associated with the packet level. To evaluate the performance of VoIP, the measurements must be in the voice frame level. This is because packetization delay is a factor that affects the delay of the voice frame. Instead of the network delay, we use one-way end-to-end, which is the latency of a voice frame from when the codec begins to collect the voice samples to when the voice frame is received at the receiver and begins to be decoded. The end-to-end delay can be found as the sum of the measured network delay and the corresponding packetization delay. Similarly, packet loss is associated with the packet level. A lost packet with larger packetization causes more damage than that with smaller packetization because more voice frames are lost. Thus, we consider voice frame loss, instead of packet loss. Packet loss can be converted to voice frame loss based on the corresponding packetization.

4.4.2 Network with High Statistical Multiplexing Traffic

Figure 4-5 is an instance of the simulation results of an adaptive packetization VoIP flow across a network with high statistical multiplexing traffic. We monitor several points in the network so that we can observe and demonstrate the action of the adaptive VoIP flow.

Figure 4-5 (a) shows the cross traffic on the bottleneck link. The envelope line is the average cross traffic on a larger time scale of one second. The shaded area within the envelope line is the cross traffic on a smaller time scale, which is sometimes bursty beyond the bottleneck link capacity. Accordingly, Figure 4-5 (b) shows the transmission rate of the adaptive VoIP flow in response to the changing cross traffic. Figure 4-5 (c) and (d) show the performance of the adaptive VoIP flow in terms of voice frame loss and end-to-end delay, respectively. Voice frame loss in Figure 4-5 (c) is normalized so that a packet loss with 10-ms packetization causes voice frame loss of 10. A packet loss with larger packetization causes correspondingly more voice frame loss. For example, a packet loss with 20-ms packetization causes voice frame loss of 20.

The result demonstrates that the proposed adaptive VoIP works as designed. The adaptive VoIP flow responds to every packet loss event. It is able to cautiously maintain a high transmission rate over the course of the session. The silence period strategy helps to prevent unnecessarily drastic rate reduction due to instantaneous spikes of packet loss. At time around 110 seconds, packet loss is very excessive. The result shows that the adaptive VoIP flow can quickly reduce the transmission rate to the minimum. When the congestion is clear, it can also quickly return to operate at the maximum transmission rate. The result also shows that the adaptive VoIP flow responds to the delay threshold strategy. This can be seen between time 20 and 40 seconds, for example. As seen in the figure, the fast response strategy results in the transmission rate that often surges to a higher rate and quickly drop back when a packet loss or the delay trend crossing above the delay threshold is observed. Both the delay threshold and fast response strategies help to prevent many potential packet losses.

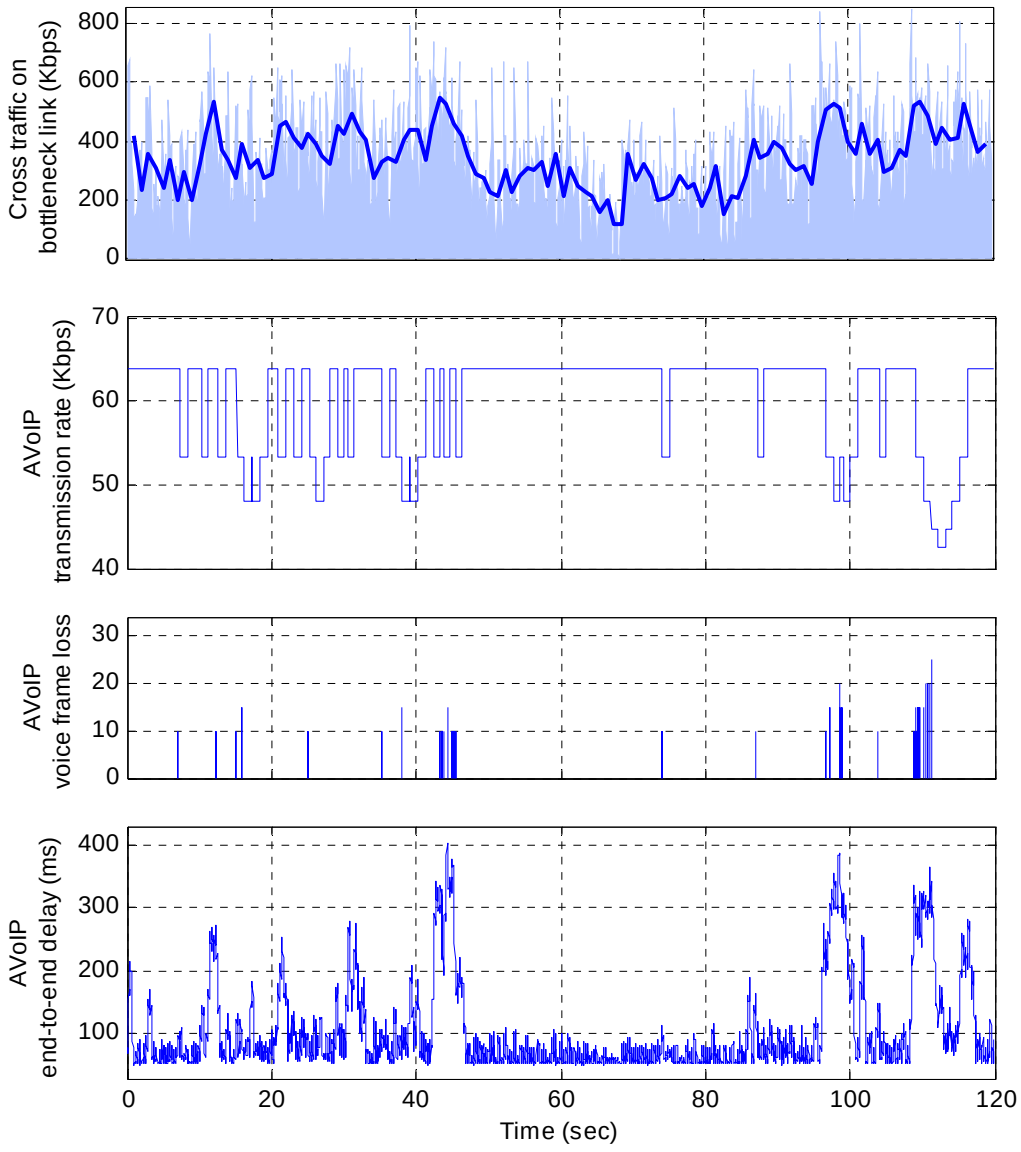

Figure 4-5 Simulation result for high statistical multiplexing

4.4.3 Comparative Performance between Adaptive VoIP and CBR VoIP

An important question is whether adaptive packetization VoIP can deliver a better performance than traditional VoIP. To answer this question, we conducted a set of simulations that compares the performance between the two. As we learned from Chapter 2, offered load is a key factor that affects the performance. In each simulation, from the network topology in Figure 4-4, we vary the offered load on the bottleneck link from 60 to 85 percent. Under the same condition, we repeat the simulation by replacing the adaptive VoIP source (and the receiver) with the following constant bit rate CBR VoIP sources: CBR10, CBR15, and CBR20 with the packetization delay of 10, 15, and 20 milliseconds, respectively. The CBR VoIP is assumed using the ADPCM codec as same as the adaptive VoIP. To characterize the performance, the measured one-way network and packet loss are converted to end-to-end delay and voice frame loss, respectively, based on the corresponding packetization of the packet. In VoIP, early arriving packets are held in a jitter buffer for a smooth playout. We use the 90^{th} percentile of end-to-end delay as a performance metric because it virtually accounts for an estimate of the jitter buffer delay. Thus, it can reflect the actual delay that the user may experience. Another performance metric is voice frame loss rate that indicates the percentage of voice frames being lost in the network.

Table 4-1 Comparative performance between adaptive VoIP and CBR VoIP

	Offered load (%)					
	60	65	70	75	80	85
AVoIP	96.45 ms 0.21%	120.84 ms 0.33%	187.57 ms 1.09%	233.28 ms 1.55%	244.29 ms 2.11%	310.87 ms 4.98%
CBR10	103.22 ms 0.24%	133.82 ms 0.41%	196.87 ms 1.42%	234.33 ms 2.05%	246.89 ms 2.63%	289.96 ms 6.19%
CBR15	101.59 ms 0.21%	128.98 ms 0.33%	202.86 ms 1.04%	245.99 ms 1.52%	261.41 ms 2.15%	321.66 ms 5.31%
CBR20	105.15 ms 0.17%	126.66 ms 0.31%	206.81 ms 0.86%	253.67 ms 1.28%	273.09 ms 1.83%	339.89 ms 4.84%

