

# A Real-Time Network Simulation Application for Multimedia over IP

Christopher M. White, Josh Raymond, and Keith A. Teague

*Speech and Audio Communications Laboratory*  
School of Electrical and Computer Engineering  
Oklahoma State University  
Stillwater, OK 74078

## ABSTRACT

This paper details a Secure Voice over IP (SVoIP) development tool, the Network Simulation Application (*Netsim*), which provides real-time network performance implementation of Quality of Service (QoS) statistics in live real-time data transmission over packet networks. This application is used to implement QoS statistics including packet loss and inter-arrival delay jitter. *Netsim* is written in Visual C++ using MFC for MS Windows environments. The program acts as a transparent gateway for a SVoIP server/client pair connected via IP networks. The user specifies QoS parameters such as mean delay, standard deviation of delay, unconditional loss probability, conditional loss probability, etc. *Netsim* initiates and accepts connection, controls packet flow, records all packet sending / arrival times, generates a data log, and performs statistical calculations.

## I. INTRODUCTION

Packet-switched networks are often used to provide interactive multimedia communications, including real-time voice, video and data services. Packet networks were not designed for real-time multimedia applications such as voice or video communications as they do not provide a dedicated end-to-end connection such as is provided by circuit switched networks [1]. Voice, video, and multimedia are affected by packet network characteristics in the same manner; this paper will emphasize Voice over Internet Protocol (VoIP) as a representation of all three. Networks exhibit non-ideal behavior that may seriously degrade performance of real-time communications systems. Particularly, as packets are transmitted from source to destination they may experience varying amounts of delay (jitter) and loss which are functions of a variety of factors. Jitter and loss may result from packets traversing different routes between source and destination due to congestion or failed links, variable queuing delays encountered at network routers, or competing network traffic [2]. A receiver must accommodate this non-ideal behavior or risk buffer underrun or overrun affecting intelligibility and naturalness as shown in Figure 1.

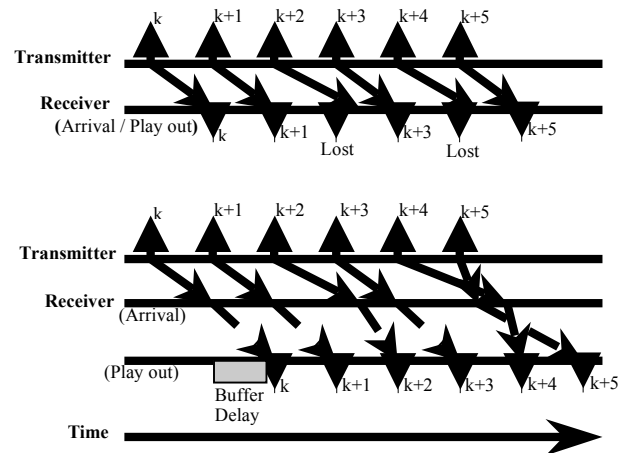


Figure 1: Packet Delay Timing Diagram

Packet loss and jitter constitute the major factors affecting the QoS in a system that uses packet networks (IP) to transmit real-time voice data. Excessive packet delay can occur when the queue in a router becomes full and subsequent packets entering the queue are delayed. Large end-to-end delay (latency) causes speech overlap as well as mutual silence at the receiver system. Large packet inter-arrival delays (jitter) cause packets to arrive too late for their designated playback time, which forces the receiver to discard the late packets ([3] [4]). If there are large queuing delays, consecutive packets may cluster at the nodes and be delivered almost simultaneously to the receiver. As a result, the packets received may exceed the capacity of the playout buffer causing an overrun. In addition to late packet loss at the application, loss can occur as a result of congestion at routers, where packets are discarded because the router can not service all the packets in its queue [5]. Congestion at routers can result in consecutive packet loss, known as burst loss. Burst loss can have a large impact on system performance, potentially leaving large gaps in voice output. Burst loss is difficult to correct due to the effects of sequential errors on playout. Congestion may also cause subsequent packets to take an alternate path to their destination. Packets may encounter different delays as they take different paths as some nodes may have varying queuing delays (wait times) along the path, possibly arriving at their destination out-of-order. If uncorrected, this could cause garbled speech output at the receiving system.

