Experiments with Video Communication on ATM-networks[*]

Thomas Husfeldt, Brian Nielsen[†] Finn Norman Pedersen, Dao Van The

June 18, 1997

Abstract

This report presents the results of a set of performance measurements related to communication of digital video on ATM-networks. High quality video produces large amounts data which must be communicated and processed in real-time. Satisfaction of this requirements require knowledge about the available system resources and the nature of the load that are put on these. We therefore benchmark our testbed consisting of a local area ATM network and a local area Ethernet to investigate the available resources and compressed video's use of bandwidth and cpu resources.

We examine three protocol/network combinations with respect to. throughput, latency and jitter: AAL-5 on an ATM-network, UDP on ATM, and UDP on Ethernet. The measurements show that AAL-5 gives the highest throughput, lowest latency, and lowest jitter. We find that a loaded Ethernet produces very high jitter values, that require special attention in a multi-media system. We conclude that AAL-5 is the better choice of the three protocols for transmission of high bandwidth real-time sensitive traffic.

We analyse and compare two video compression techniques, MPEG-1 and MPEG-2. We record and compress a test video which we then analyze for its usage of bandwidth and the cpu-time required compress and decompress it. Our analysis indicates that MPEG-2 gives a better quality/bandwidth ratio than MPEG-1, and also that the variation in bandwidth and cpu usage is smaller, and thus is easier to manage.

For transmission of live video, the combination of AAL-5 and MPEG-2 gives the best result: The lowest end-to-end delay and the lowest variation in end-to-end delay.

Our new insight in the real-time performance characteristics of communication protocols for video communication and of compressed video have given us a solid foundation for designing and constructing multi-media applications and support systems.

[*]This work has been carried out in the research programme "Networks and Paradigms for the Next Generation of Distributed Systems" which is supported by the Danish Research Council, SNF. We like to thank Birger Andersen, Wladyslaw Pietraszek, and Arne Skou, for commenting a draft version of this report.

[†]Author for contact. E-mail: `bnielsen@cs.auc.dk`

Contents

1 Introduction

1.1 Distributed Multi-Media Applications

In the near future distributed multi-media applications like video conferencing, computer supported cooperative work, remote education, and tele-robotics will become feasible. A decisive factor is the introduction of new high speed network technologies that can handle the bandwidth and real-time requirements put forward by the new application-types. In addition to the transmission quality delivered by the lowest level in the network, also the applied communication protocols and other computer systems software have significant influence on the achievable application quality.

Often distributed multi-media applications require video to be communicated over the network. Video is a very demanding data type to support because high volumes of data need to be transmitted on the network and processed at hosts. To save bandwidth video frames are compressed by the sender before transmission, and subsequently decompressed by the receiver, before they are displayed. Normally compression reduces data volumes by about 20-30 times, but even a compressed video stream in a decent quality is fairly demanding. For example, VHS quality compressed video uses approximately 2 Mbit/s of bandwidth, PAL broadcast quality uses 4-6 Mbit/s, studio production quality uses 8-9 Mbit/s, and finally, high definition TV requires more than 15 Mbit/s. Video conferencing type applications can usually do with a lower quality than TV signals. A streams quality can be lowered by reducing picture size, using a lower frame rate, and a less accurate compression (more lossy).

There are two primary disadvantages of using compression: First, it produces variable bit rate (VBR) traffic because not every frame can be compressed by the same amount. VBR-traffic makes it more difficult to manage network resources than traffic with a constant bit rate. Second, compression/decompression put heavy and variable computational loads on the hosts.

In addition to the bandwidth requirements numerous other real-time constraints apply. For example, the total end-to-end delay, from a frame is grabbed until the frame is replayed, should be less than 250 ms to preserve the user's impression of participating in a live interaction with each other. The frames of the video stream must be replayed periodically with a jitter tolerance of about 10 ms to avoid visible distortion. Similarly, audio should be "lip-synchronized" with the video such that a video frame and the corresponding audio-frame are replayed within 120 ms of each other. It is important to realize that these requirements are total *end-to-end* requirements, and cannot directly be converted to host and network requirements. For example, fast hosts and buffering techniques can to some degree compensate for a network with much jitter. Likewise, the choice of compression technology can compensate for slow hosts by trading processing time for lower compression rate (thus higher bandwidth usage). This reports evaluates a couple of specific protocol and compression technology configurations.