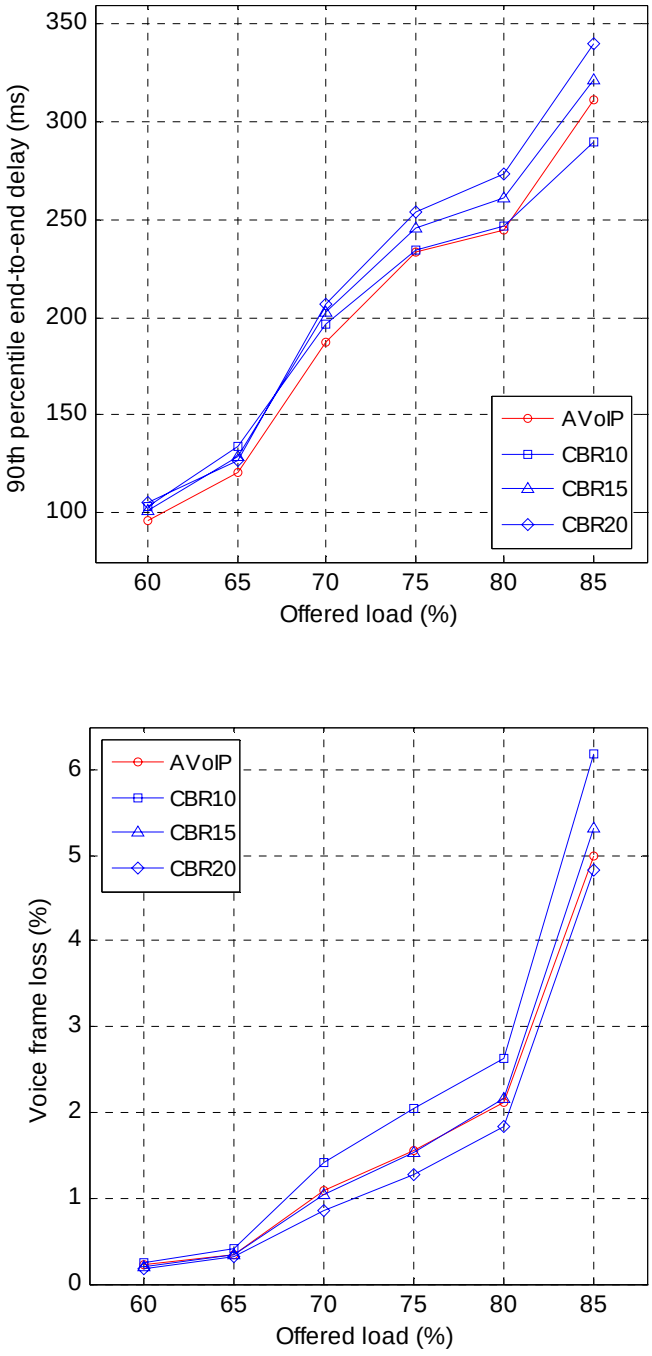

Figure 4-6 Plot of comparative performance between Adaptive VoIP and CBR VoIP

Table 4-1 shows the simulation results. Each cell in the table includes the 90th percentile of end-to-end delay on the top and voice frame loss rate on the bottom. Figure 4-6 (a) and (b) are the corresponding plots of the 90th percentile of end-to-end delay and voice frame loss rate, respectively, from the table. Comparing among the CBR VoIP sources, CBR15 and CBR20 tend to give a better performance under the offered load of 65 percent or below. This is because they require less network bandwidth than CBR10; thus, causing less congestion. Beyond the offered load of 70 percent, the available bandwidth is decreasing and very limited. Since CBR10 demands the most network bandwidth, it causes severe congestion and excessive packet loss. As a consequence of FIFO queue, the packet loss affects all traffic sharing the bottleneck link and reduces the overall network delay. The same situation happens to CBR15, and CBR20. Therefore, CBR10 gives the best end-to-end delay performance, followed by CBR15 and CBR20. On the contrary, CBR10 gives the worst voice frame loss rate, followed by CBR15, and CBR20. Although CBR15 and CBR20 give less voice frame loss, it should be noted that their voice frame losses are consecutive, based on packetization. It is usually difficult to recover voice quality from consecutive losses. Thus, the same voice frame loss rate impacts voice quality of CBR20, more than CBR15, and CBR10, respectively. The results show that the adaptive VoIP can compromise between the end-to-end delay and voice frame loss performance. Overall, the adaptive VoIP appears to give an optimal performance because its flexibility in adapting the transmission rate helps to minimize network congestion. Beyond the offered load of 75 percent, the adaptive VoIP rarely operates at the maximum transmission rate, using 10-ms packetization. As the effect of packet loss, CBR10 still gives a better end-to-end delay performance, but a worse voice frame loss rate.

Although it may seem that the adaptive VoIP does not give a significant advantage over CBR VoIP, the following should be noted. In this simulation, the VoIP flow does not make a significant traffic over the large bottleneck link. Thus, the performance is largely determined by the overall traffic load. Higher performance gain can be expected when the VoIP traffic is more significant. The advantage of the adaptive VoIP is its flexibility to adapt to the optimal transmission for the current network condition, and balance the delay and loss performance. Inflexibility is the problem for traditional CBR VoIP because the network load constantly changes.

4.5 Performance Study in Low Statistical Multiplexing Environment

By low statistical multiplexing environment, we mean that the competing traffic is somewhat sensitive to the adaptive VoIP flow under study. In addition, the behavior of each individual flow also has an impact on the other flows. This can be a model for circumstances, for instance, in which elephants are a majority of the competing traffic. In this environment, several aspects other than the delay and loss performance need to be studied. The adaptive VoIP must demonstrate that it has the capability to compete with other flows to acquire its share of network bandwidth. In addition, it should provide a fair bandwidth allocation when adaptive VoIP flows share the same network. Although TCP-friendliness is not a major concern in our design, the adaptive VoIP must not be unresponsive, but rather be friendly to the competing TCP flows, to a certain extent. In the following, we present the performance study that examines the issues mentioned above. The simulation compares the performance of the adaptive VoIP with traditional VoIP, as well as demonstrates how the adaptive VoIP interacts with other traffic types.

4.5.1 Simulation Setup

An objective of this performance study is to investigate the interaction among traffic flows that compete for the shared network bandwidth. To achieve such an objective, it is necessary to use a simple bottleneck network topology. This allows the interpretation of the results to be more straightforward and reliable. All the simulations are based on the Network Simulator 2 or ns-2 [12], with the network topology shown in Figure 4-7. The adaptive VoIP source stays in all simulations, and is assumed using the ADPCM codec with the packetization variable among 10, 15, 20, 25, and 30 milliseconds. The corresponding transmission rates are 64, 53.3, 48, 44.8, and 42.6 Kbps, respectively. The other traffic sources vary based on the study cases. All nodes implement FIFO scheduling and drop-tail queuing. The bottleneck link has a limited capacity, depending on the study cases, ensuring to create contention among traffic flows. The propagation delay on the bottleneck link is set to 5 milliseconds.

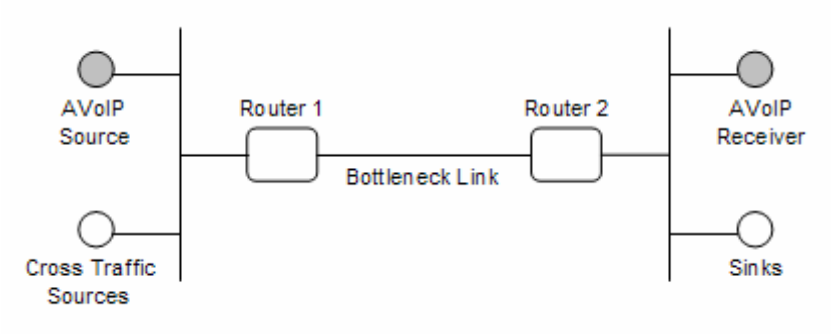

Figure 4-7 Simulation network topology for low statistical multiplexing

4.5.2 Comparative Performance between Adaptive VoIP and CBR VoIP

TCP is the most common protocol used in packet-switched networks. Any developed protocol thus must be tested with TCP to ensure that it can work in the most common environment. In this section, we examine how the adaptive VoIP interacts with TCP as well as study the performance, in comparison with traditional VoIP. This set of simulations is based on the network topology in Figure 4-7, where the other traffic source is an FTP agent running on the TCP Reno protocol. The FTP packet size is set to 1040 bytes. The long session of FTP ensures that the TCP congestion control is fully performed. Since TCP constantly attempts to acquire the available bandwidth, the bottleneck link capacity of 256 Kbps is sufficient to ensure contention. In the simulation, the adaptive VoIP must defend its required bandwidth from 42.6 to 64 Kbps. To compare the performance with traditional VoIP, we repeat each simulation by replacing the adaptive VoIP source (and the receiver) with the following constant bit rate CBR VoIP sources: CBR10, CBR15, and CBR20 with the packetization delay of 10, 15, and 20 milliseconds, respectively. The CBR VoIP is assumed using the ADPCM codec as same as the adaptive VoIP.

Figure 4-8 is the simulation result showing the interaction in the case of the adaptive VoIP; where (a), (b), and (c) is the adaptive VoIP transmission rate, voice frame loss and end-to-end delay, respectively; and (d) and (e) is the FTP average transmission rate on a 1-ms time scale and TCP window size, respectively. Note that voice frame loss in the figure is normalized so that a packet loss with 10-ms packetization causes voice frame loss of 10. A packet loss with larger packetization causes correspondingly more voice frame loss. For example, a packet loss with 20-ms packetization causes voice frame loss of 20.

