Active Jitter Control

Stephen F. Bush and Amit Kulkarni and Scott Evans and Lou Galup

Abstract—
This study is the first known implementation of jitter control in an active network. Jitter control is performed by active packets in a distributed packet by packet basis within an active network rather than on a per flow basis as in today's passive networks. This provides many new benefits and challenges. The concept and results of an experimental validation of this method are presented.

Keywords—**Active Networks, Jitter Control, Multimedia Quality of Service**

I. Introduction

This paper presents the concept and experimental validation of an active network based jitter control mechanism. The results show an improvement in the Quality of Service (QoS) for real-time multimedia distribution over the next generation Internet.

Previous work on jitter control has assumed a non-active network and has usually involved queuing and scheduling algorithms applied uniformly to a packet flow, for example [4]. In a non-active network, data control algorithms are node-centric and packets of data are viewed as non-active entities to be forwarded from an entry point in the network to a destination, or multiple destinations for multicast streams. Customized processing of these non-active packet flows requires special low level setup within the network and also requires the implementation of a jitter control algorithm at every node of the network. To fulfill the latter requirement, one has to standardize the packet header format and the algorithm itself to ensure compliance between vendors. Experience has shown that standardization is usually a long process. This inhibits experimentation and the rapid introduction of new services in the network, e.g. the new jitter control technique proposed in this paper. There is a general consensus that network architecture must be made programmable to enable application-specific customization of network resources and the rapid introduction of new services in the network.

Two approaches have emerged on how to make networks programmable. One effort is spearheaded by the OpenSig community, which argues that networks can be architected using a set of open programmable interfaces, opening access to network devices like the switches, routers and hubs, thereby enabling third-party developers to devise custom solutions for their applications. The other approach is that of active networks in which packets can contain custom code that executes in the execution environment provided by the nodes of the network. Thus an active network is no longer node-centric but packet-centric; that is, the code in the active packets is executed at the intermediate network devices as the packet travels through the network. The result is that customized processing is accomplished as the packet is transmitted through the network. This enables experimentation and the rapid deployment of new services. We choose the active network approach for a number of reasons. Firstly, the OpenSig effort still requires a higher degree of standardization because vendors have to agree on the API for the programmable interfaces and then they have to implement the API in their devices, as opposed to active networks in which only the execution platform needs to be standardized. Secondly, active networking provides a finer granularity of control and customization than the OpenSig standard, which has a fixed set of API calls that may or may not entirely meet application requirements. Thirdly, the interface API is fixed and rigidly implemented at the time of manufacturing or during upgrades. This makes it difficult to make improvements to the protocol until a new API is approved and implemented.

Stephen F. Bush, General Electric Corporate Research and Development, KWC-512, One Research Circle, Niskayuna, NY 12309 e-mail: bushsf@crd.ge.com (http://www.crd.ge.com/people/bush)

This work extends the state of the art by adding a new Quality of Service technique based upon active networking. The advantages of the active network approach validated within this paper are several. First, jitter control is performed on a per-packet basis, not per-flow as in current networks. This means that individual packets can vary their jitter control algorithm based on their overall value to the final product. For example, a packet providing visual background that is not moving may have a much higher tolerance for jitter than a packet in a more dynamic portion of the visual or audio transmission. Another advantage is that active networks allow a higher resolution of control in a more dynamic manner because changing a jitter control algorithm does not require years of standardization and vendor compliance. Finally, jitter control within the network better utilizes system resources. The amount of buffer space required in the network and at the end systems is smaller than if a single large jitter control buffer were used only at the end system. The result is a cheaper end system that is especially critical for small wireless devices such as video phones.