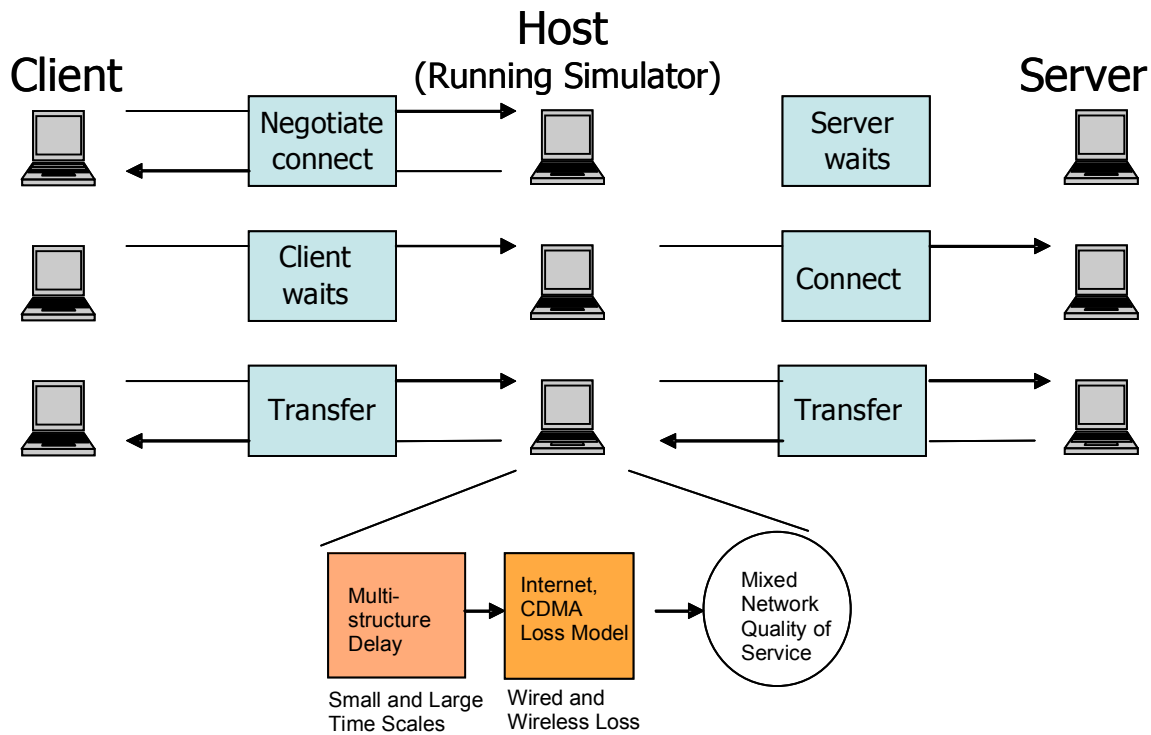


Figure 2: Overview

The voice output of the communication system can be seriously degraded resulting in low overall QoS if there are no mechanisms in place to manage and correct for the aforementioned QoS issues. Understanding the VoIP system response to these environments offers insight into preventative measures that will increase the system QoS in the presence of network anomalies. Furthermore, as modern communication moves toward packet network transmission, there is potential for aggregate concatenations of dissimilar networks where non-ideal behavior becomes more complex, particularly with error prone packet wireless networks. *Netsim* provides a mechanism for better understanding VoIP system response to varied QoS through collection of empirical data over a real network connection in a process that closely simulates an actual network configuration.

#### Purpose of study

This study details the design and development of a SVoIP tool, the Network Simulation Application (*Netsim*), to implement and imitate the non-ideal behavior of packet networks in real-time communications. This application interacts with a SVoIP system by intercepting the transfer of real-time packet data and imposing the desired network characteristics using an underlying network model. This application is useful to determine the SVoIP system response to a non-ideal network and the quality of a voice packet stream transmission. An overview of the application is shown in Figure 2.

In order to design better SVoIP systems, it is necessary to understand the response to similar and mixed network attributes. Error management schemes and uninterrupted transmission flow require foreknowledge of system response to non-ideal network behavior [6]. This study depicts an application which implements network models found in the literature in a real-time communications application. It serves to create a testing environment suitable for VoIP application enhancement. The development of this tool facilitates an information base with which to make educated decisions during the design of a VoIP communication application, or during its refinement.

## II. PACKET NETWORK MODELS

Many general packet network models are proposed in the literature, but it is important that the model capture the characteristics of the network being modeled. As examples, two models found in the literature which model packet loss for the Internet and cellular data are combined with a multi-structure IP packet network delay jitter model for this study. The widely used Gilbert model [7] (burst loss model) is used for the Internet link, the correlated fading channel model [8] is used for cellular data (IS-95 and IS-2000 cdma2000) radio links, and our multi-structured network delay jitter model [9] is used for inter-arrival delay. The Gilbert model captures the burst loss error characteristics of the Internet using both unconditional and conditional loss probabilities. The correlated fading channel model uses a first-order Markov chain based on Rayleigh distributed fading to model burst loss due to multi-path fading in CDMA data channels. The multi-structure network delay jitter model captures the stationary and non-stationary multi-structure characteristics [10] of packet network delay. These models are incorporated into *Netsim* to monitor the response of a SVoIP system in artificial mixed network environments. Network model parameters are specified to customize the response.