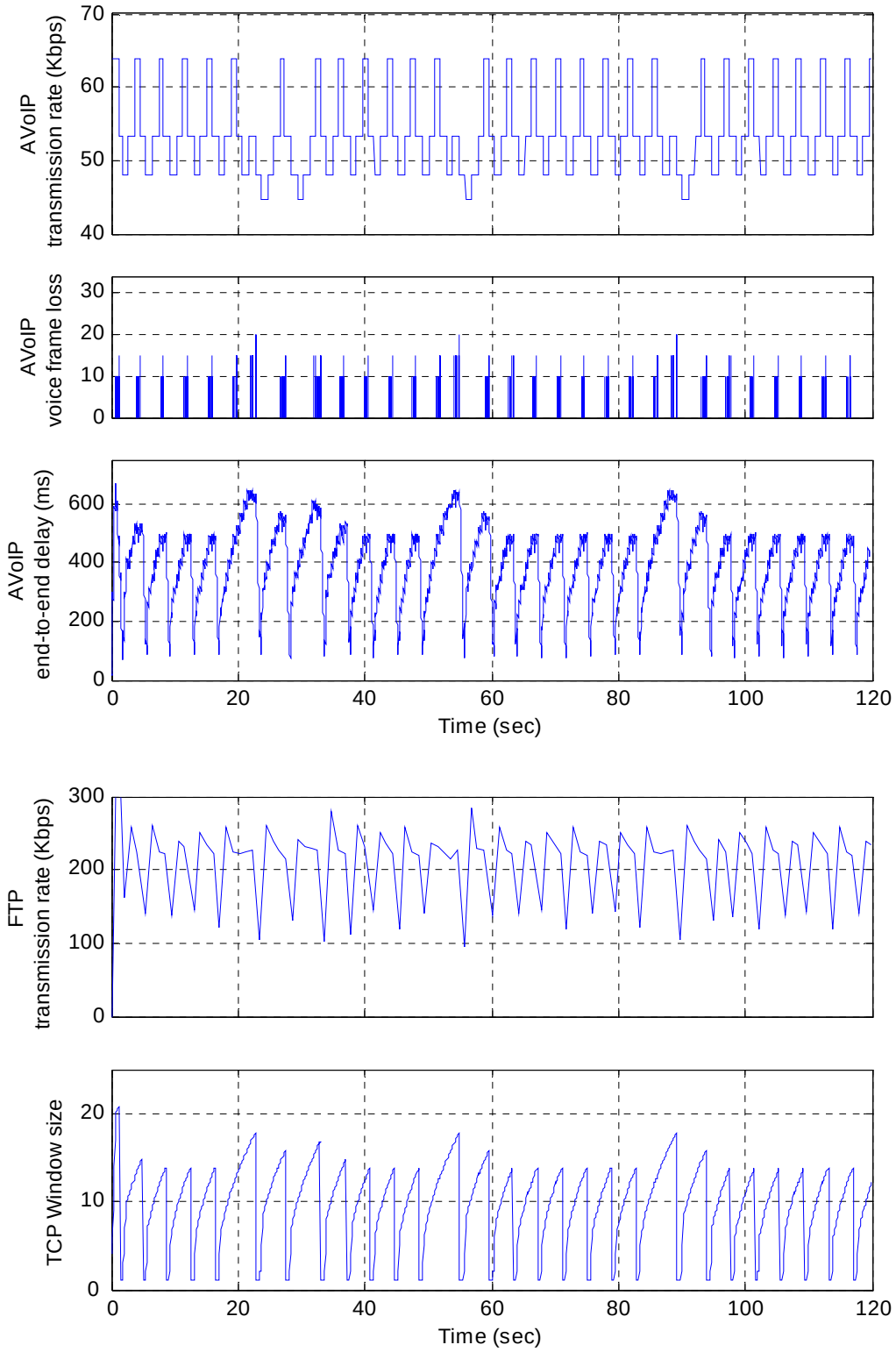

Figure 4-8 Interaction between adaptive VoIP and TCP flows

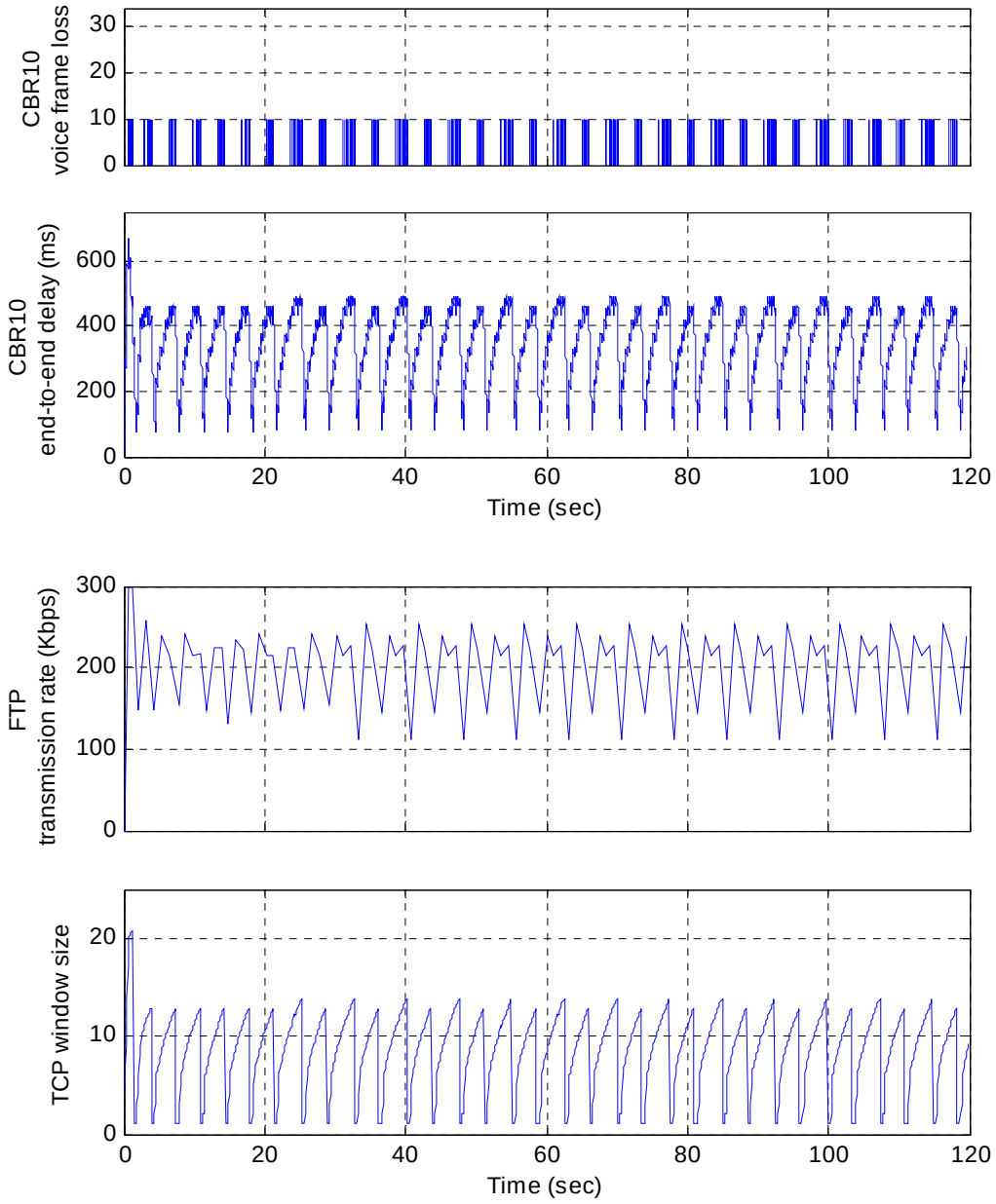

Figure 4-9 Interaction between CBR10 VoIP and TCP flows

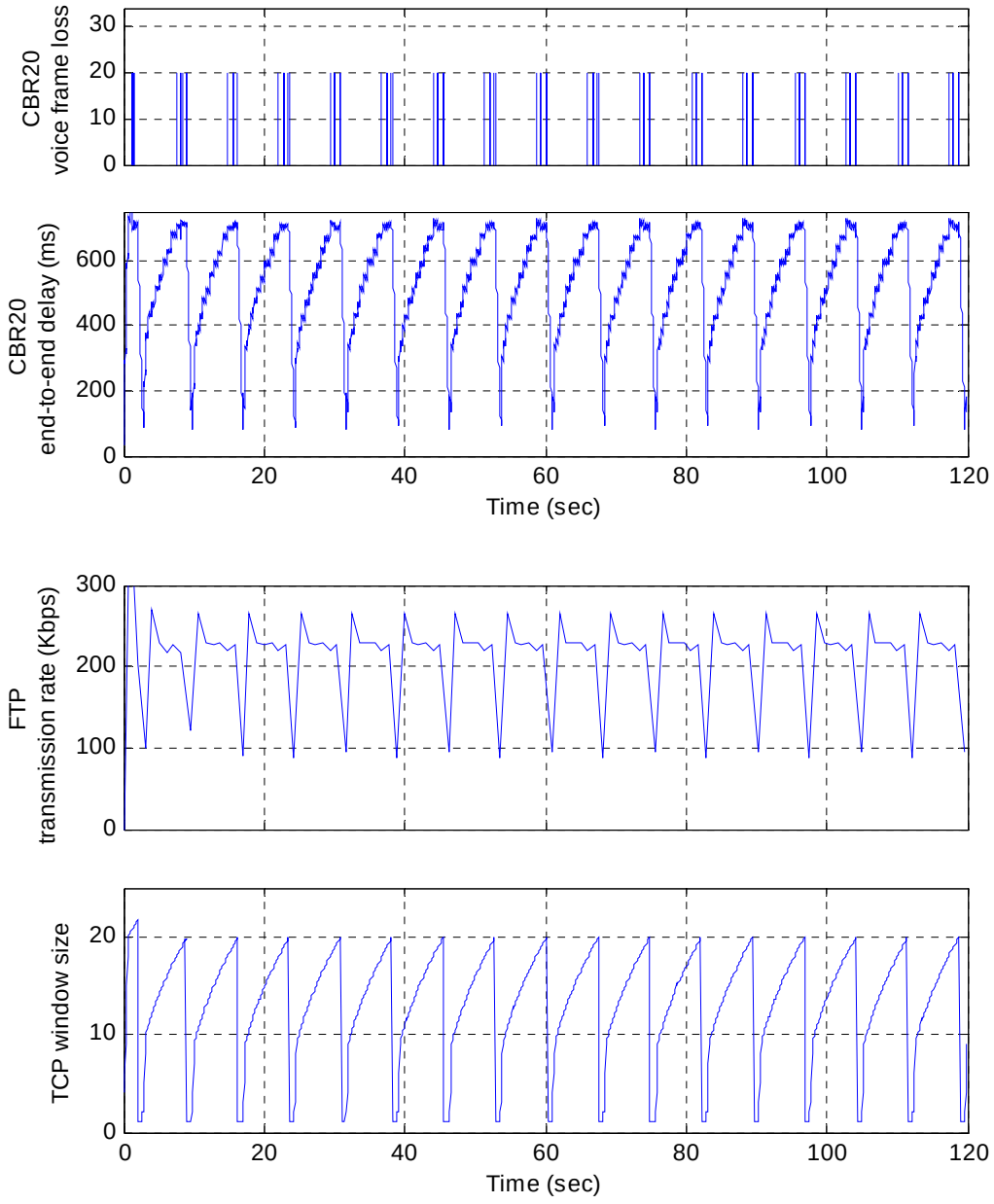

Figure 4-10 Interaction between CBR20 VoIP and TCP flows

The result in Figure 4-8 shows that the TCP control mechanism is the main driving force of the interaction. The TCP window size, in Figure 4-8 (e), shows how TCP progresses to acquire the available bandwidth, starting from the slow start phase (exponential increase) and followed by the congestion avoidance phase (additive increase). When TCP hits the maximum available bandwidth, a packet loss timeout occurs. The next cycles of the TCP window size then repeat over and over. This TCP behavior drives the adaptive VoIP such that the voice frame loss is mostly periodic and the end-to-end delay has a saw-like shape, as seen in Figure 4-8 (b) and (c), respectively. Accordingly, the adaptive VoIP and TCP transmission rates peak about the same time. That means both the adaptive VoIP and TCP observe packet loss about the same time as well. The adaptive VoIP is quick to respond to packet loss, based on the fast response strategy. At the same time, the TCP window size starts over at one. This results in a relatively synchronized transmission rate between the adaptive VoIP and TCP, as seen in Figure 4-8 (a) and (d), respectively.

At time around 25, 55, and 85 seconds, for example, it can be seen from Figure 4-8 (e) that the TCP window size advances more than other times. At the same times, the adaptive VoIP reduces the transmission rate. This is a result of asymmetrical observation that two competing flows do not always infer the same network condition at the same time. During these times, TCP does not observe packet loss, but the adaptive VoIP does. Hence, TCP can gain more bandwidth, while the adaptive VoIP loses its bandwidth share. When large packets from TCP occupy more of the bottleneck queue, it increases the delay to the adaptive VoIP. While the increasing delay triggers the delay threshold, the adaptive VoIP learns that reducing the transmission rate does not help to improve the delay trend; instead, it is worse. Therefore, increasing the transmission is a better move to maintain the aggressiveness against the competing flow. As can be seen in Figure 4-8 (a), right after the incidents, the adaptive VoIP is able to reclaim its necessary bandwidth share. Without the delay threshold strategy, the adaptive VoIP could have been trapped to operate at the minimum transmission rate. Given that, it would be difficult for the adaptive VoIP to compete with TCP and reclaim its bandwidth share.

Similar to Figure 4-8, Figure 4-9 and 4-10 are the results showing the interaction in the case of the CBR10 and CBR20 VoIP, respectively; where (a) and (b) is the CBR VoIP voice frame loss and end-to-end delay, respectively; and (c) and (d) is the FTP average