The metric used for jitter in these experiments comes for RFC1889, *RTP: A Transport Protocol for Real-Time Applications"* [1]. The jitter metric is defined as the difference in send and receive times between two packets, i and j. Let S_i be the send time of packet i and R_i be the receive time of packet i by the next hop. Let the difference between send and arrival times be defined as $D(i,j) = (R_j - S_j) - (R_i - S_i) = (R_j - R_i) - (S_j - S_i)$. The difference, $D(i,j)$, provides the jitter between any two particular packets, however, a jitter value which measures the accumulated jitter over all packets is required. Let the accumulated jitter be defined as $J = J + (|D(i-1,i)| - J)/16$. This jitter metric provides a smoothed measurement of accumulated jitter based on the current jitter and a weighted value of past jitter .

II. Jitter Control Technique

The active jitter control technique differs from previous jitter control techniques because individual packets have the ability to determine their own delay, on a hop by hop basis, in order to

minimize jitter. Jitter is related to the variance in packet inter-arrival time at a node and it is denoted by $\sigma_f(n)$, where n is the node, and f is a particular packet flow through the node. The total jitter is the sum of the jitter at each of the intermediate nodes in the path from the source, node 1 to the destination, node r. Thus the total jitter is given by $\sum_1^r \sigma_f(n)$. The ultimate goal is to reduce jitter at the receiver.

The active jitter control technique requires that active video packets contain code that allows each packet to be delayed by the minimum amount of time necessary in order to minimize the jitter. Jitter is caused by uneven load within the network. If the load can be predicted reasonably well, then both the route and the packet delay that minimizes jitter can be calculated based on the maximum load anticipated at each node during the anticipated lifetime of the video stream connection at the link by the active network based predictive method described in [2].

Without a predictive load mechanism, packets determine the necessary delay that minimizes jitter at node n by interacting with a Java code object that updates a distribution of the variation in packet delay at the current node and flow. The packet then delays itself by a value (given by the code object) that is equal to a given quantile of the distribution. The code object is stored in named cache called *SmallState* that is available at all nodes in an active network. SmallState enables the active packets to leave information at a node that can be retrieved later by other packets belonging to the same flow. An active network enables active packets to control their SmallState cache. Packets explicitly create, access, modify and destroy SmallState. Packets also define access policies for SmallStates that they create.

The additional increase in jitter caused by the additional sources of jitter in the active execution environment is minimized by careful placement of control and measurement within the active packet. For example, accessing small-state is included as part of the transfer delay whenever possible so that access variation is automatically minimized along with other delay variations. Also, the amount of jitter control computation within the packet is minimized. As mentioned earlier, a Java code object stored in the SmallState maintains a distribution of the transfer times. A packet delays itself by a quantile of the transfer delay distribution stored in the Java distribution object. Although the particular quantile is fixed in the results shown in this paper, the interesting point of being active is that the quantile is determined not only on a node by node basis but a packet by packet basis. The calculation of the quantile can be arbitrarily complex as long as it is included as part of the transfer delay.

III. Active Jitter Control Challenges

The challenges involved in designing an active jitter control method highlight the challenges and benefits that active networks pose in general. The challenge is a fundamental shift in the paradigm from external control of passive network flow behavior to the control of external behavior from within the active packet. Enabling distributed jitter control from within each packet has tremendous benefits as explained in the previous section.

Firstly, the very nature of being active means that the code in the active packet executes on each node that it traverses on the way to its destination. The active packet carries general purpose code and can have a high degree of variance in terms of processing time. The specific active network implementation used to test the algorithm in this paper is the Magician execution environment described in [3]. In the Magician active network implementation, only one packet can access SmallState at a time. Therefore a jitter control algorithm that requires SmallState usage can cause packets to delay while waiting for access to any SmallState information. This adds additional variation to the processing time. Also, SmallState must be used carefully, since it is limited in space and time; that is, the state space should be minimized. Certain implementations of SmallState also have timeouts, the expiry of which automatically frees the space. Active packets that delay themselves in order to attempt to minimize jitter may wake up to a SmallState with modified values since other packets may have passed through while the jitter controlled packet was sleeping. Finally, control and measurement from within the active packet requires great care because of the following subtle point: the act of control and measurement from within the active packet adds delay and variance to the time spent by the active packet at a node. This skew caused by control and measurement is less apparent in passive networks, since the control is applied externally upon the packets. The efforts to meet these challenges were described previously.

