SMDS MEASURMENTS AND MODELING TO PREDICT PERFORMANCE

S.H.Low
EECS Department
University of California
Berkeley, CA 94720

K.M. Nichols
Advanced Technology Group
Apple Computer, Inc.
20525 Mariani Avenue M/S 76-3K
Cupertino, CA 95014

ABSTRACT This paper describes a performance study of a trial Switched Multimegabit Data Service (SMDS) link from the perspective of customers evaluating the feasibility of the link for some target applications. The goals were to take all measurements on the customer premises and to develop a methodology general enough to be used by customers to evaluate the link.

We measured a lightly loaded system and developed a model of the SMDS connection suitable for evaluating applications via analysis or simulation. This paper documents our methodology, presents the SMDS connection delay values, and a likely breakdown of the constituents of that delay. We used this data to create a simulation model and to simulate a simple application.

In the trial configuration, where geographical distances were small, SMDS network delay was one of the notable components of end-to-end delay in the SMDS connection. However, for most packets, throughput is limited by the T1 capacity for transmitting SMDS cells, not by the SMDS network capacity.

I. INTRODUCTION

Switched Multimegabit Data Service (SMDS) is a datagram service intended for LAN interconnection which may be offered by Local Exchange and Interexchange Carriers[1,2,3]. SMDS is expected to be deployed as a public offering by Pacific Bell sometime next year. Initial service is proposed with T1 rate access paths (1.54 Mb/sec line speed or 1.17 Mb/sec data rate), and later service with T3 rate access paths (45 Mb/s line speed or 34 Mb/sec data rate). Apple Computer participated in a Pacific Bell trial using T1 access lines.

We describe a set of simple experiments that can be done on the customer premises to evaluate the service. Our aim was to treat the system as a "black box" and design experiments to deduce the essential functions of the box. These experiments were augmented by our general knowledge of the elements contained within the box (T1 links, switch, routers, etc.). From the measurements and our knowledge of the system architecture, we created a model that could be used for analysis or simulation. We implemented and validated a simulation model.

The measurements we made reflect the properties of the trial link and are likely to change when the service is deployed as a product. Our methodology, however, will still be applicable. This approach can be valuable for initial evaluation of delays and bandwidth by the service subscriber. Experiments could be performed periodically, as the switching hub becomes busier, to see if, and how, the performance changes.

The paper is organized as follows. Section II describes the trial connection. Section III presents our measurement experiments and how we deduce network performance from these measurements. In section IV, we develop an end-to-end model of the SMDS connection based on the experimental data. In section V we present a simulation model and some results using this end-to-end model. We conclude in section VI.

II. THE TRIAL CONNECTION

Our experiments took place on Apple equipment connected via the SMDS-T1 service provided by Pacific Bell for a trial period. Two locations were connected through the SMDS network and each site had an isolated Ethernet connected to the link via a Cisco router, an ADC Kentrox DataSMART™ segmenting (L2_PDUs) device, and associated cabling. The SMDS network[1] was a "black box" to

[1]In this document, "SMDS network" means the switching elements and associated hardware and "SMDS end-to-end connection" means the entire path from source to destination.

CH3133-6/92/0000-2444 $3.00 © 1992 IEEE

us and we used the measurements to create a model of its internals. Our sites were connected to the SMDS network via T1 links. On a path through the SMDS network, a message is processed at one or more switching stages, and finally onto the T1 link and reassembled at the destination. In the trial, there was only one switch.

Our experimental setup is shown in figure 1. In the figure, *ws1* is a Silicon Graphics SGI/PI; *ws2* is a SUN Sparcstation which was moved between the two Ethernets to perform the measurements. To obtain measurements on the Ethernet, we used a network monitoring program, NetMinder™ Ethernet, running on a Macintosh IIci; *monitor* denotes this machine. *E1* and *E2* are the Ethernets connected to the Cisco AGS+ routers, *router1* and *router2* respectively. Each router is connected to a Kentrox DataSMART™ segmentation and reassembly device (*DSU1* and *DSU2*) which is connected to the SMDS network by a T1 access link. We cannot measure the individual contribution of the the DataSMART™, but we know that it operates in cut-through mode with little overhead and we know the transmission rate of SMDS (L2) cells across the T1 link.