transmission rate on a 1-ms time scale and TCP window size, respectively. The results show that the CBR VoIP voice frame loss and end-to-end delay is completely determined by the behavior of TCP. There is basically no interaction because the CBR VoIP has no control mechanism. The TCP window size, in Figure 4-9 (d) and Figure 4-10 (d), can repeat its progress cycle in a periodic pattern. This, in turn, results in the CBR VoIP having a periodic pattern of voice frame loss and saw-like shape end-to-end delay, as seen in Figure 4-9 (a), (b), and Figure 4-10 (a), (b). The only difference between the cases of CBR10 and CBR20 is that they have a different bandwidth requirement. CBR10 VoIP requires 64 Kbps. When competing for shared bandwidth, TCP observes packet loss timeouts sooner. As TCP backs off more often, the bottleneck queue is less filled with large TCP packets. As a result, CBR10 VoIP has a lower end-to-end delay. This, however, comes at a price. Because packet losses occur more often, CBR10 VoIP has a higher voice frame loss. On the other hand, CBR20 requires 48 Kbps. When competing for shared bandwidth, TCP observes packet loss timeouts later. As TCP backs off less often, the bottleneck queue is more filled with large TCP packets. As a result, CBR20 VoIP has a higher end-to-end delay. Because packet losses occur less often, CBR20 VoIP has a lower voice frame loss.

Table 4-2 Comparative Performance between adaptive VoIP and CBR VoIP

	90^{th} Percentile End-to-End Delay (ms)	Voice Frame Loss Rate (%)
AVoIP	524.25	3.556
CBR10	471.38	4.858
CBR15	593.72	2.737
CBR20	706.23	1.750

Table 4-2 shows comparative performance of the 90^{th} percentile of end-to-end delay and voice frame loss rate among the adaptive VoIP, CBR10, CBR15, and CBR20, when competing with TCP. The result suggests that, in the case of the CBR VoIP, sending more packets (such as CBR10) allows the CBR VoIP to get a greater share of the bottleneck

bandwidth. Because it has no control mechanism, the CBR VoIP flows are unresponsive. The consequence of sending more packets, when competing with TCP, is greater packet losses. Packet loss does not only affect the CBR VoIP itself, but also affects the TCP throughput. That is a reason why researchers are promoting TCP-friendliness to minimize the impact from unresponsive CBR flows. The dilemma of VoIP is that both delay and loss affect the overall performance. As seen from the table, having a lower delay results in a higher loss and vice versa. This is particularly the case for the CBR VoIP because it uses a fixed packetization. The adaptive VoIP has the flexibility to minimize congestion and, at the same time, maintain aggressiveness. This results in a balance between the delay and loss that gives an optimal performance.

4.5.3 Heterogeneous Network with TCP Flows

In this section, we extend the previous simulation to a more realistic environment, where many TCP flows share the bottleneck link. The adaptive VoIP flow under study is expected to encounter more stress in this environment. This is because the increasing number of TCP flows creates more contention in sharing the limited resource. An interesting question to study is whether the adaptive VoIP is able to maintain its aggressiveness to compete with TCP and to not easily get trapped to operate at the minimum transmission rate. We use the same network topology in Figure 4-7, where the other traffic sources include four FTP agents, based on the TCP Reno protocol. The bottleneck link capacity is set to 512 Kbps. Figure 4-11 is the simulation result showing the interaction among all the competing flows. Figure 4-11 (a) shows the adaptive VoIP transmission rate. Figure 4-11 (b), (c), (d), and (e) shows the average transmission rate on a 1-ms time scale of FTP0, FTP1, FTP2, and FTP3, respectively.