IV. Active Jitter Control Implementation and Simulation

The active packets contain code that causes the packets to delay themselves at an intermediate node n by an amount equal to the a given quantile of the transfer delay distribution. The transfer time of a packet is t_s. The packets deposit their arrival times t_a in SmallState upon arrival at intermediate node n. For each packet, if $t_s < q_d$, the packet calculates the average inter-arrival time variance and sets its required delay time at node n to $t_a - t_s - q_d$ where q_d is the delay at a previously calculated quantile. The packet sleeps until the delay time expires, then forwards itself to its next destination node. If $t_s \geq q_d$, the packet is not delayed and is forwarded immediately to the next destination.

The configuration for experimental validation is shown in Figure 1 and Figure 2. In Figure 2, the active data packets are generated from active host node AH-1 and transmitted to active node AN-1 for transmission through the active network to AN-4. In Figure 1, the active network is used with no jitter control. The packets are passed along as soon as they are received.

V. Active Jitter Test Results

Figure 2 shows the configuration for the experimental operation. Each active packet carries the executable jitter control algorithm as well as the raw MPEG data as shown in the lower left of the figure. Figures 3 through 4 show the first hop jitter measured at the first node in the path through the active network shown in Figure 2. Figure 3 shows the jitter without any jitter control applied. The expected transfer time was 500 milliseconds with a transfer delay variance of 81254.9. Figure 4 is the jitter with the active jitter control algorithm set to delay at the

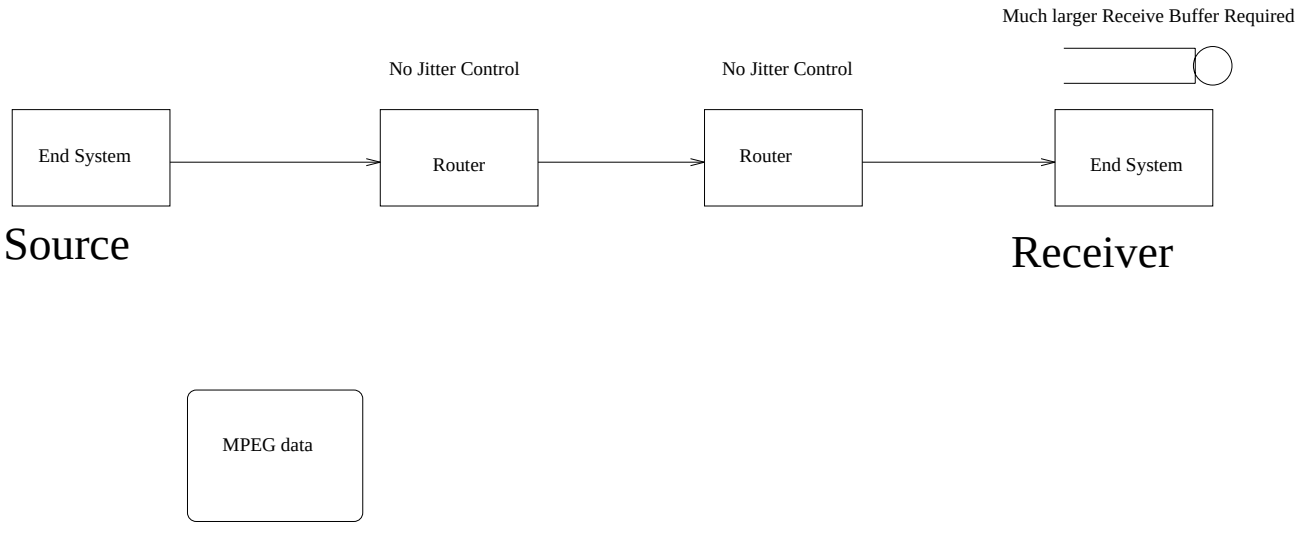


Fig. 1. Current Technology Non-Active Configuration.