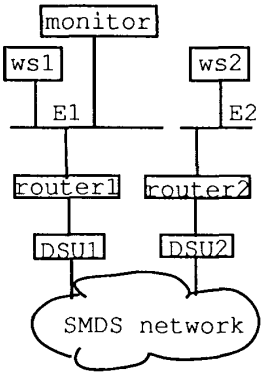

Figure 1. Experimental SMDS Connection

The experiment was conducted in a controlled environment. That is, the machines shown were the only ones generating traffic on the Ethernets, the routers were not connected to networks other than those shown, and the SMDS network was known to be very lightly loaded. In an SMDS network subject to increased traffic demands, we would expect to do more repetitions of the experiments and derive values for periods of light and heavy use.

III. EXPERIMENTAL MEASUREMENTS

Our measurement experiments are divided into two types: one set to determine the delay, or latency, of a single Ethernet frame through the network, and another set to determine the network throughput when a stream of packets is sent. For these experiments we used two Unix™ commands, *ping* and *spray*, and took measurements on *E1*.

A. Network Delay

Our objective is to estimate the delay of the entire SMDS connection. By this, we mean the time it takes *a single packet* to cross the SMDS network, the sum of the delay at each processing point along the path. (When a stream of packets are sent across the network, this delay is referred to as the network latency.) Our goal is a measurement methodology that can be performed on the customer premises using nonspecialized equipment. We use the Unix™ command *ping* together with Ethernet measurements for this purpose.

Ping sends a packet to the target machine, which is acknowledged by the target machine and the source machine reports the time at which this acknowledgement is received. Both the initial packet and the acknowledgement have the same size, specified on the command line. The round trip time reported by the operating system command has a precision on the order of 10 milliseconds, so we used the network *monitor* to obtain higher precision measurements and bypass the source machine's operating system overhead. In all the *ping* experiments, *ws1* was the source machine and *monitor* was used to record and timestamp every packet on *E1* sent by either *ws1* or *ws2*. Packets are timestamped when transmission begins [4].

We can *ping* the routers and *ping* from the routers, though we have no way of measuring packet times on the SMDS side of the routers. However, we can *ping* each router from *ws1* in independent measurement experiments.

We can obtain the round trip time from the time a request begins transmission onto the source machine's Ethernet until the time its acknowledgement begins to be transmitted on the same Ethernet. When

ws1 pings ws2, the delay consists of the transmission time at *E1*, the routing time at the two routers, the transmission time of the two T1 links, the SMDS network delay, the transmission time at *E2*, the *ping* acknowledgement processing time at *ws2*, and the return trip.

The major problem we faced in these measurements was that our measurement variance was in the millisecond range and some of the delays were also in this range. We generally made decisions to use or discard delays in this range based on whether we could verify them with precision or not.

Working from figure 1, the delays experienced by a packet can be decomposed into:

- E_tx: Ethernet transmission delay (packet size/10 Mbps)
- rtr_dly: processing delay at a router
- L2_tx: the time to send the segmented packet as L2_PDUs at T1 rate
- sw_dly: the SMDS network delay
- ack_ws: *ping* acknowledgement processing time at the target workstation
- ack_rtr: *ping* acknowledgement processing time at the target router