The quality of applications and the techniques used to implement them are sensitive to the network and protocol performance. Relevant metrics include throughput, latency and variation in transmission delay (termed jitter in the following). Throughput influences the number of streams that can be supported. Network latency determines how fast hosts can exchange synchronization information. Nework jitter causes a variable delay which the scheduling and buffering strategies must take into account to satisfy real-time constraints.

1.2 The Experiments

To evaluate the significance of these influences we have designed and performed a set of basic experiments which benchmark different aspects of typical networks, protocols, and compression techniques used to implement distributed multi-media applications. This report contains our findings. The experiments fall into two categories:

Network communication: These experiments compare two network technologies, Ethernet and asynchronous transfer mode (ATM), and two protocols, the ATM-native adaptation layer 5[1] (AAL-5), and the internet protocol User Datagram Protocol (UDP/IP). We evaluate three combinations, AAL-5 on ATM, UDP on ATM and UDP on Ethernet, wrt. achievable throughput, average latency and jitter probability distribution. The affect on jitter by a loaded network is also examined.

Compressed video: These experiments provide insight in the bandwidth, computation compression and decompression load produced by a typical compressed video fragment. We look both at the MPEG-1 and MPEG-2 compression techniques.

Figure 1 provides an overview of the network and MPEG experiments.

ATM has been announced as the future high-speed network technology, and has from the beginning been designed to support different classes of services, and is thereby able to carry constant, variable, or available bit-rate traffic, with loose or strict timing requirements. Further, when an application opens a connection it specifies the level of quality of service (e.g., service class and bandwidth) it needs. The network's admission control function uses this information to reserve resources for that connection, or possibly reject it. ATM usually only provides a soft (statistical) guarantee for the provision of the requested level of service. If, however, an application should exceed its allocated bandwidth, the network's policing function lower the drop-priority of the outstripped cells, or drop them altogether, depending on policy. Current Ethernets do not have

[1]Normally ATM adaptation layer 2 (AAL-2) would be the prefered protocol to transfer variable bit rate video traffic, but the ATM network in our testbed (see Section 1.3) does (currently) not implement AAL-2. Instead we use AAL-5, intended for general purpose data traffic.

these capabilities, and we therefore expect ATM to provide a more predictable performance than Ethernet, particularly on loaded networks.

	Network Experiments					
	SS20-SS20			U1-U1		
	AAL-5	UDP_{atm}	UDP_{eth}	AAL-5	UDP_{atm}	UDP_{eth}
Throughput	6	5	5	6	5	5
Latency	7	7	7	7	7	7
$\text{Jitter}_{\text{unloaded}}$	12	13	14			
$\text{Jitter}_{\text{loaded}}$	15	16	17			

MPEG Experiments		
	MPEG-1	MPEG-2
Bandwidth Profile	22	24
CPU decompression Profile	25	26
CPU compression Profile	28	29

Figure 1: Overview of the Experiments. The table contains the figure numbers of the figure with the result.

Note that both of the examined protocols are only partly reliable. Both CRC-checks the payload, but neither retransmits lost or faulty data. We chose these protocols in preference to the fully reliable Transmission Control Protocol (TCP) because retransmission is rarely necessary or even desirable for video. First, TCP's unbounded retransmission may produce a high and unpredictable delay on data units. For real-time applications it is usually more important to get recent data rather than complete and ordered, but late, data. Thus, it is usually better to skip a late video-frame rather than waiting for it, and consequently also delaying subsequent frames. Moreover, many compression techniques are designed such that the receiver can recover from a faulty frame or pixel block. Thus, retransmission can be avoided.

1.3 The Testbed

All experiments were run on an actual network. The testbed is a local area network consisting of an ATM network, an Ethernet, and 4 hosts configured as double homed internet hosts. The testbed is illustrated on Figure 2.

The following components were involved:

- Two identical Sparcstation 20 (SS20) workstations running Solaris 5.5.1 operating system from Sun Microsystems. Each workstation has *two* SuperSPARC 50 Mhz CPU's and 96 Mbytes RAM (specInt95 rating $<$ 2.46[2][5] per processor).