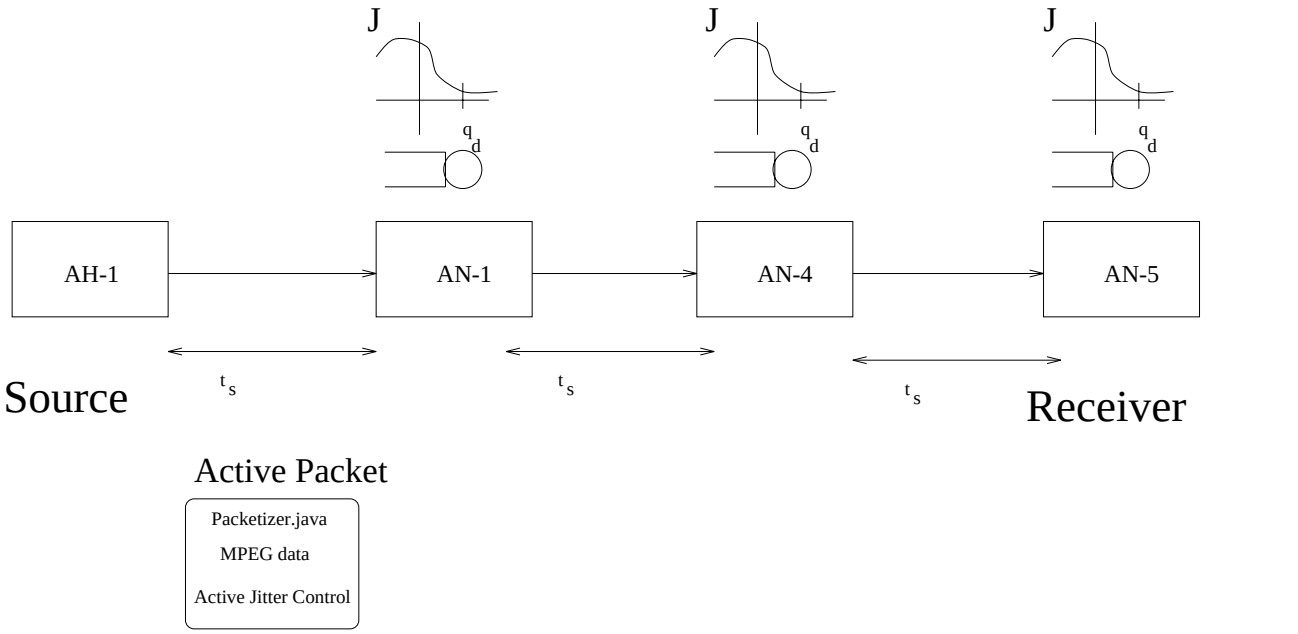


Fig. 2. The Active Network Video Configuration.

0.001 quantile of the transfer delay variance. In this case, the expected transfer time was 2962 milliseconds with a variance of 13996.9.

Similarly, Figures 5 and 6 show the jitter without and with the active jitter control applied respectively at the receiver i.e. node AN-5. The expected transfer time at the receiver without jitter control was 2608 milliseconds with a variance of 3.4×10^7. But with active jitter control the expected transfer time is 7624.56 milliseconds with a variance of 190796. It was observed that at all hops there was a clear trade-off in reduced jitter at the cost of additional delay.

From the figures, one can observe that the delay caused by the active jitter control technique increases the average end-to-end delay 2.57 times. Some of this delay can be attributed to the tradeoff in jitter at the cost of end-to-end delay. To obtain a low overall jitter one requires to delay each packet by a higher average value, which results in a higher average end-to-end delay for packet transmission. In addition, part of the delay and variation is due to a combination of the overhead of carrying a small amount of executable code in each packet and the time spent accessing SmallState. However, this paper presents on-going work, and one of our future goals is to quantify, these overheads and find ways to minimize them.