We assume that the fixed overhead at the DSUs and the transmission time between the DSUs and the routers are negligible. Now the one-way packet delay can be expressed as:

$$\text{delay} = 2\cdot(E\_tx+rtr\_dly+L2\_tx)+sw\_dly \quad (1)$$

We compute two quantities, E_tx and L2_tx and we measure two different ping round trips. We can ping the remote workstation, *ws2*, from *ws1* (ping_remote) and the remote router *router2* from *ws1* (ping_r2). We have:

$$ping\_remote = 2\cdot delay+ack\_ws-E\_tx \quad (2)$$

$$ping\_r2 = delay-E\_tx+sw\_dly+2\cdot L2\_tx+ack\_rtr$$
$$= 2\cdot(rtr\_dly+2\cdot L2\_tx+sw\_dly) +E\_tx+ack\_rtr \quad (3)$$

In addition, we made two sets of measurements on *E1* to get a measured estimate of the time it takes the target machine to turn around the acknowledgement. The routers are identical; we assume the processing times at each will be the same. The times we measure on the local network are just the target machine's turnaround time plus the time it takes to transmit the frame over the Ethernet at 10 Mbps. We *pinged* the local router *router1* from *ws1* (ping_r1). We physically moved *ws2* to *E1* and pinged it from *ws1* (ping_loc). Now we can write:

$$ping\_loc = E\_tx + ack\_ws \quad (4)$$
$$ping\_r1 = E\_tx+ack\_rtr \quad (5)$$

Combining (3) and (5):

$$sw\_dly = \frac{ping\_r1-ping\_r2}{2} - rtr\_dly-2\cdot L2\_tx$$

and combining (1), (2), and (4) gives:

$$sw\_dly = \frac{ping\_remote-ping\_loc}{2} -E\_tx-2\cdot rtr\_dly-2\cdot L2\_tx$$

These two equations can be solved for the unknown values, sw_dly and rtr_dly, but our rtr_dly is less than 1 millisecond. This is on the order of our measurement variance, so we choose to ignore rtr_dly and obtain:

$$sw\_dly = \frac{ping\_r1-ping\_r2}{2} - 2\cdot L2\_tx \quad (6)$$

and

$$sw\_dly = \frac{ping\_remote-ping\_loc}{2} -2\cdot L2\_tx-E\_tx \quad (7)$$

Several hundred measurements were taken at each of eight different packet sizes, but the results at each packet size showed little variation. Average measured values at each packet size are recorded in table 1.

Table 1: SMDS measurements (milliseconds)

size (bytes)	ping remote	ping router2
64	17.4	16.9
200	24.9	23.3
400	33.9	33.5
600	45.3	43.5
800	55.0	53.5
1000	66.2	63.5
1200	76.0	73.5
1400	87.2	83.5

The Ethernet delay, E_tx, is computed as the size of the packet in bits divided by 10 Mbps. To compute L2_tx we note that each Ethernet frame of size *s* bytes is first stripped of its 18 byte Ethernet header then 40 bytes of header are added to form an L3_PDU[1].

The L3_PDU is then segmented into a sequence of 53 byte L2_PDUs, each of which contains 44 bytes of information and 9 bytes of L2 header. Ten L2_PDUs are transmitted across the SNI on the T1 access link every 3 milliseconds. Hence the effective data rate is:

$$L2\_tx = \text{ceil}\left[\frac{s+22}{44}\right] \times \frac{3\ ms}{10} \quad (8)$$

Our measurements found both acknowledgement times (ack_ws and ack_rtr) take an average of 5 milliseconds, so ping_loc = ping_r1 = 5+E_tx.

The results of computing sw_dly using both equations (6) and (7) are listed in table 2. The differences between the two are less than a millisecond, in most cases, much less.

Table 2: Computed values (milliseconds)

size (bytes)	E_tx	L2_tx	SMDS (6)	SMDS (7)
64	0.1	0.6	4.8	4.7
200	0.2	1.8	5.9	6.8
400	0.3	2.9	8.3	8.0
600	0.5	4.2	10.3	10.3
800	0.6	5.6	12.8	12.7
1000	0.8	7.0	14.7	14.9
1200	1.0	8.3	17.2	17.3
1400	1.1	9.7	19.1	19.5

Since the two results were so close, either would suffice. Using equation (6) does not require pinging the routers while using equation (7) requires that the workstations being pinged are carefully controlled. We will use (6) for the remainder of this paper.