[2]for 75Mhz processor version

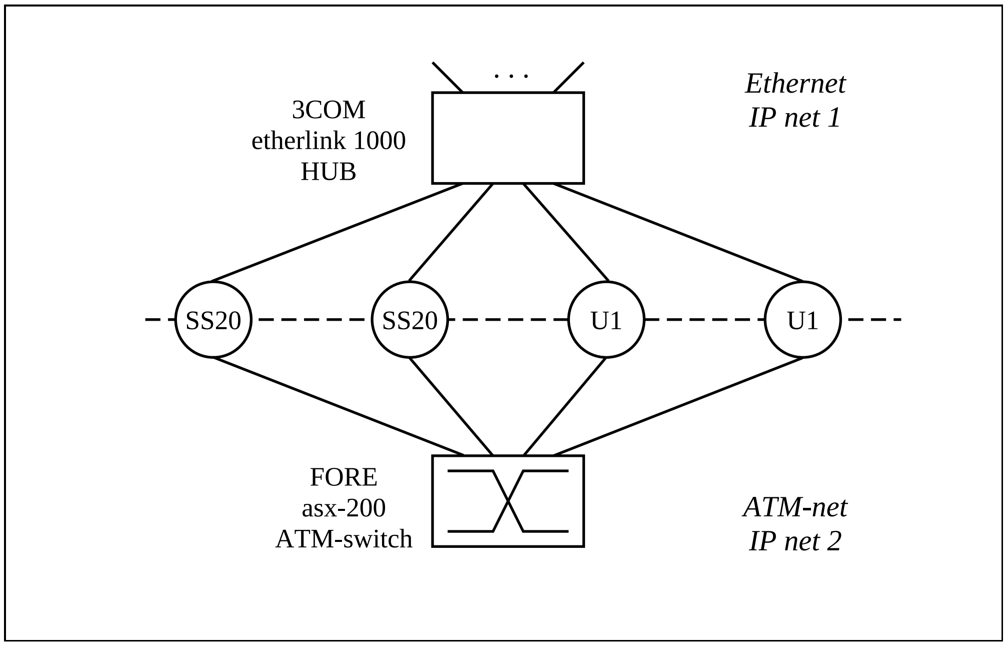

Figure 2: The testbed consists of both an ATM network and an Ethernet.

- Two identical Sparcstation Ultra 1 (U1) workstations running Solaris 5.5.1. Each workstation has a single ultraSPARC 143 Mhz CPU and 64 Mbytes RAM (specINT95 rating: 4.81[5]). All 4 workstations can be characterized as modern workstations, but has no special graphics capabilities, such as graphics hardware accelerators.

- Two ATM SBA-200 Sbus adapters connected to the SS20 hosts, and two ATM SBA-200e Sbus adapters connected to the U1 hosts. Both adapters are from Fore Systems.

- ASX-200 ATM switch from Fore Systems equipped with two four-port UTP-5 network modules. Both network modules have maximum port speed of 155 Mbit/s.

- An IEEE 802.3 type Ethernet. This net connects all computers in the computer science department, and the Ethernet in the testbed is therefore shared with other computers. However, all 4 hosts in the testbed were connected to the same HUB, a Linkswitch 1000 from 3COM. The HUB is configured to forward Ethernet packets in a “fragment free forwarding mode” which gives a forwarding delay of 64 μs[1]. The Ethernet provides 10 Mbit/s between any two hosts in the testbed.

All ATM experiments were conducted with enabled Fore Systems’ proprietary signalling protocol (SPANS) used for establishing switched virtual circuits (SVC) on demand. The ATM network does not carry any traffic not produced by the experiments. Applications access the network through Fore-Systems (sockets like) API.

To obtain as accurate time measurements as possible, disturbing effects caused by unrelated system activity were minimized by scheduling all test-programs in Solaris' real-time mode, and by executing the tests when the network was expected to be lightly loaded. Processes scheduled in real-time mode have higher priority than competing user and system processes, but lower priority than interrupts.

To measure elapsed time we use a nano-second resolution real-time timer. This timer is unrelated to the systems real-time clock, and not subject to resets and drifts performed manually or by network time synchronization protocols. The timer is accessed through the `gethrtime` system call.

1.4 Fore Systems QoS

Since a connection's performance depends on its quality of service settings, we here briefly describe how QoS is set in the Fore Systems API, and describe what settings are used in the experiments. Additional information about the testing methods are described in the sections containing the test result.