VI. New Active Jitter Control Capabilities

In this section some of the new jitter control capabilities provided by the active nature of the implementation are briefly outlined. It must be remembered that the processing time variance caused by executing these new capabilities should be included within the scope of the jitter control algorithm itself along with

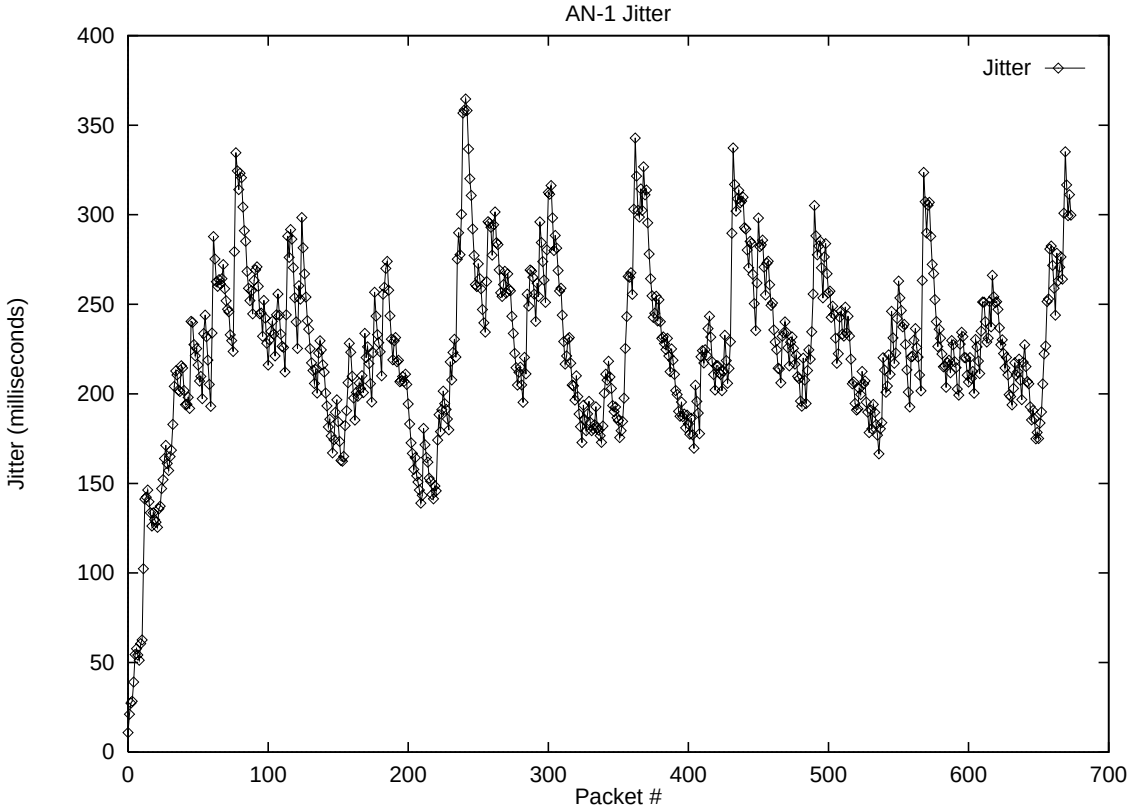


Fig. 3. Jitter at Node One without Active Control.