B.Validation

Although we present a methodology that can be employed strictly from the customer premises, we had the opportunity to validate our measurements at Pacific Bell's SMDS switch site. There we took a set of *ping* measurements between workstations, then bypassed the SMDS network and repeated the experiments. The difference between these two values, divided by two, is the delay through the SMDS network. These values are recorded in table 3.

To compare the SMDS delay of table 3 to the values we recorded at our site, we must consider the propagation delay over the T1 lines. The SMDS delay in table 2 includes this propagation delay. Computing the propagation delay of the T1 lines as in [5], i.e. multiply the distance in miles by 0.016 ms/mile, and using the fact that the T1 links traversed around 100 miles, we obtain a value of 1.6 ms and subtract this from the SMDS delays in table 2.

Table 3: PacBell measurements (milliseconds)

size (bytes)	ping remote	ping no switch	SMDS delay
100	21	14	3.5
300	32	20	6.0
500	41	25	8.0
700	54	30	12.0
900	60	36	12.0
1100	71	41	15.0
1300	82	46	18.0

The PacBell setup used a different router and segmentation/reassembly unit at one end of their connection, but this does not affect the validation.

In figure 2, the SMDS delay curves resulting from the two sets of measurements are plotted. We denote the measurements made on our premises with CPE and those made at Pacific Bell with PacBell.

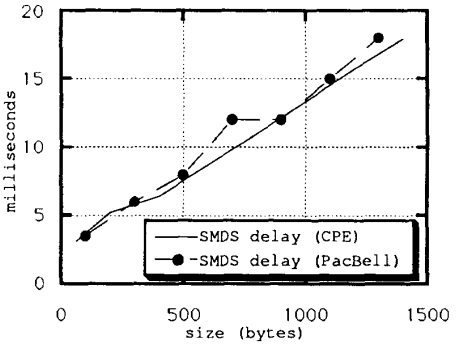

Figure 2. SMDS Switch Network Delay

Since the SMDS network delay increases linearly with size, a linear fit program was used to find the best linear curve to fit to each set of data in figure 2. The CPE line is a particularly good fit to the data. We can express sw_dly as a function of packet size *s* (in bytes) by:

$$sw\_dly_{CPE} = 2.41+0.011 \cdot s \text{ ms} \quad (9)$$

$$sw\_dly_{PB} = 2.45+0.012 \cdot s \text{ ms} \quad (10)$$

Our understanding is that sw_dly includes some packet size dependent processing, accounting for the linear growth in packet size. Since our measurements have a certain amount of error and imprecision, we believe these results are satisfactory.

C. Network Throughput

Now that we have determined the packet delay, we would like to deduce the network throughput. Throughput is defined as the network output rate when the input is loaded with a stream of packets. One method of determining this value is to load the input at a known rate and measure the output. Since our two locations were separated by some miles, we employed a variant of this technique. We used the Unix *spray* command to load the network.

The experimenter was located at *ws1* and remotely logged in to *ws2*. In a *spray* experiment, *ws2* sends *ws1* a stream of 1000 packets of packet size *s* at fixed interval *t*. After receiving the last packet, the target machine sends an acknowledgement with the total number of packets received and the total reception time. Although the smallest *t* that can be set on the command line is 10 ms, it is possible to use *spray* to send 1000 packets *as fast as possible* by setting the interpacket time on the command line to zero. With this approach, we could measure the throughput of a saturated network[2].

We used the network monitor to obtain more precise measurements. The monitor counted the number of packets received on destination *E1*, while *ws1* counted the number of packets it received. Thus, the first measurement is an estimate of the SMDS network throughput and the second is the number which got through the SMDS network and were not dropped by *ws1*. We measured the total number of packets, N_p, on *E1* during the reception time, t ms, and computed the throughput as $s \cdot N_p/t$ Kbps. For sufficiently large input loads, the number of received packets reported by the monitor was indeed higher than that reported by *ws1*, suggesting that *ws1* has a smaller throughput than the SMDS network due to operating system overhead.