Before a connection is established, the application and network negotiates the connection's QoS. The application states a certain desired target level and a minimum acceptable level. The network replies with the actual allocated QoS level; this is called the selected QoS. The network may not be able to honour the target level, and consequently, the selected QoS may be less than the target. If the network is unable to honour the minimum level, the connection is rejected. Figure 3 shows the adjustable parameters.

	Peak bandwidth	Mean bandwidth	Mean Burst length
Target	154000	154000	95
Minimum	0	0	95
Selected	154000	154000	95

Figure 3: QoS Parameters

The parameters have the following meaning [22]:

Mean bandwidth: the average bandwidth expected over the lifetime of the connection, measured in kilobits per second.

Peak bandwidth: the maximum (burst) rate at which the source produces data, measured in kilobits per second.

Mean burst length: the average size in kilobits of a burst sent at peak bandwidth.

If no QoS parameters are specified for a connection, the network assumes that the connection carries available bit rate traffic, i.e., no quality of service guarantees are provided. Such connections will always be admitted to the network. In particular, our testbed implements UDP on available bit rate connections.

Unless otherwise noted, the experiments were run with a *selected* QoS close to the networks upper limit: mean is 154000 kbit/s, peak is 154000 kbit/s, burst is 95 kbit.

2 Network Experiments

This section reports the results of the network performance experiments. The three performance metrics, throughput, latency, and jitter are determined for each of the three configurations, AAL-5 on ATM, UDP on ATM, and UDP on ethernet.

2.1 Throughput

Purpose
The goal is to determine the maximum throughput that can be achieved between two hosts for each of the three configurations. Throughput is the number of megabits per second that can be transferred. Further, the purpose is to determine the effect on throughput using hosts of different speeds.

Method
The throughput between two hosts is determined by measuring the round trip time of a message: One host (client) sends a message of a given size to the second host (server). The server sends a message of the same size back to the client. The total amount of data transferred is thus twice the message size. Throughput can be calculated by dividing the total data amount by the round trip time, the time elapsed from the time the client begins to send its message until it has received the server's message. The message size is varied from 1 byte to 200 kbytes in steps of 200 bytes. The entire measurement series is repeated 10 times to produce an average throughput. Note that by using this method the throughput includes the overhead of two times latency. For small packets this latency contributes significantly to the troughput, and the actual achievable throughput will therefore be somewhat higher than our measured throughput. For large messages the transfer time dominates, and the latency is insignificant.

The computed average throughput is plotted against the message size. The maximum throughput is achieved at the resulting curve's extremum. Finally, note that the UDP receiver buffers have been increased from the systems default to 250KBytes using the `setsockopt` system call. This prevents loss of UDP packets during the tests. The round trip program is sketched in Figure 4.

The influence of hosts is resolved by performing the experiments between pairs of identical machines, i.e., SS20-SS20 and U1-U1.

Expectations
We expect to see the 10 Mbit/s ethernet fully utilized, but it should be clearly outperformed by the 155 Mbit/s ATM-net. We expect higher throughput from the faster machines (U1) as protocol processing should be done faster on these machines. Finally, the throughput of AAL-5 should be higher than UDP on ATM, because AAL-5 is ATM-native.

```
Client

for(number_iterations) {
   for(packsz=1 to 200KB step 200){
      start=gethrtime();
      do_send (packsz);
      do_recv (packsz);
      stop=gethrtime();
   }
}
do_send (packsz) {//packetize message
   number_pks = packsz div MTU;
   leftover= packsz - number_pks*MTU;
   for(number_pks) {
      send(MTU bytes);
   }
   send(leftover bytes);
}
do_recv(packsz) {//like do_send }

Server

for(no_iterations) {
   for(packsz=1 to 200KB step 200){
      do_recv (packsz);
      do_send (packsz);
   }
}
```

Figure 4: Code sketch for round trip measurements

Results

In Figure 5 we have plotted throughput versus packet size for the UDP-ethernet (U1-U1), UDP-ATM (U1-U1), and UDP-ATM (SS20-SS20) configurations. The throughput is low at first, but increases with larger messages. There are two reasons for this. First, the cost of communicating small messages can easily be dominated by latency (i.e., protocol and network controller setup time). Second, larger messages are much better able to utilize pipelining, e.g, the copying of the next message fragment (message transfer unit) to the controller, can be overlapped with the controllers fragmentation into ATM cells of the previous, which again can be overlapped with the sending of cells from a third message unit.