## III. APPLICATION DESIGN

### A. Architecture

*NetSim* is written in Microsoft Visual C++ v6.0 as a dialog box-based application. C++ is ideal for rapid application development through the dialog box editor and the MFC application framework. Also, C++ eases complex multithreaded programming, and offers support for network application development. Using Microsoft Visual C++ enables the application to run in Microsoft Windows environments.

*NetSim* contains three main threads: *netControlReceiver* thread, *netControlTransmitter* thread, and *listenControl* thread. Each of these threads runs as multiple instances for each network communication channel created. *listenControl* listens for an incoming connection and starts *netControlReceiver* and *netControlTransmitter* when the connection is created. *netControlReceiver* waits for data to be received on the channel for which it was created and sends that data to *netControlTransmitter* to send on the network to the client. *netControlTransmitter* waits for data from *netControlReceiver*, and when it receives data to be sent, it delays for a certain amount of time depending on the jitter model parameters, drops the packet depending on the loss model parameters, and if the packet is not dropped it is sent to the client. *netControlTransmitter* also adds each jitter or loss data point to a queue for graphical display at the end of the program.

### B. Flow of Control

When *NetSim* starts, a dialog box appears that allows the user to configure the connection settings, network simulation model parameters, and start the connection, as shown in Figure 3. This dialog box also displays information about whether the connection has completed and if there were any errors. The Client Information box allows the user to enter the IP address, control port (TCP), and communication port (UDP) of the client computer. The control port is used by the two connecting computers to send control information, such as connection parameters, back and forth. The UDP port is the port on which voice is sent and received between the two computers. Similarly, the Server Information box allows the user to enter information about the server IP address, control port, and communication port.

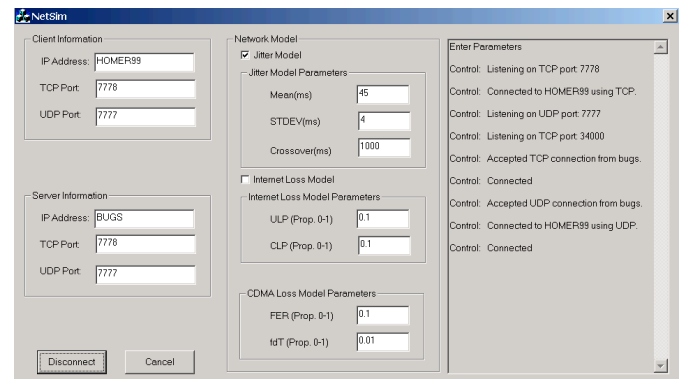


Figure 3: Main GUI

The Network Model box contains configuration parameters used in simulating the network. The Jitter Model Parameters box allows the user to enter information about the jitter model being used, such as the mean jitter (ms), standard deviation, and crossover. The Internet Loss Model box allows for internet loss parameters, such as conditional loss probability (*clp*) and unconditional loss probability (*ulp*), to be configured. The CDMA Loss Model Parameters box allows for CDMA loss to be configured through normalized Doppler bandwidth fading (*fdT*), and average frame error rate (*FER*). There are check boxes named Jitter Model and Internet Loss Model that allow the jitter modeling and loss modeling to be enabled. With both models disabled, the connection is transparent.

When the necessary parameters are configured, clicking the “Connect” (currently labeled “Disconnect”) causes *NetSim* to attempt to connect to the server, and wait for a connection from the client. As soon as the button is clicked the text changes from “Connect” to “Disconnect”. On the right side of the box is a read only edit box that displays connection information, including the progress as *NetSim* attempts to connect to the server, and the status of the client connection.