We were unable to measure input load on *E2* at the same time, thus we record throughput as a function of the packet size used. We can expect throughput to increase with packet size until the upper bound is reached. Figure 3 plots the measured network throughput. The alternating pattern is a result of the unused portions of the L2_PDUs reflected arithmetically by the ceiling function.

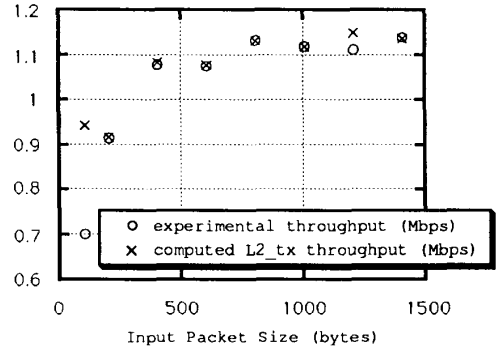

Figure 3. Maximum SMDS Network Throughput

We found that the throughput is approximately equal to the access link throughput determined by L2_tx, except at the smallest packet size. Thus the end-to-end throughput appears to be limited by the SMDS network only at the smallest packet sizes.

IV. A MODEL BASED ON THE MEASUREMENTS

In this section we propose a simple model for the SMDS link based on our experimental data on network delay and throughput.

We now have the model of figure 4. Packets are generated and received by the workstations, sent over the Ethernet, segmented into L2_PDUs and sent over the T1 link by a SAR (segmentation and reassembly) module to the SMDS network where they emerge on the destination T1 link to be reassembled at the SAR, transmitted over the Ethernet, and received by the destination workstation. The SAR delay is given by equation (8), the Ethernet delay by dividing packet size by the 10 Mbps transmission rate, and packets

[2]Since packets were dropped in the connection, we could verify that each point resulted in the maximum throughput for that packet size.

should be delayed by a fixed propagation delay based on distance as discussed above. We now need a model of the SMDS network delay. Of course, this is not completely accurate physically but leads to a model of reasonable accuracy.

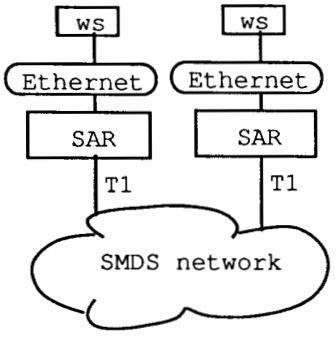

Figure 4. SMDS connection model

In the previous section, we found that throughput was limited by L2_tx in all our measurements except at 106 bytes. If we assume that the SMDS network can be represented by a single server with service time given by equation (9), we find that this yields throughput values far below those experimentally obtained. Indeed, the peak value (for large packets) is 0.6 Mbps, which is clearly not correct.

This suggests that the model of the SMDS network must consist of more than one stage of service. A set of N store-and-forward servers in tandem where each has a processing time of

$$d_i(s) \quad i = 1,2,...N$$

has a total end-to-end delay of

$$D(s) = \sum_{i=1}^{N} d_i(s)$$

as a function of packet size s bytes, yet the throughput is limited only by the processing delay of the slowest stage. Hence, the throughput, $s/d_{max}(s)$, where $d_{max}(s)$ is the largest $d_i(s)$ may be much larger than s/D(s). Alternatively, there may be some stages operating in cut-through fashion, but this would be impossible to determine from our data and may create a more difficult model.