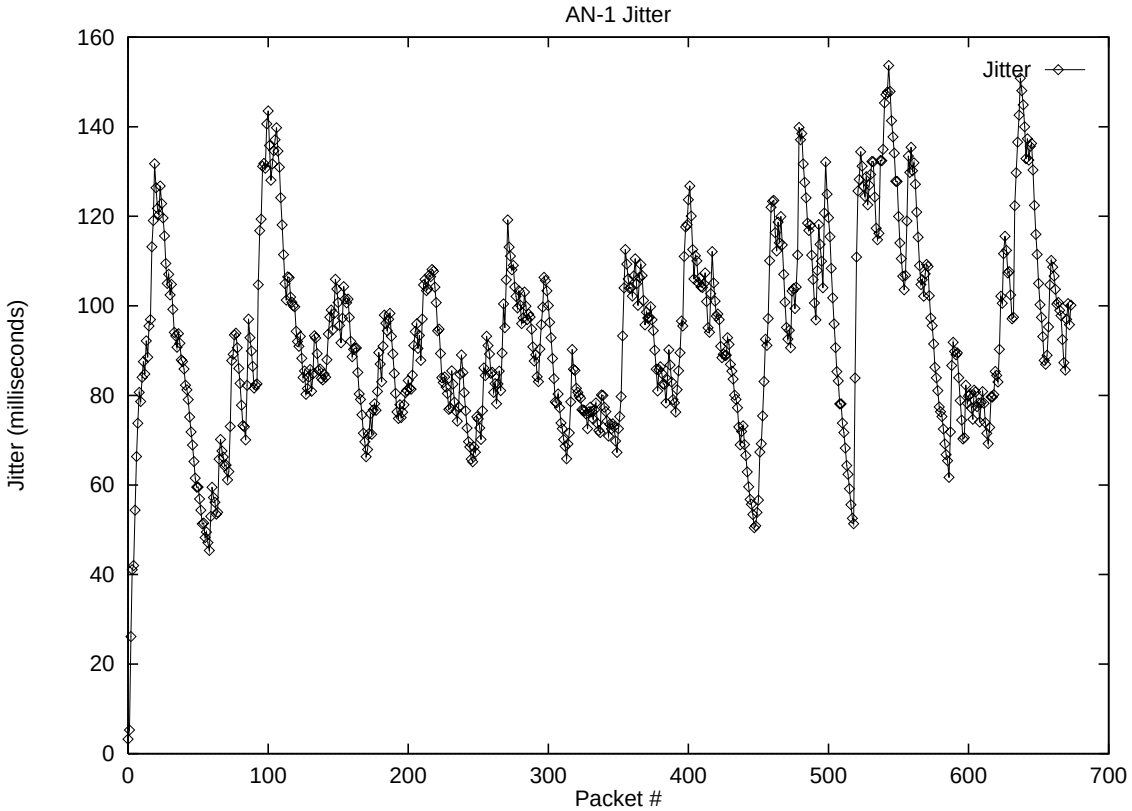


Fig. 4. Jitter at Node One with Active Control.