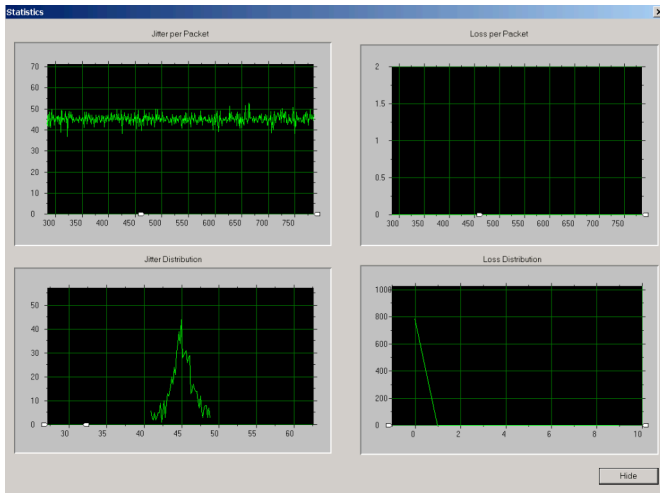


Figure 4: Graphical Display

When the user is finished, clicking “Disconnect” will disconnect each connection and display a dialog box containing several graphs of data collected during the simulation, as shown in Figure 4. The *Jitter per Packet* graph shows jitter between each packet. The x-axis is the packet number sent and the y-axis is the jitter as measured by internal timers (not simply the jitter model values). The *Jitter Distribution* graph shows a distribution of the jitter data. The x-axis is the number of milliseconds of measured jitter as measured and the y-axis is number of packets. The *Loss per Packet* graph is similar to the *Jitter per Packet* graph, showing how many packets were dropped each time a packet was received. More than one packet at a time can be dropped for a bursty effect due to the models. Finally, the *Loss Distribution* graph shows how many times packets were dropped, and how many packets were dropped in each burst. On the x-axis is the number of packets dropped, and the y-axis is the number of times the corresponding number of packets was dropped. In the example shown, loss modeling was turned off, so no packets were dropped.