Let d(s) be the SMDS delay for a packet of size *s*. We propose that the delay of the throughput limiting service stage take the form:

$$d_{max}(s) = m \cdot s + c$$

and we estimate the parameters m and c from our measurements. Equating the throughput of this stage to the throughput measured in our experiment we have:

$$\frac{106 \cdot 8}{m \cdot 106 + c} = 0.7 \text{ Mbps} \qquad (11)$$

Since we lack a second point[3], we considered two options. The first was to use the same slope values as the delay in equation (9), 11 microseconds/byte. However, this leads to an equation that limits throughput at about 0.7 Mbps for all packet sizes, clearly too small. As an alternative, we guess a value for the 200 byte packet that is just above our experimentally determined value of 0.91 Mbps, although this may underestimate the SMDS throughput. Thus we use:

$$\frac{206 \cdot 8}{m \cdot 206 + c} = 1.0 \text{ Mbps} \qquad (12)$$

Combining equations (11) and (12), we obtain m=4.8 μsecs/byte and c=740 μsecs, a server with delay

$$d_{max}(s) = 4.8 \cdot s + 740 \text{ μsecs.} \qquad (13)$$

If we attribute the rest of the delay, 6.4·s+1670 μsecs, to a single server, this server will limit throughput to a value less than that obtained experimentally. We divide the rest of the delay between two store-and-forward servers, basing this on our understanding of the SMDS switch having a symmetric arrangement. Thus, we represent the SMDS network by three tandem servers, one with service delay given by (13) and two with service delays given by:

$$d(s) = 3.2 \cdot s + 835 \text{ μsecs.} \qquad (14)$$

Note that the predicted throughput is the minimum of all the servers, rather than a single curve. This is not the only way to model the SMDS network. We might assume that the delays can be pipelined, and then have

[3]Clearly, we would recommend finding a second limiting point, if possible. The trial network was disabled shortly after we took our measurements.

only two server stages, one at 4.8 microseconds/byte and the other at 6.4 microseconds/byte.

This section has presented a model for an end-to-end SMDS connection. The service times of the T1 stages are given by equation (8); the SMDS network comprises three stages, one with service time given by (13) bracketed by two stages with service time given by equation (14). In this model, packets at and above 200 bytes are limited by the process of converting layer 3 packets (L3_PDUs) into L2_PDUs and transmission on the T1 access links.

V. SIMULATION MODEL AND RESULTS

In this section, we outline the event-driven simulation based on the model developed in the previous section. The simulation model is validated and a simple application is simulated. We discuss the limitations of this model.

A block diagram of the simulation is shown in figure 5. The path is bidirectional, so each link consists of two one-way connections in either direction. For simulation purposes, we assume experimental, unloaded (or lightly loaded) Ethernets so that we may dispense with a complex model of the Ethernet. The three stages of the SMDS network we modeled in section IV are shown as two identical access modules and a switch module.

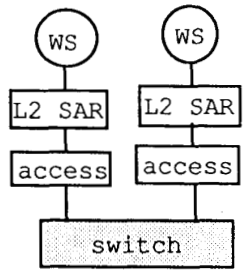

Figure 5. SMDS Network Simulation Model

An entire Ethernet packet is received by the L2_SAR before being sent over the T1 link as L2_PDUs, so each packet is delayed by its transmission time on the Ethernet before it is passed to the L2_SAR module. For simulation purposes, both the Ethernet and the T1 propagation delays can be accounted for by scheduling the arrival event of a packet at the source side's switch access module from the source's L2_SAR (or at the destination L2_SAR from the switch access) at the appropriate delays.

The L2_SAR holds a packet for a time determined by equation (8). The access modules have service time given by equation (13), and the switch module service time is given by equation (12). A workstation is considered to be busy for the amount of time it takes to send the packet over the Ethernet only. The other modules are considered to be busy for the service delay time only. The modules service traffic in both directions, but not concurrently.

We have insufficient data to determine when packets are dropped in the network, thus all packets are buffered. When each packet is completely received, the delay from the time the packet was created until the time the packet is received is recorded and this value is used to compute the average delay with a confidence level of 95% at +/-10% precision.