UDP on ethernet reaches a saturation point very quickly and reaches a throughput of 9 Mbit/s. The curve for the SS20 hosts are identical, and is therefore omitted from Figure 5. This is close to the theoretical limit of 10Mbit/s, and both host types seem to be able to fully utilize the ethernet.

The SS20 hosts achieve 57 Mbit/s for message sizes of 200 kbytes. As expected the faster U1 hosts achieve a higher thoughput, 117 Mbit/s with 200 kbytes messages. The theoretical bandwidth of an 155 Mbit/s ATM network that can be used for user data is 134 Mbit/s (5 bytes of an 53 byte cell is used for cell header, leaving 48 bytes for user data). Thus UDP on fast hosts is approaching the theoretical limit.

Figure 6 shows the throughput of AAL5 versus message size for the U1 and

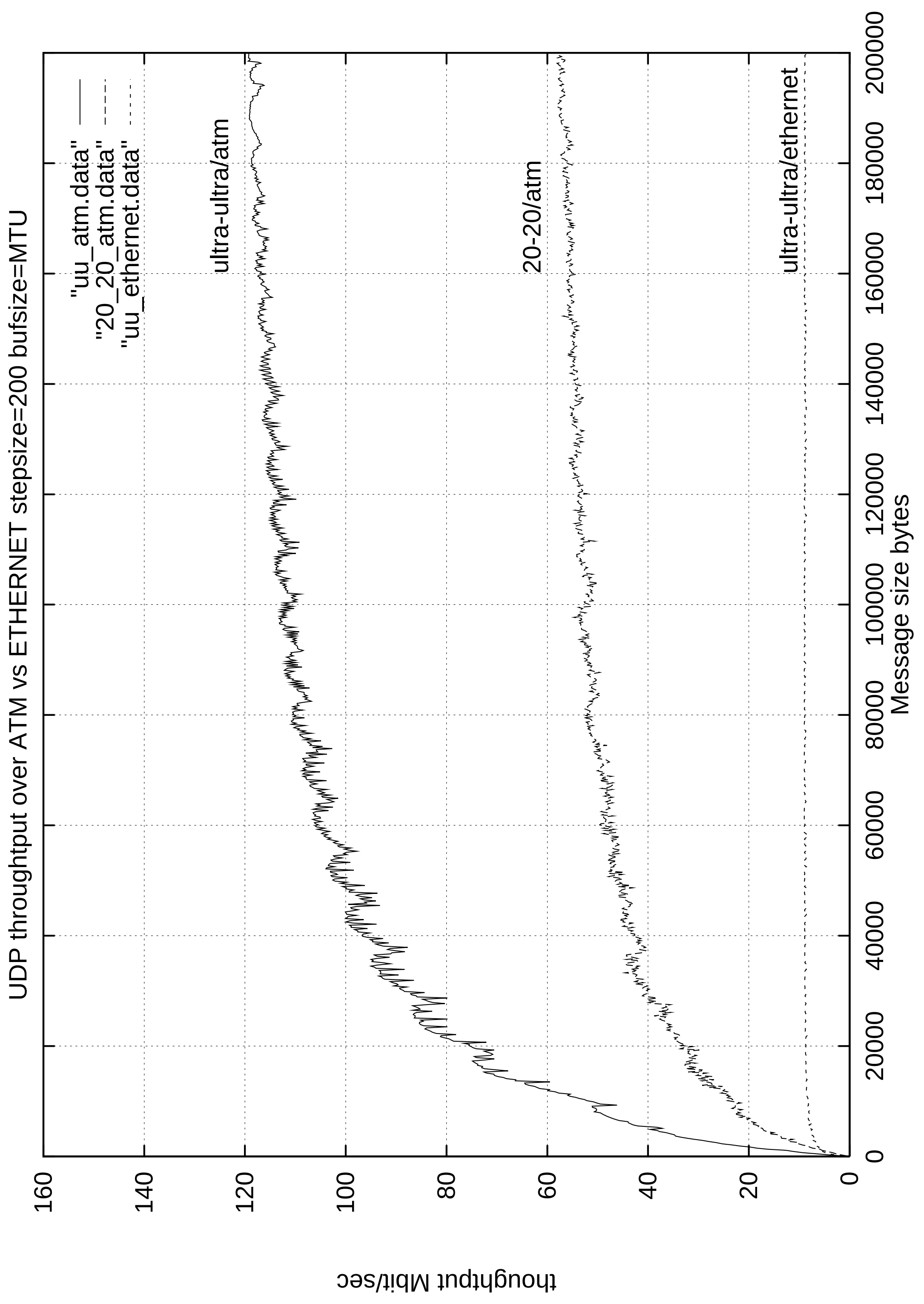

Figure 5: UDP performance on ATM (U1-U1), ATM (SS20-SS20) and ethernet (U1-U1). SS20 on ethernet gives identical performance as the U1 hosts.