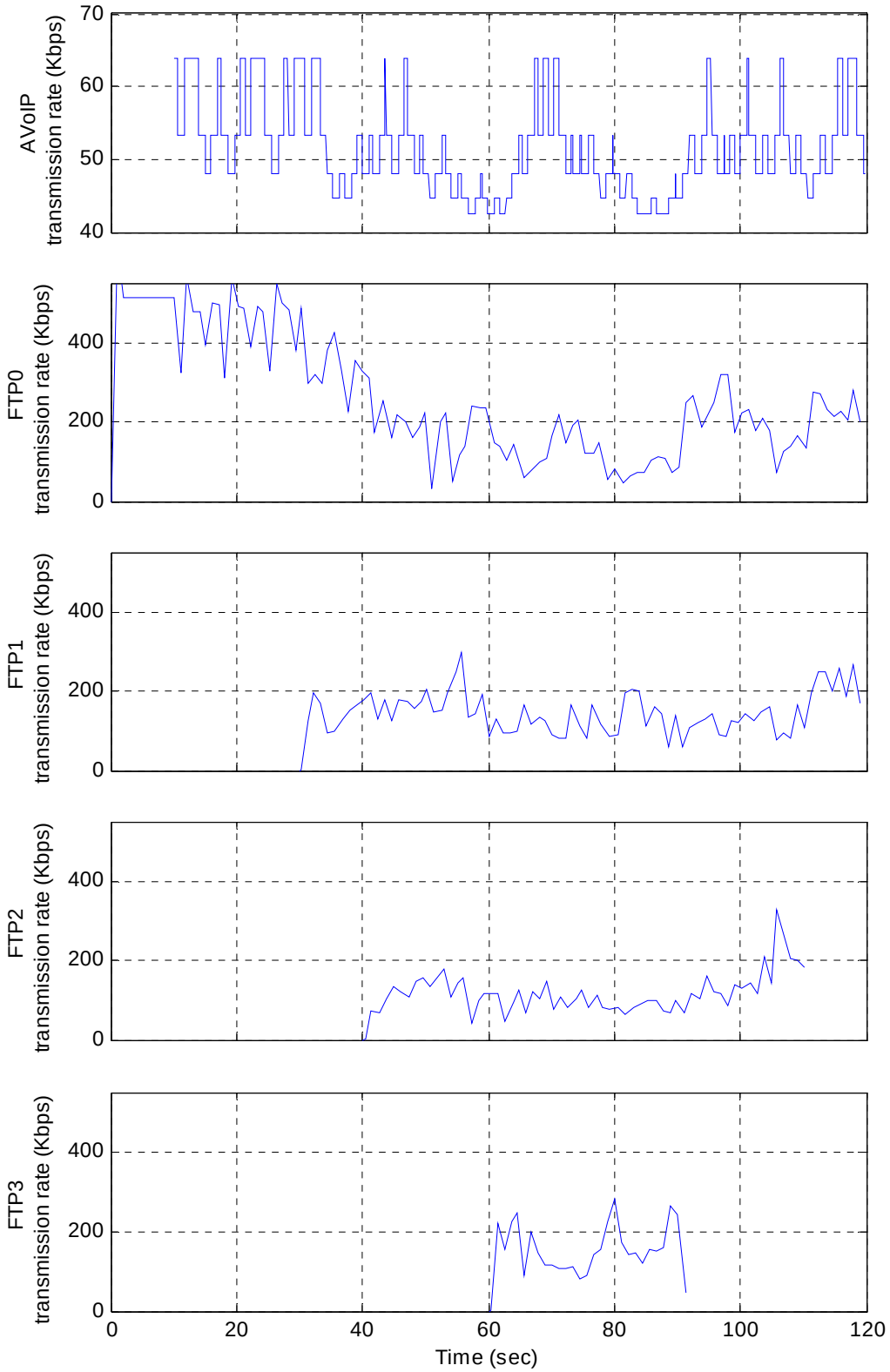

Figure 4-11 Heterogeneous network with TCP flows

From the figure, FTP0 starts off at time zero and gets the full bandwidth capacity. The adaptive VoIP session begins at time 10 seconds. It can operate at higher transmission rates, which oscillate between 64 and 48 Kbps. The rate oscillation is due to the interaction with TCP, as described in the previous section. At time 30 and 40 seconds, FTP1 and FTP2 begin their sessions, respectively. The adaptive VoIP now competes with three TCP flows. The transmission rate of FTP0 drops quickly, as new FTP sessions started. At equilibrium, the transmission rates of FTP0, FTP1, and FTP2 converge by which each gets an average bandwidth share around 160 Kbps. As seen from the figure, the adaptive VoIP responds to the decreasing available bandwidth as well, by operating at a lower transmission rate than the previous period, with one competing TCP flow. When FTP3 starts at time 60 seconds, with four competing TCP flows, the adaptive VoIP further reduces the transmission rate to the minimum. The result shows that, from time to time, the adaptive VoIP is able to increase the transmission rate to maintain its aggressiveness. The adaptive VoIP appears not to get trapped to operate at the minimum transmission rate, even when competing with many TCP flows. From time 90 seconds onward, FTP3 and FTP2 end their sessions, respectively. FTP0 and FTP1 converge to a new equilibrium with the transmission rate around 250 Kbps. As more bandwidth becomes available, the adaptive VoIP can also quickly move to operate at a new equilibrium with a higher transmission rate. This simulation demonstrates that the adaptive VoIP has the aggressiveness to compete in a TCP-dominant network, in order to acquire its needed share of bandwidth. At the same time, the adaptive VoIP is responsive to network congestion and also TCP-friendly. As more TCP flows share the same network, the adaptive VoIP tends to operate at a lower transmission rate.

4.5.4 Heterogeneous Network with CBR VoIP Flows

The problem of unresponsive flows is a current research topic, particularly in the area of TCP-friendliness. Unresponsive flows do not use any congestion control. So, they do not respond to network congestion. This behavior can result in both unfairness and congestion collapse. The impact of unfairness is that well-behaved responsive flows, such as TCP, could encounter bandwidth starvation. As the adaptive VoIP implements a congestion control, it could face the same problem as TCP. When competing with unresponsive flows, the adaptive VoIP would be likely to operate at a lower transmission rate. This is simply because the adaptive VoIP responds to packet losses, but the unresponsive flows do not. In this section, we investigate the above mentioned issue. We examine the interaction between the adaptive VoIP and CBR VoIP, as well as study how the adaptive VoIP is affected by the unresponsive CBR VoIP. We use the same network topology in Figure 4-7, where the other traffic source is a 64Kbps CBR VoIP source, based on the ADPCM codec with 10-ms packetization. The adaptive VoIP source is based on the same codec with variable packetization, giving the transmission rates among 64, 53.3, 48, 44.8, and 42.6 Kbps. Because both sources require bandwidth of 64 Kbps at most, the bottleneck bandwidth must be less than 128 Kbps in order to create congestion. Thus, the bottleneck link capacity is set to 110 Kbps.

Figure 4-12 is the simulation result showing the interaction between the adaptive VoIP and CBR VoIP. Figure 4-12 (a), (b), and (c) shows the adaptive VoIP transmission rate, voice frame loss, and end-to-end delay, respectively. Figure 4-12 (d) and (e) shows the CBR VoIP voice frame loss and end-to-end delay, respectively. Note that voice frame loss in the figure is normalized so that a packet loss with 10-ms packetization causes voice frame loss of 10. A packet loss with larger packetization causes more voice frame loss as the basis of 10. For example, a packet loss with 20-ms packetization causes voice frame loss of 20.

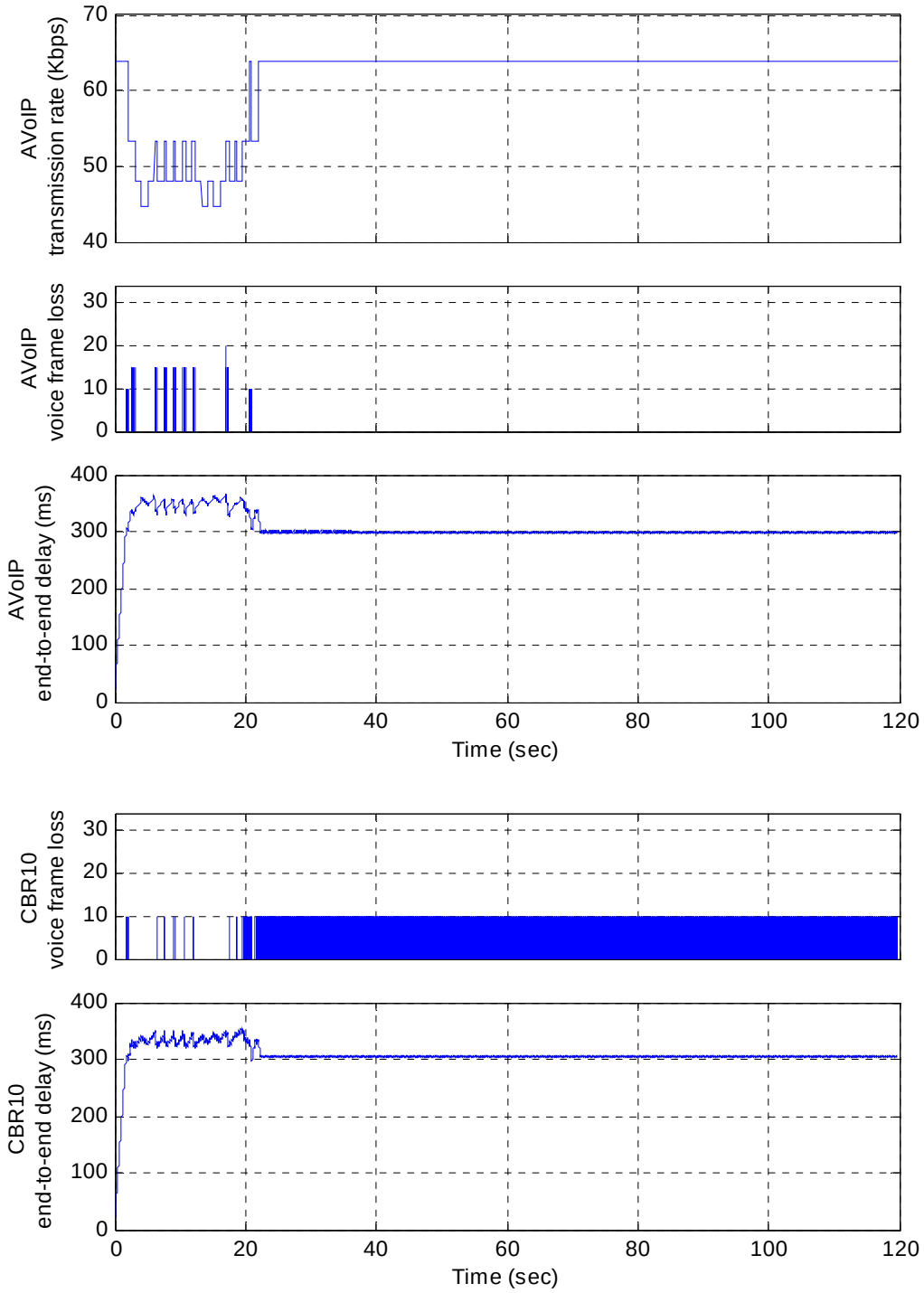

Figure 4-12 Heterogeneous network with CBR VoIP flows

The figure shows an especially interesting result. The adaptive VoIP can operate at the maximum transmission rate without any voice frame loss, after time around 20 seconds. At the same time, with its fixed rate of 64 Kbps, the CBR VoIP experiences excessive voice frame loss. It could have been the reverse. With congestion control, the adaptive VoIP would operate at a lower transmission rate. The following analysis explains what happens. At the beginning, both flows start off with the transmission rate of 64 Kbps. The aggregate transmission rate overflows the bottleneck link of 110 Kbps. Both flows thus experience packet losses. Whereas the CBR VoIP is unresponsive, the adaptive VoIP responds to packet losses and quickly reduces its transmission rate to 44.8 Kbps. As a result, the aggregate transmission rate is less than the bottleneck link capacity and the congestion is clear. Soon, this allows the adaptive VoIP to observe no indication of congestion and starts to increase the transmission rate again. For the first 22 seconds, the process of reducing and increasing the transmission rate repeats over and over as the adaptive VoIP attempts to maintain its aggressiveness to acquire its needed share of bandwidth.