A. Validation

The first simulation experiment verifies that the simulated SMDS connection delay is sufficiently close to the measured delay. The values are presented in table 4; the measured delay is computed from equation (2) and the values in tables 1 and 2. These results are quite close, as one would expect.

Table 4: Delay validation

size (bytes)	simulated delay	measured delay
64	6.0ms	6.2ms
200	10.2	10.0
400	15.1	14.5
600	20.7	20.1
800	25.7	25.0
1000	31.2	31.0
1200	36.2	36.5
1400	41.7	41.7

Next, the throughput of the simulation model is compared with the measured throughput. The connection is loaded by sending each packet size at intervals resulting in a rate of 1.3 Mbps. In figure 6, the simulated values are plotted against the measured values from figure 3. Once again, we have generally close agreement.

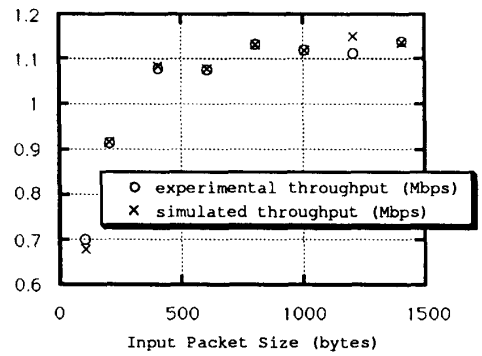

Figure 6. Throughput validation

A few caveats are in order. These simulations have not additional traffic in the switch. We do not have sufficient information to model this in any way that is not purely speculative. For example, we don't know if the only sharing is at the central stage, or if all three stages are shared, or what type of queueing for service is used. Cut-through operation would have a significant effect on the results.

B. A Multimedia Application

In this application, we assume two-way communications of variable bit rate video data with additional voice information added. This model can be viewed as a video teleconference or as the viewing of a digital "movie" file across the SMDS link. It is explained in more detail in [7]. The video part of the data comes from simulated encoder data for 300 frames of a 112 by 160 pixel video conferencing sequence, compressed to an average bit rate of 64 Kbits/sec and transmitted at a rate of 15 frames/second. We continuously loop through these 300 samples and double the original rate, from 15 frames/second to 30 frames/second to simulate higher quality video.

The average video frame size is 533 bytes/frame. Every 120 frames the intraframe (or key frame), a full frame of nearly 2000 bytes, is generated. In this scheme, each complete video frame is generated, audio is added, and the result is sent as one message. Since the Ethernet maximum packet size is 1526 bytes, all messages larger than this (primarily the intraframes) will be broken into two packets by the protocol. To eliminate acknowledgement traffic, the system under study is assumed to use the UDP protocol, adding 44 bytes of overhead to each Ethernet packet.

Each workstation in figure 5 is both a source and a receiver for packets. As a source, it generates frames at the frame rate. As a receiver, frames are dropped if they arrive too late to be used. When the initial frame (a two packet intraframe) is completely received, a clock is started at the receiver that is continually incremented by one frame time. If the next frame does not arrive precisely at or before the next clock increment, it is dropped. Delaying the start of the receiver clock (corresponding to display of the first frame) can allow more variation in the delay between transmitter and receiver.

We are interested in the percentage of packets which are dropped and whether these packets represent intraframes. The latter is important for two reasons: 1) the two-packet intraframes are more likely to arrive late and 2) when an intraframe is dropped, the video data which follows will be useless until the next intraframe is received. The percentage of dropped packets must be kept small since there is no method of recovery other than waiting for the next intraframe. This is likely the only feasible way to do video conferencing on a network with so much latency since feedback and recovery would longer than a frame time. Furthermore, additional short feedback packets would degrade the performance of the entire end-to-end system.