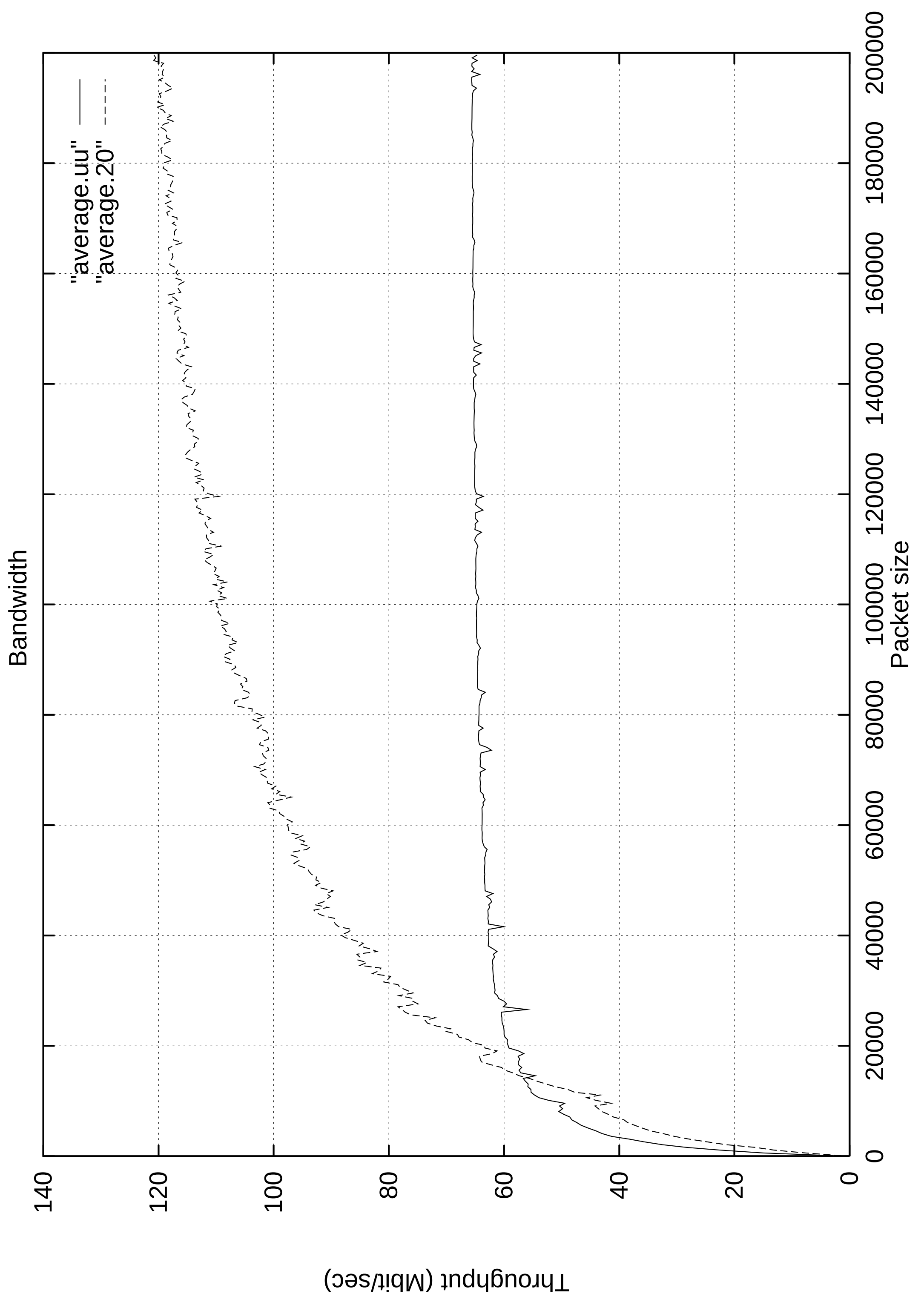

Figure 6: AAL-5 performance on SS20-SS20 and U1-U1

SS20 hosts. Similar to the UDP measurements, the throughput is low at first, but increases with message size. However, the figure shows a very unexpected result. The slower SS20 hosts outperform the faster U1's by far! The SS20 hosts obtain nearly 120 Mbit/s whereas the U1 hosts obtain 65 Mbit/s. Moreover, the U1 hosts' throughput with the ATM-native AAL-5 is only half of what they achieve using UDP. Observe however, that the U1 hosts get a higher throughput for messages up to 15 kbytes, but then flattens abruptly, compared to the other measurements. Our latency experiments confirm that U1's indeed are faster for small messages, see Section 2.2. We do not know the cause of the U1's poor performance, and we have investigated probable causes: We have examined a modified testbed where the U1 hosts has been replaced by two processor Sun Sparc Ultra 2's with 256 Mbytes of ram. However, the same low AAL-5 performance persisted in this configuration. Also, we tried sending from a SS20 to a U1. Here the 120 Mbit/s could be reached. Thus, the problem occur when a U1 host is used as a sender. It could seem like some sort of flow control mechanism kicks in, but we are also investigating differences in adaptors and software, operating system patches, differences in buffer space allocation.

A general comment about performance of the ATM network is that it seem to require the transmission of very large chunks of data to utilize the theoretical bandwidth on a single connection. From our experiment we cannot conclude whether the large data chunk must be sent as few large packets or if it suffices to send a large number of small packets in rapid succession.

Finally, we have observed that the network can drop packets when very large messages (10 Mbytes and over) are sent at the fastest possible rate. This indicates that it may not be possible to maintain the high throughput over longer periods of time. We conclude that further experiments are needed to uncover this problem.

2.2 Latency

Purpose
We like to determine the communication latency between two hosts. The latency is the minimum time it takes to send a message from one host to another. The effect on latency by using hosts of different speeds should be examined.

Method