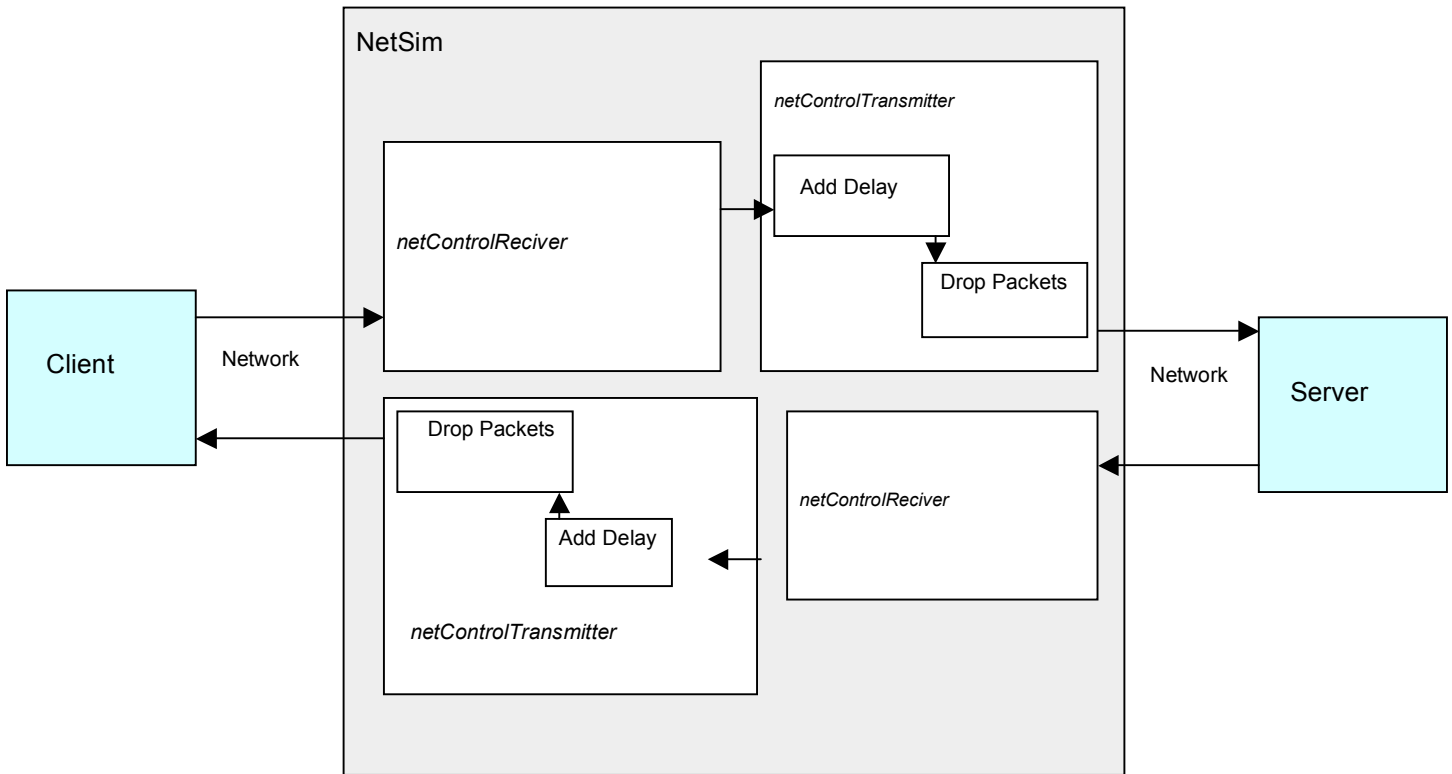


Figure 5: Control Operation

### C. The Application

When the user has entered all necessary configuration information in the initial dialog box and clicks "Connect", *NetSim* attempts to connect to the server on the control port and begins listening for the client by starting *listenControl* for the control port. Each connection is given a hard-coded ID that can be used within the threads for computer connection identification. Calculating jitter and loss using the models can be computationally expensive; therefore an array of model values is created and filled with calculated values before the connection is established. When a connection is received on the client communication channel, a connection to the server communication channel is attempted.

As each connection is established, an instance of *netControlReceiver* and *netControlTransmitter* is created for each connection to monitor and send data on each channel. *netControlReceiver* repeatedly calls *recv()* for the connection, waiting for data to be received on the network connection. When data is received, a critical section is entered and the data is added to a queue, then the critical section is exited and it begins waiting for data again. *netControlTransmitter* loops waiting for data to enter the queue. Once data is received in the queue, the thread checks to see what the next jitter value should be and loops for that many milliseconds, introducing delay into the connection. The delay is recorded and added to a queue to be graphed when the user clicks "Disconnect" on the main dialog box. The thread then checks the next loss value; if a packet should be dropped a count of the loss is incremented, and the number of packets dropped is added to a queue to be graphed later. Finally, if the packet is not dropped, the current packet is sent on the network. When the threads are created the parent class for the appropriate channel is passed to each thread, so that *netControlReceiver* is receiving on the correct channel and *netControlTransmitter* is sending on the correct channel (to the correct computer).

### IV. CONCLUSION

*Netsim* provides a convenient mechanism for performing real-time packet network simulation. Network performance parameters are easily set in order to customize response, and new models can be incorporated into the code if desired. *Netsim* has been used very successfully as an aid in the development of a SVoIP system for operation on mixed wired and wireless networks.

### REFERENCES

- [1] E. J. Daniel, K. A. Teague, "Performance of FNBDT and Low Rate Voice (MELP) over Packet Networks", *Conference Record of the 35<sup>th</sup> Asilomar Conference on Signals, Systems and Computers*, pp 1568 – 1572, Nov. 4-7, 2001.
- [2] M. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet Packet Loss: Measurement and Implication for End-to-End QoS," *Proc. IEEE ICPP Workshop '98*, pp. 3-12, 1998.
- [3] L. Zheng, L. Zhang, D. Xu, "Characteristics of Network Delay and Delay Jitter and its Effect on Voice over IP (VoIP)" *IEEE International Conference on Communications ICC 2001*, Vol. 1, pp. 122-126, 2001.
- [4] C. Fulton, S. Li, "Delay Jitter First-Order and Second-Order Statistical Functions of General Traffic on High-Speed Multimedia Networks" *IEEE/ACM Transactions on Networking*, Vol. 6, No. 2, April 1998.
- [5] T. Yletyinen, R. Kantola, "Voice Packet Interarrival Jitter Over IP Switching," *SBT/IEEE International Telecom Symposium ITS '98*, Vol. 1, pp. 16-21, 1998.
- [6] C.M. White, E.J. Daniel, K.A. Teague, "A Network Performance Application for Modeling, Simulation, and Characterization of Packet Network Behavior," *37th Asilomar Conference on Signals, Systems, and Computers*, Nov. 2003.
- [7] E. N. Gilbert, "Capacity of a Burst-Noise Channel", *Bell System Tech. Journal*, Vol. 39, pp. 1253-1266, September 1960.
- [8] M. Zorzi, R. R. Rao, L. B. Milstein, "Error Statistics in Data Transmission Over Fading Channels", *IEEE Transactions on Communications*, Vol. 46, Issue 11, pp 1468-1477, November 1998
- [9] E.J. Daniel, C.M. White, K.A. Teague, "An Inter-Arrival Delay Jitter Model using Multi-Structure Network Delay Characteristics for Packet Networks," *37th Asilomar Conference on Signals, Systems, and Computers*, Nov. 2003.
- [10] Q. Li, D. L. Mills, "Jitter-Based Delay-Boundary Prediction of Wide-Area Networks" *IEEE/ACM Transactions on Networking*, Vol. 9, No. 5, October 2001.