At time around 20 seconds, the adaptive VoIP has an opportunity to operate at the maximum transmission rate of 64 Kbps. This is probably because of the recent shuffling of the transmission rate, which also causes the CBR VoIP to experience packet loss at the moment. When both flows transmit packets at 64 Kbps, the packet drop rate at the bottleneck link can be simply calculated as 18 Kbps. With the same packet size of 80 bytes, the drop rate is 28.125 packets per second. In other words, one packet is dropped every 35.55 milliseconds. At 64 Kbps, both sources transmit a packet every 10 milliseconds. In the FIFO queue, packets from both sources interleave to enter the bottleneck link. So, both sources would have an equal chance of their packets being dropped. However, the shuffling of the transmission rate allows the adaptive VoIP to transmit packets in the order in which it never observes packet loss. On the other hand, this causes the CBR VoIP to transmit packets that are synchronized with the packet drop rate. Because of its fixed transmission rate, the CBR VoIP continues to encounter excessive packet loss, without an opportunity to change the outcome.

This simulation shows that the adaptive VoIP, just like TCP, faces unfairness when sharing the network with unresponsive CBR VoIP flows. The adaptive VoIP needs to operate at a lower transmission rate in order to avoid congestion, and could encounter bandwidth

starvation. Nevertheless, flexibility is the strength of the adaptive VoIP. In a network with CBR traffic, adapting the transmission rate could result in the shuffling effect that allows the adaptive VoIP to be in a better position to acquire its needed share of bandwidth.

4.5.5 Homogeneous Adaptive VoIP Network

In the previous sections, we have seen how the adaptive VoIP interacts with TCP and unresponsive CBR VoIP. The last scenario is to study how the adaptive VoIP flows behave in a homogeneous adaptive VoIP network. Since all the adaptive VoIP flows implement the congestion control, the overall network congestion should be minimized. This would allow optimal performance to all the competing flows. With the same congestion control, it can also be expected that a fair bandwidth allocation should be provided among the competing flows. In this section, we set up a homogeneous adaptive VoIP network, based on the network topology in Figure 4-7, with three adaptive VoIP sources. The adaptive VoIP sources are based on the ADPCM codec with variable packetization, giving the transmission rates among 64, 53.3, 48, 44.8, and 42.6 Kbps. In order to ensure congestion in the simulation, the bottleneck link capacity is set to 120 Kbps. Specifically, with two competing flows, the aggregate bandwidth is at most 128 Kbps. With three competing flows, the aggregate bandwidth is at most 127.8.

Figure 4-13 is the simulation result showing transmission rate and voice frame loss of the three adaptive VoIP flows: AVoIP0, AVoIP1 and AVoIP2, respectively. Note that voice frame loss in the figures is normalized so that a packet loss with 10-ms packetization causes voice frame loss of 10. A packet loss with larger packetization causes correspondingly more voice frame loss. For example, a packet loss with 20-ms packetization causes voice frame loss of 20.

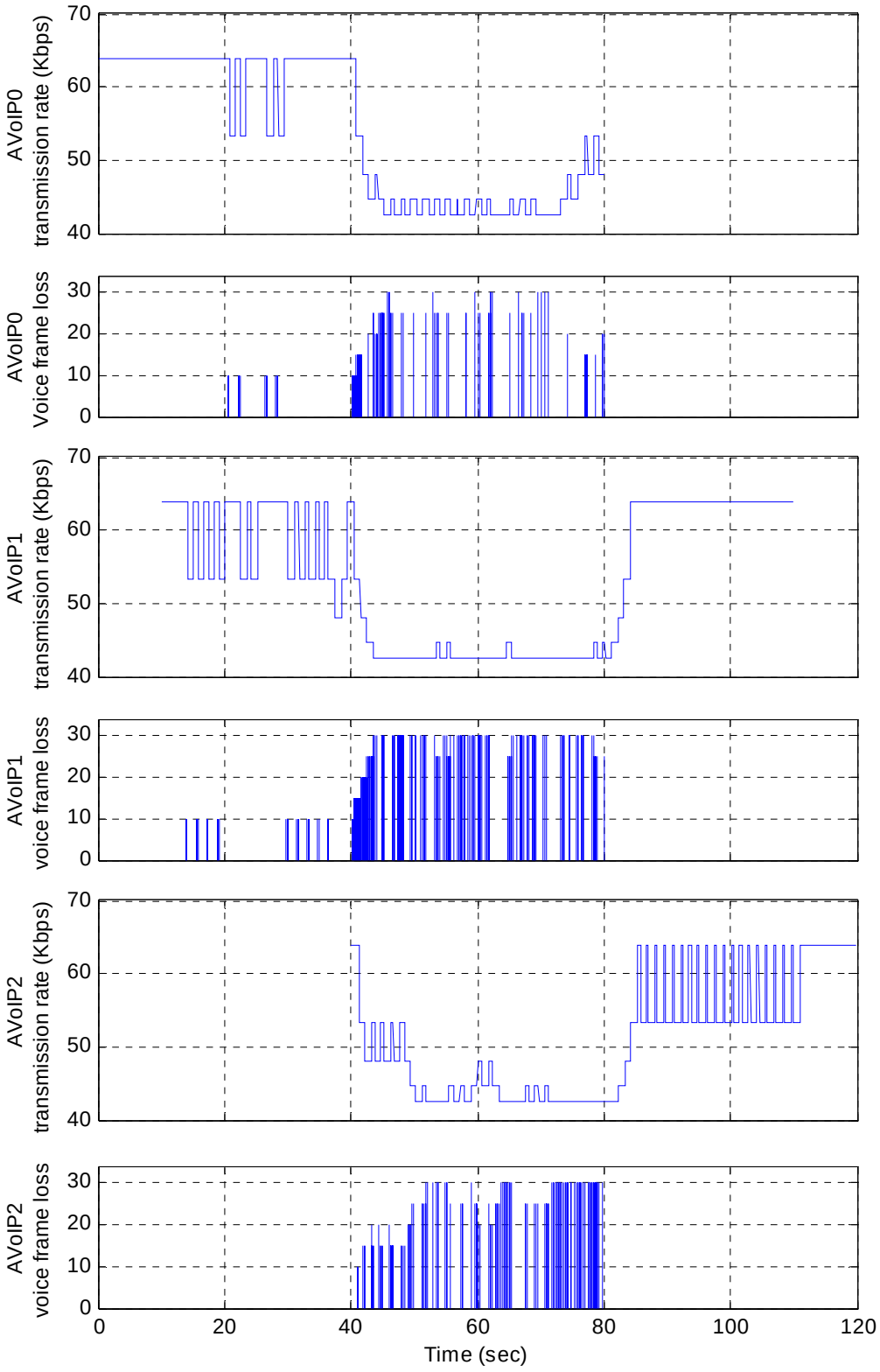

Figure 4-13 Homogeneous adaptive VoIP network

From the figure, AVoIP0 starts off at time zero second and gets the needed maximum bandwidth of 64 Kbps. AVoIP1 starts at time 10 seconds. So, the aggregate bandwidth from both flows is now 128 Kbps, which is more than the bottleneck bandwidth. Both flows observe packet losses and respond accordingly, which results in the transmission rate alternating between 64 and 53.3 Kbps. AVoIP2 starts at time 40 seconds. At the moment, the three flows transmit packets at the maximum rate. This gives the aggregate bandwidth of 192 Kbps, which is much more than the bottleneck bandwidth. As a consequence, all three flows experience excessive packet loss. The figure shows that the three flows can quickly respond to the decreasing bandwidth, by reducing the transmission rate to the minimum of 42.6 Kbps. Nevertheless, the three flows still suffer some packet losses because the new aggregate bandwidth of 127.8 Kbps is slightly more than the bottleneck bandwidth.

At time 80 seconds, AVoIP0 ends its session. The figure shows that both AVoIP1 and AVoIP2 can quickly respond to the bandwidth that becomes available. Both flows attempt to increase the transmission rate to the maximum. As seen from the figure, AVoIP1 is able to operate at 64 Kbps with no packet loss; while AVoIP2 has the transmission rate alternating between 64 and 53.3 Kbps, with no packet loss as well. What happens is a result of asymmetric observation. AVoIP2 appears to have a lower delay threshold than AVoIP1. Hence, AVoIP2 has its fast response strategy triggered sooner than AVoIP1. As AVoIP2 reduces its transmission rate in response to the delay threshold strategy, it proactively prevents congestion and packet loss from happening. As a result, AVoIP1 continues to observe no congestion. Although asymmetric observation results in an unfair bandwidth allocation between the two flows, both have the best possible performance. Because congestion never happens, both AVoIP1 and AVoIP2 experience no packet loss and very small delay. Although AVoIP1 gets more bandwidth than AVoIP2, the impact is very limited. Since rate adaptation is done by packetization, the only impact to AVoIP2 is an extra 5-ms packetization delay added to the end-to-end delay.