The latency is decided by measuring the round trip time (see Section 2.1) of a very small message (1 byte). Assuming symmetry in the communication delay between the two hosts, the latency is calculated as half the round trip time. The round trip time is measured 100 times to produce an average.

Expectations
We expect to see lower latency on the ATM network because of its order-of-magnitude higher theoretical bandwidth. We also expect latency to be lower

on the faster machines (U1) because protocol processing can be done faster. Finally, because AAL-5 is ATM-native, we expect it to be a little faster compared with UDP on ATM.

Results

The results are tabulated in Figure 7. The lowest latency (201 μs) is achieved by U1-hosts communication via AAL-5. This is considerably faster than using UDP on ATM between the same hosts, 361 μs. A partly explanation is that a 1 byte AAL-5 message can be transfered in a single ATM cell, whereas a UDP message may require several cells due to the extra header information created by IP and UDP encapsulation. However, we find it quite surprising that UDP on ethernet is only slightly slower (408μs). This indicates that protocol overhead and controller setup time is the limiting factor.

On the SS20 hosts all average latencies have about doubled, i.e., the SS20's are only half as fast on latency compared to the U1's. AAL-5 is still the fastest protocol with 491μs. A new surprise is that UDP on ethernet (846μs) appears a little faster than UDP on ATM (889μs). We do not have any apparent explanation for this. Note however, that the difference is only 43 μs, so this result should be interpreted cautiously.

	Latency (μs)					
	SS20-SS20			U1-U1		
	AAL-5	UDP$_{atm}$	UDP$_{eth}$	AAL-5	UDP$_{atm}$	UDP$_{eth}$
average	491	889	846	201	361	408
minimum	408	835	675	186	230	353
maximum	956	1046	4375	323	669	494

Figure 7: Network Latency

2.3 Jitter

In a perfect network for real-time communication the transfer time of packets of equal size would be constant. This would make it easy to predict communication delays, and to plan activities accordingly. However, real networks are imperfect. Buffering at hosts and buffering at intermediate (shared) switched and HUBS introduce a variable, and often unpredictable, delay. Jitter, to be defined below, is a measure of the discrepancy between a real-network and an ideal network.