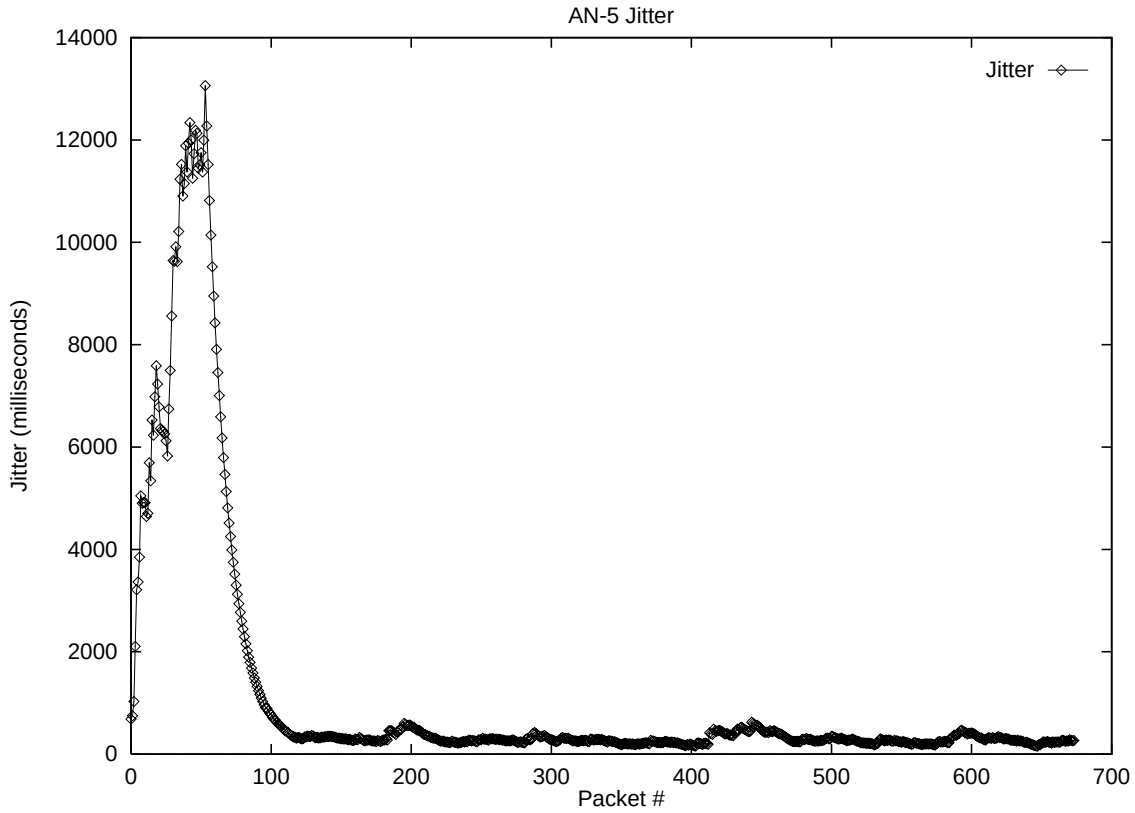


Fig. 5. Jitter at Node Five without Active Control.