This simulation demonstrates the ability of the adaptive VoIP to constantly adjust its transmission rate to the optimal. In a homogeneous network of the adaptive VoIP, the network can have the flexibility to accept more VoIP flows without causing severe performance degradation. Namely, the adaptive VoIP flows can automatically choose to operate at a lower transmission rate when the network gets congested. The result also shows

that adaptive VoIP can provide fair overall bandwidth allocation, but with a variation. This is due to the fact that the rate adaptation is large-grain. It is impossible for the adaptive VoIP to operate at any transmission rate that gives a fair share of bandwidth. Asymmetrical observation is another effect because the competing flows may not infer the same network condition at the same time. Asymmetrical observation is not uncommon. In well developed TCP-friendly protocols, researchers need to deal with this problem as well, in order to achieve fair bandwidth allocation.

4.6 Summary

The main focus of this chapter is to design an adaptive-rate control that makes use of the components we have proposed in the previous two chapters; namely, rate adaptation and network state detection. Integrating the three components allows us to develop a novel and unique adaptive-rate VoIP called Sync & Sense enabled adaptive packetization VoIP. Since rate adaptation is based on packetization, our adaptive VoIP can work with any constant bitrate speech coder. The advantage is that rate adaptation is independent to the coder and is likely to be more transparent to the user. By incorporating the Sync & Sense methodology, our adaptive VoIP has the ability to make the control decision based, not only on the commonly used packet loss, but also one-way network delay. In the design of the adaptive-rate control, we identify three key issues: placement of the control, decision metrics, and increase/decrease algorithm.

It is clear that the placement of the control should be on the receiver. Because Sync & Sense performs on the receiver, this allows the control to immediately and continuously process the observations about the state of the network. The control sends a command feedback to the sender only when there is a need to adjust the transmission rate. In terms of the decision metrics, we use packet loss as it is an effective indication of network congestion. In addition, the control monitors the trend of one-way network delay. Studies have shown that there is a correlation between one-way network delay and packet loss in which packet loss is likely, which we denote as the delay threshold. Monitoring the delay trend allows the control to respond sooner in anticipation of the potential packet losses. Therefore, the increase/decrease algorithm is based on both the packet loss and delay threshold strategies,

running in parallel. Since rate adaptation relies on the speech coder being used, the algorithm is limited to a stepwise fashion. That is, it reduces the transmission rate by one step when congestion is observed, and increase the transmission rate by one step in the absence of congestion. In the packet loss strategy, the algorithm conservatively responds to every packet loss event. In the delay threshold strategy, the algorithm monitors the delay trend if it has crossed the delay threshold. The decision to adjust the transmission rate is based on the four possible scenarios. The delay trend allows the algorithm to test the condition of self-induced congestion. As a result, the algorithm can effectively respond to the congestion. Specifically, if the congestion is self-induced, the algorithm responds by reducing the rate. If the congestion is none self-induced, the algorithm increases the rate in order to optimize the performance. If the congestion is due to TCP traffic, the algorithm increases the rate to maintain its aggressiveness to compete for its needed share of bandwidth.

The simulation study is divided into two parts. In high statistical multiplexing, the aggregate competing traffic is relatively insensitive to the adaptive VoIP flow under study. The result shows that our adaptive VoIP is aware of network congestion and can react promptly and appropriately. Under any offered load, it can compromise between the delay and loss performance, and give an optimal performance. In low statistical multiplexing, the competing traffic is sensitive to the adaptive VoIP flow under study. Thus, besides the performance, interaction among the competing flows is of interest in the simulation. In a heterogeneous network with TCP traffic, the result shows that our adaptive VoIP has the aggressiveness to compete for its needed share of bandwidth. At the same time, it demonstrates a certain degree of TCP-friendliness in which it responds to congestion in the same manner as TCP does. In a heterogeneous network with CBR VoIP traffic, the result shows that our adaptive VoIP encounters the unfairness, just like TCP does. However, flexibility is the strength of the adaptive VoIP that could allow it to be in a better position to acquire the needed share of bandwidth. In a homogeneous adaptive VoIP network, the flows can automatically choose to operate at a lower transmission rate when the network gets congested. This allows the network to accept more VoIP flows without causing severe performance degradation. The result also demonstrates that our adaptive VoIP can provide fair overall bandwidth allocation, but with a variation. This is due to the effect of asymmetrical observation, as well as the fact that the rate adaptation is large-grain. In terms

of comparative performance, the simulation results also show that our adaptive VoIP performs better than traditional CBR VoIP in all aspects mentioned above. Inflexibility is a problem for CBR VoIP because the transmission often does not match the changing network condition.

Chapter 5

Conclusion

This dissertation addresses the inherent problem of VoIP: the mismatch between VoIP and the network. Namely, VoIP has a strict requirement of bandwidth, delay, and loss, but the network (particularly best-effort service networks) never guarantees such a requirement. To deal with this problem, we focus on the endpoint approach to enhance VoIP with an adaptive-rate control, called adaptive-rate VoIP. Adaptive-rate VoIP has the ability to detect the state of the network and adjust the transmission accordingly. Therefore, it is network-aware and intelligent enough to optimize its performance, making it resilient and robust to the service offered by the network. Adaptive-rate VoIP is generally composed of three components: rate adaptation, network state detection, and adaptive-rate control. In this dissertation, we take a comprehensive approach in studying and developing an adaptive-rate VoIP system. We carefully look at each component, identify the problems, and find the solution to overcome them. Below, we describe the contributions of this dissertation, followed by future research areas.

5.1 Contributions

The contribution of this work is threefold. First, we study optimizing packetization, which can be used as an alternative means for rate adaptation, instead of using variable bitrate speech coders. As an advantage, optimizing packetization allows any constant bitrate speech coder to perform rate adaptation. Since rate adaptation is independent of the coder, it causes no impact on the output voice frames from the coder. Second, we propose a novel measurement methodology called Sync & Sense of periodic stream. Sync & Sense is unique in that it can virtually synchronize the transmission and reception timing of the VoIP session,

without requiring a synchronized clock. Sync & Sense has the ability to measure one-way network delay and obtain the full spectrum of the delays of the VoIP session. In addition, Sync & Sense can estimate the available network bandwidth. Sync & Sense truly serves the objective of detecting the state of the network for VoIP, as well as providing a wider range of indications about the network condition. Third, we carefully consider the design choices of adaptive-rate control. The goal is to design an adaptive-rate control that makes use of the two proposed components of rate adaptation and network state detection, as well as possesses the desirable properties. Specifically, we ensure that the control is quick to respond to changing network conditions, aggressive enough to compete with TCP for its needed share of bandwidth, responsive to network congestion, friendly to the competing TCP traffic, and convergent to a fair bandwidth allocation among the homogeneous adaptive VoIP flows. The integration of the three proposed components is a novel and unique adaptive-rate VoIP called Sync & Sense Enabled Adaptive Packetization VoIP.

The contributions mentioned above are not restricted to our adaptive VoIP. The proposed work in each component is unique in itself and can be adopted by other systems. The knowledge and findings about optimizing packetization show the general benefits and limitations of adaptive-rate VoIP. They can be applied in any adaptive-rate VoIP system. Sync & Sense offers an indispensable method of detecting the state of the network. Sync & Sense can be beneficial to many VoIP applications. For example, a VoIP agent, including routers and gateways, can use the observations of the delay, loss, and bandwidth as provided by Sync & Sense to compare alternate routes and choose the optimal one for transmitting packets. Sync & Sense can be used for management purposes to monitor and collect performance statistics. This can also be used as the input for perceptual speech quality assessment algorithms. The design consideration of adaptive-rate control offers a practical guide that discusses the issues and the desirable properties. Since the proposed adaptive-rate control is independent of the speech coder, it can be applied in any VoIP system.

5.2 Future Research

The studies in this dissertation are conducted using a simulation methodology. The simulation simplifies many procedures and allows us to closely refine the design, validate the

assumption, and examine the interested issues. Throughout the simulations, end-to-end delay and voice frame loss are used to evaluate the performance. Extending the evaluation scheme is an interesting future research. This will allow us to evaluate voice quality, which is more useful than performance. This may be done by feeding the end-to-end delay and voice frame loss results into an objective speech quality assessment scheme. Further step would be extending the studies in a real-world experiment. We will have the opportunity to verify the simulation results, discover new findings, refine the existing works, and conduct subjective speech quality assessment. Another area of future research is about Sync & Sense. Sync & Sense relies on the common network assumption that a network is a series of queue servers in which packets get queued behind each other while in transit. This, however, may not be the case in some circumstances. Multi-channel links, for example, violate the assumption that there is a single end-to-end forwarding path. As the network becomes more heterogeneous, wireless networks are often part of the forwarding path. Since wireless networks employ additional mechanisms to deal with the link reliability, handover, and etc, it has a potential to affect the assumption as well. Future research is needed to study each individual case and explore the options to improve the capability of Sync & Sense.

The proposed adaptive VoIP is designed with the objective that it should possess the desirable properties. We achieve the objective to a certain degree. However, several areas can be an interesting future research. Further refinement would be needed if we wish to ensure the adaptive VoIP to be TCP-friendly to a higher degree. In a homogeneous adaptive VoIP network, more studies would be needed to ensure that a fair bandwidth allocation is provided, especially in a larger network scale. Extending the study of the adaptive VoIP into a wireless area would be a challenging future research. Whereas the proposed adaptive VoIP is an option that makes use of the Sync & Sense methodology, Sync & Sense offers tremendous benefits that can enable different kinds of adaptive VoIP as well. Future research may explore other alternatives that can benefit from Sync & Sense. For example, Sync & Sense enabled adaptive route VoIP would allow the sender to assess the observations of the delay, loss, and bandwidth among different routes and switch to transmit packets on the optimal route.