The average packet size was 716 bytes, and the average packet delay was 25.5 ms at 15 fr/sec and 26.8 fr/sec at 30 fr/sec. These numbers are consistent with our test results; the packets are delayed slightly due to contention with traffic in the other direction. Switch utilization was 13% at 15 fr/sec and 26% at 30 fr/sec. The average time between received packets was equal to the frame rate, with 98% of all frames arriving within 13 ms of this average, either early or late. This variability led to some frames being dropped and an occasional single frame being buffered at the receiver. In the first simulations, about one percent of the frames were dropped, but all of them were intraframes. Since intraframes made up about one percent of the frames sent, this represents

nearly all of the intraframes, rendering the video unintelligible.

To decrease the number of dropped frames, we delayed the start of the receiver clock (i.e., the display of the first frame is delayed) by 6.7 ms, 10% of a frame at 15 fr/sec or 20% of a frame at 30 fr/sec. Delay remained the same, but no frames were dropped.

Additional data traffic through the connection would increase the packet delay and, perhaps more importantly, the variation of that delay. This may necessitate buffering more frames at the receiver. Buffering frames would not affect a movie-playing application, but could cause synchronization problems for a video conference. We also caution that more variation is introduced on an Ethernet carrying additional data traffic.

Our results show that it is possible to use the SMDS connection for this application, but some delay in the start of the receiver clock, or short buffering of the initial frame, is required. Further, we may experience unacceptable degradation as the switch network traffic increases.

VI. CONCLUSIONS

We presented a set of simple experiments to measure the SMDS network performance. Although the deployed system may differ, we believe the *methodology* to evaluate the service will remain applicable. We measured the trial connection and found an end-to-end delay well within the guaranteed maximum of 140 ms at T1 offering. The connection delivers a throughput very near the T1 rate for large packets and is limited by the switch processing only for very small packets. When the endpoints are widely separated geographically, propagation delays become increasingly important.

SMDS is intended primarily for inter-LAN connection and data applications. It should function well for this purpose since file transfer applications are not delay-sensitive and usually comprise large packets. Further, applications that produce small packets often produce packets infrequently and do not require large bandwidth. On the other hand, any application, or set of applications composed of a stream of small packets may create large queues in the network and cause large end-to-end delays.

We developed a model of an SMDS connection based on the measurement data and used it to simulate a multimedia application over the SMDS connection. SMDS was not designed to serve continuous media applications, but we found acceptable performance under our limited conditions.

Future work may include more complete simulations, with more accurate LAN models and a more varied workload. The effects of additional traffic in the switch should be included since we feel this will have an important effect on performance. We would like to extend our model to the proposed SMDS-T3 service, but we need a cost-effective way to monitor faster than Ethernet rates.

Acknowledgements

The *ping* experiment was based on previous work by Michael Wong of U.C. Berkeley, Joseph Pang of Pacific Bell and S.H. Low. We express thanks to Professor Pravin Varaiya of U.C. Berkeley and Frank Liu of Pacific Bell for useful discussion and to Scott Stein, Terry Pass, and Janine Roeth of Apple for help setting up the experimental equipment.

REFERENCES

[1] Bellcore, TR-TSV-000772, "Generic System Requirements in Support of SMDS", Issue 1, May 1991.

[2] David M. Piscitello and Michael Kramer, "Internetworking using SMDS in TCP/IP Environments", Computer Communications Review, July, 1990.

[3] C. Hemrick and L. Lang, "Introduction to SMDS, an early broadband service", Proceedings of the XIII International Switching Symposium (ISS 90), May 1990.

[4] Neon Software, "NetMinder™ Ethernet user's manual", 1990

[5] M.A. Rodrigues and V.R. Saksena, "Performance Analysis of a LAN/WAN Bridging Architecture", IEEE Journal on Selected Areas in Communications, Vol 9, No.2, February, 1991, pp. 265-270.

[6] G.H. Clapp, "LAN Interconnection Across SMDS", IEEE Network Magazine, September, 1991, pp. 25-32.

[7] K.M. Nichols, "Network Performance of Packet Video on a Local Area Network", Proceedings of the IEEE International Conference on Computers and Communications, March, 1992