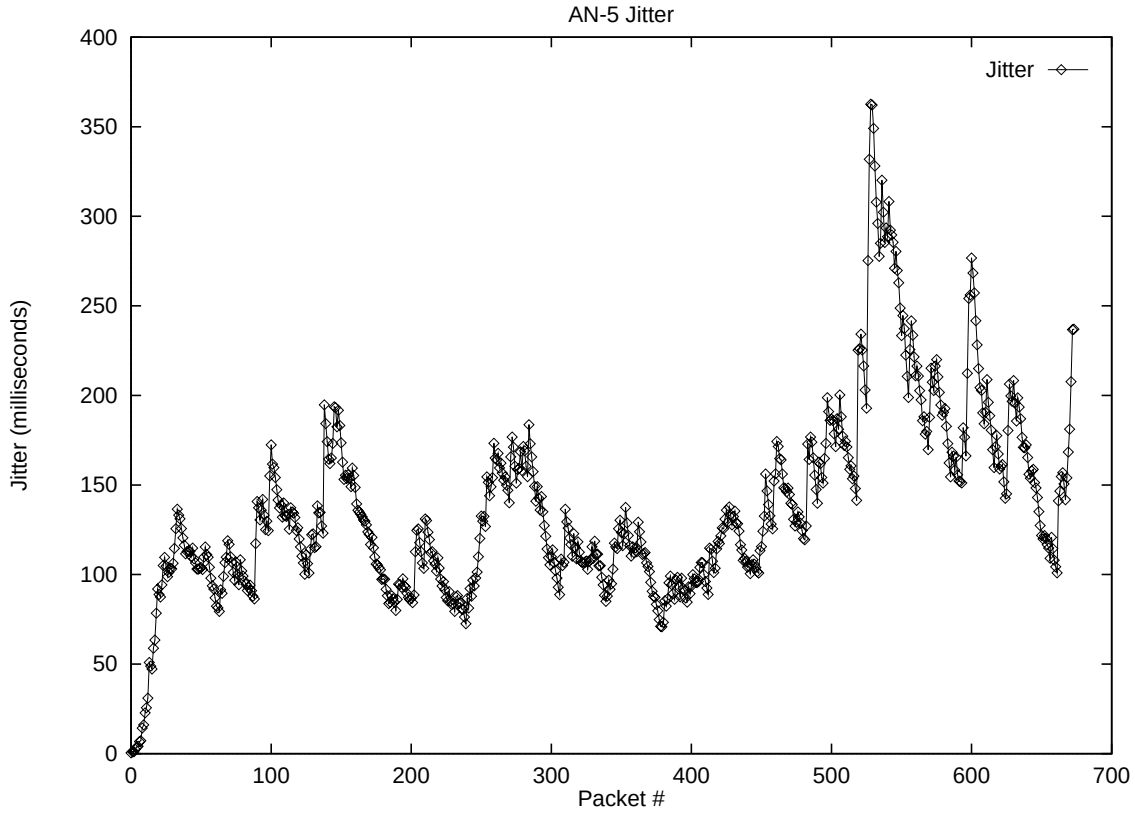


Fig. 6. Jitter at Node Five with Active Control.

the transfer time variance.

The objectives and constraints that would likely require optimization include jitter, end-to-end delay, and expected queue size at each node. Information specific to the real time information content which could be determined by the active packet include the importance of the individual packet to the overall quality and resolution of the image. For example a packet carrying data of relatively low interest or outside the user's view of the image can tolerate more jitter and longer delay and set the quantile accordingly.

The parameters under the control of the active packet are the packet size, jitter delay quantile, route to destination, and priority (scheduling). Although the work presented in this paper has only demonstrated active jitter control via the jitter delay quantile, all of the above mentioned parameters can be utilized by an active packet to affect the resulting real time product quality.

VII. Conclusion

This paper has shown that despite considerable challenges, active jitter control is not only possible, but opens up new opportunities for finer resolution and more sophisticated control. The advantages of the active network approach validated within this paper are the following. First, the jitter control is performed on a per packet basis, not per flow as in today's networks. This means that individual packets can vary their jitter control algorithm based on their overall value to the final product. For example, a packet providing visual background that is not in motion may have a much higher tolerance for jitter than a packet in a more dynamic portion of the visual or audio transmission. Another advantage is that active networks allow higher level control of the network in a dynamic manner. Thus, changing a jitter control algorithm does not require years of standardization and vendor compliance. The challenges are the additional complexity in working from within the packet instead of outside the packet and the additional variance caused by packet execution and SmallState access conflicts.

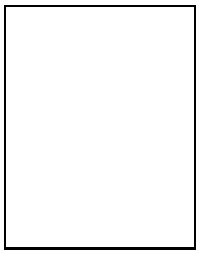
References

[1] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A transport protocol for real-time applications, January 1996. Status: PROPOSED STANDARD.

[2] Stephen F. Bush. Active Virtual Network Management Protocol. In *PADS '99*, May 1999.

[3] A. B. Kulkarni, G. J. Minden, R. Hill, Y. Wijata, S. Sheth, H. Pindi, F. Wahhab, A. Gopinath, and A. Nagarajan. Implementation of a Prototype Active Network. In *OPENARCH '98*, 1998.

[4] Dinesh C. Verma, Hui Zhang, and Domenico Ferrari. Delay jitter control for real-time communication in a packet switching network. Technical report, Internation Computer Sciences Institute, 1991. TR-91-007.



Stephen F. Bush Stephen F. Bush is a Computer Scientist at General Electric Research and Development (GE CR & D) in Niskayuna, NY. Steve is currently the Principal Investigator for the DARPA funded Active Networks Project at GE. Before joining GE CR & D, Stephen was a researcher at the Information and Telecommunications Technologies Center (ITTC) at the University of Kansas where he contributed to the DARPA Rapidly Deployable Radio Networks Project. He received his B.S. in Electrical and Computer Engineering from Carnegie Mellon University and M.S. in Computer Science from Cleveland State University. He has worked many years for industry in the areas of computer integrated manufacturing and factory automation and control. Steve received the award of Achievement for Professional Initiative and Performance for his work as Technical Project Leader at GE Information Systems in the areas of network management and control while pursuing his Ph.D. at Case Western Reserve University. Steve completed his Ph.D. research at the University of Kansas where he received a Strobel Scholarship Award. He can be reached at bushsf@crd.ge.com and http://www.crd.ge.com/people/bush.