Bibliography

[1] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," Internet Engineering Task Force, RFC 2205, September 1997.

[2] S. Blake, D. Black, M. Carlson, E. Davis, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services," Internet Engineering Task Force, RFC 2475, December 1998.

[3] S. Shenker. "Fundamental Design Issues for the Future Internet," *IEEE Journal of Selected Areas in Communications*, Volume 13, Number 7, September 1995.

[4] J. H. Saltzer, D. P. Reed, and D. D. Clark. "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems*, Volume 2, Number 4, November 1984.

[5] Z. Qiao, L. Sun, N. Heilemann, and E. Ifeachor. "A new method for VoIP Quality of Service control use combined adaptive sender rate and priority marking," *ICC 2004 – IEEE International Conference on Communications*, Volume 27, Number 1, June 2004.

[6] F. Beritelli, G. Ruggeri, and G. Schembra. "TCP-Friendly Transmission of Voice over IP," Proceedings of IEEE ICC, New York, USA, April 2002.

[7] ETSI EN 301 704 V7.2.1 (2000-04), Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding (GSM 06.90 version 7.2.1 Release 1998).

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 1889, January 1996.

[9] R. Rejaie, M. Handley, and D. Estrin. "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," Proceedings of IEEE INFOCOM, New York, USA, March 1999.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. “Equation-Based Congestion Control for Unicast Applications,” Proceedings of ACM SIGCOMM, Stockholm, Sweden, August 2000.

[11] J.-C. Bolot, and A. Vega-Garcia. “Control mechanisms for packet audio in the internet,” Proceedings of IEEE INFOCOM, San Francisco, USA, March 1996.

[12] The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns/.

[13] M. S. Taqqu, W.Willinger, and R. Sherman. “Proof of a Fundamental Result in Self-Similar Traffic Modeling,” *ACM Computer Communications Review*, pp. 5 – 23, April 1997.

[14] K. Claffy, G. J. Miller, and K. Thompson. “The nature of the beast: recent traffic measurement from an Internet backbone,” Proceedings of INET '98, Geneva, Switzerland, July 1998.

[15] K. Thompson, G. J. Miller, and R. Wilder. “Wide-Area Internet Traffic Patterns and Characteristics,” *IEEE Network*, pp. 10 – 23, November 1997.

[16] B. Ngamwongwattana. “Optimizing Packetization for Minimal End-to-End Delay in VoIP Networks,” Master Thesis, University of Pittsburgh, 2001.

[17] ITU-T Recommendation G.114. “One-way Transmission Time,” May 2000.

[18] R. Dai. “A White Paper on SMOS,” http:// www.sageinst.com, November 2001.

[19] V. Abreu-Sernandez and C. Garcia-Mateo. “Adaptive multi-rate speech coder for VoIP transmission,” *Electronics Letters*, Volume 36, Number 23, November 2000.

[20] A. Barberis, C. Casetti, J.C. De Martin, and M. Meo. “A Simulation Study of Adaptive Voice Communications on IP Networks,” *Computer Communications*, 2001.

[21] S. Mohamed, F. Cervantes-pérez, and H. Afifi. "Integrating Networks Measurements and Speech Quality Subjective Scores for Control Purposes," Proceedings of IEEE INFOCOM, April 2001.

[22] K. Claffy, G. Miller, and H-W. Braum. “Measurement Considerations for Assessing Unidirectional Latencies,” *Internetworking: Research and Experience*, 4 (3), pp. 121-132, September 1993.

[23] D. D. Clark and D. L. Tennenhouse. “Architectural Considerations for a New Generation of Protocols,” Proceedings of the SIGCOMM Symposium on

Communications Architectures and Protocols, *Computer Communications Review*, Volume 20, Number 4, September 1990.

[24] D. L. Mills. "Network Time Protocol," Internet Engineering Task Force, RFC 1305, March 1992.

[25] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE/ACM Transaction on Networking*, Volume 17, Issue 6, November 2003.

[26] J. C. Bolot. "Characterizing End-to-End Packet Delay and Loss in the Internet," Proceedings of ACM SIGCOMM, San Francisco, USA, September 1993.

[27] F. Tobagi, A. Markopoulou, and M. Karam. "Is the Internet ready for VoIP?," Proceedings of IWDC, Capri, Italy, September 2002.

[28] A. Markopoulou, F. Tobagi, and M. Karam. "Assessment of VoIP quality over Internet backbones," Proceedings of IEEE INFOCOM, New York, NY, USA, June 2002.

[29] R. Zopf. "Real-Time Transport Protocol (RTP) Payload for Comfort Noise (CN)," Internet Engineering Task Force, RFC 3389, September 2002.

[30] J. Postel. "Transmission Control Protocol," Internet Engineering Task Force, RFC 793, September 1981.

[31] V. Jacobson. "Congestion Avoidance and Control," Proceedings of ACM SIGCOMM, *Computer Communication Review*, August 1988.

[32] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne. "Adaptive playout mechanisms for packetized audio applications in wide-area network," Proceedings of IEEE INFOCOM, Toronto, Canada, June 1994.

[33] S. B. Moon, J. Kurose, D. Towsley, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms," *ACM Multimedia Systems*, 1998.

[34] A. Kansal and A. Karandikar. "Adaptive Delay Estimation for Low Jitter Audio over Internet," Proceedings of IEEE GLOBECOM, San Antonio, TX, USA, November 2001.

[35] J. W. Seo, S. J. Woo, and K. S. Bae. "A Study on the Application of an AMR Speech Coder to VoIP," Proceedings of IEEE International Conference on Acoustics, Speech, and Signal, Volume 3, May 2001.

[36] L. S. Brakmo, S. W. O'Malley, and L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance," Proceedings of ACM SIGCOMM, August 1994.

[37] S. B. Moon. "Measurement and Analysis of End-to-End Delay and Loss in the Internet," Ph.D. Thesis, University of Massachusetts, Amherst, 2000.

[38] V. Paxon. "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. Thesis, University of California, Berkeley, 1997.

[39] L. Roychoudhuri, E. Al-Shaer, H. Hamed, and G. B. Brewster. "On Studying the Impact of the Internet Delays on Audio Transmission," IEEE Workshop on IP Operations and Management, 2002.

[40] Y. R. Yang, M. Kim, and S. Lam. "Transient Behaviors of TCP-friendly Congestion Control Protocols," Proceedings of IEEE INFOCOM, April 2001.

[41] S. Jin, L. Guo, I. Matta, and A. Bestavros. "TCP-Friendly SIMD Congestion Control and its Convergence Behavior," Proceedings of IEEE ICNP, November 2001.

[42] S. Floyd and K. Fall. "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999.

[43] Y. R. Yang and S. Lam. "General AIMD congestion control," Proceedings of IEEE ICNP, November 2000.

[44] D. Bansal and H. Balakrishnan. "Binomial congestion control algorithms," Proceedings of IEEE INFOCOM, April 2001.

[45] D. Sisalem and A. Wolisz. "LDA+: A TCP-Friendly Adaptation Scheme for Multimedia Communication," Proceedings of IEEE International Conference on Multimedia, August 2000.

[46] Y. Joo, V. Ribeiro, A. Feldmann, A. C. Gilbert, and W. Willinger. "TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations," Proceedings of ACM SIGCOMM, *Computer Communication Review*, 2001.

[47] N. Brownlee and K. Claffy. "Understanding Internet traffic streams: Dragonflies and tortoises," *IEEE Communications Magazine*, Volume 40, Number 10, October 2002.

[48] Q. Li and D. L. Mills. "Jitter-based delay-boundary prediction of wide-area networks," IEEE/ACM Transactions on Networking, Volume 9, Number 5, October 2001.

[49] V. Jacobson. “Pathchar: A Tool to Infer Characteristics of Internet Paths,” ftp://ftp.ee.lbl.gov/pathchar/, April 1997.

[50] A. B. Downey. “Using Pathchar to Estimate Internet Link Characteristics,” Proceedings of ACM SIGCOMM, September 1999.

[51] B. A. Mah. “Pchar: A Tool for Measuring Internet Path Characteristics,” http://www.kitchenlab.org/www/bmah/Software/pchar/, February 2005.

[52] R. L. Carter and M. E. Crovella. “Measuring Bottleneck Link Speed in Packet-Switched Networks,” *Performance Evaluation*, Volume 27, 1996.

[53] K. Lai and M. Baker. “Measuring Bandwidth,” Proceedings of IEEE INFOCOM, April 1999.

[54] K. Lai and M. Baker. “Nettimer: A Tool for Measuring Bottleneck Link Bandwidth,” Proceedings of the USENIX Symposium on Internet Technologies and Systems, March 2001.

[55] C. Dovrolis. “Pathrate: A measurement tool for the capacity of network paths,” http://www.pathrate.org, January 2004.

[56] M. Jain and C. Dovrolis. “End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput,” *IEEE/ACM Transaction on Networking*, Volume 11, Issue 4, August 2003.

[57] B. Melander, M. Bjorkman, and P. Gunningberg. “A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks,” Proceedings of IEEE GLOBECOM, 2000.

[58] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. “pathChirp: Efficient Available Bandwidth Estimation for Network Paths,” Proceedings of Passive and Active Measurement Workshop, April 2003.

[59] S. Shalunov, B. Teitelbaum, and A. Karp. “A One-way Active Measurement Protocol (OWAMP),” Internet Engineering Task Force, RFC 4656, September 2006.

[60] W. Matthews and L. Cottrell. “The PingER Project: Active Internet Performance Monitoring for the HENP Community,” *IEEE Communications Magazine*, May 2000.