Purpose

The purpose is to determine the amount of jitter introduced by the network, and the effect on jitter by network load.

Method

Due to the lack of accurate global clocks in a distributed system, measuring the jitter requires some care due Let t_s be the global time of the sending of the message, t_r the global time of its reception, and d the (constant) transmission delay. The jitter ϵ can then be calculated as the difference between expected arrival time and actual arrival time:

$$\epsilon = t_r - t_s - d \tag{1}$$

A frequency distribution of the jitter can be obtained by performing several measurements. Under the (tentative) assumption that the delay variation is caused by random error, the frequency distribution should be approximately normally distributed around 0.

A direct measurement of jitter according to the above definition cannot be done in a distributed system because it requires accurate global timing. Instead, we use the following method to obtain an estimate of the jitter, see Figure 8. The sender sends messages with a fixed period P to the receiver. The receiver then marks the time of reception of that message. Let t_i be the reception of the message i. Now, if we could rely on the first message getting transmitted perfectly (i.e., without jitter) it would be easy to calculate the jitter of message j ϵ_j as:

$$\epsilon_j = t_j - t_1 - P(j-1) \quad ,j > 1 \tag{2}$$

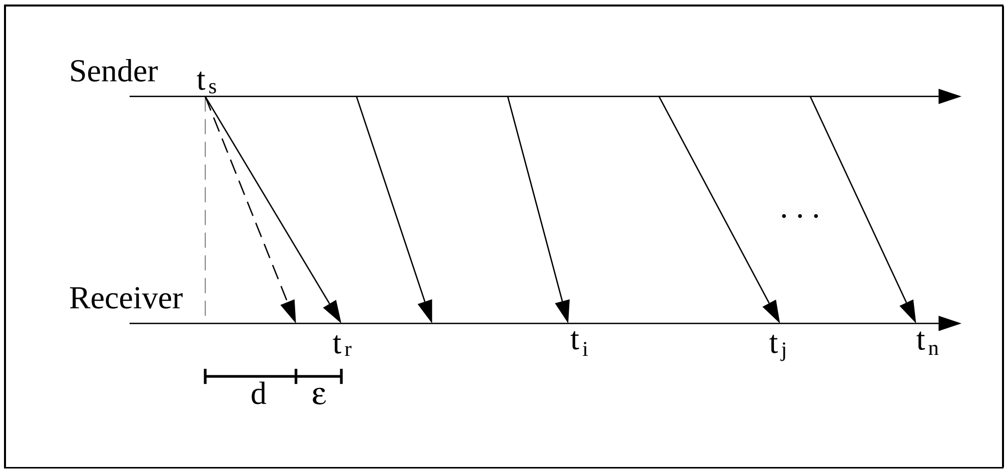

Figure 8: Jitter

This assumption is obviously invalid, and jitter on the first message would cause the subsequent, possibly ideally, received messages to appear being delayed variably. Instead, we use a statistical trick to obtain an estimate $\mathcal{E}_i$ of the jitter of message i. Instead of using only the first message, all messages are used as zero-points[3]. The zero-points are then used as a basis for calculating an average jitter for each message. The assumption of random error on the transfer delay makes this estimate reliable.

The jitter $\mathcal{E}_{ji}$ of message j with message i as zero-point (thus assumed to be received ideally, with constant delay) can be calculated as:

$$\mathcal{E}_{ji} = \begin{cases} t_j - P(j-i) - t_i, & j > i \\ t_j - t_i + (i-j)P, & j < i \end{cases} \tag{3}$$

Consequently:

$$\mathcal{E}_j = \frac{\sum_{1 \leq i \leq n} \mathcal{E}_{ji}}{n-1} \tag{4}$$

An alternative method would be to directly measure the jitter of a message round trip. The disadvantage of this is however, that the round trip jitter contains the sum of two network jitters. If jitter is assumed to be caused by random error, the probability of getting two consecutive extreme delays would be half, and hence, the resulting probability distribution would be 'slimmer' than our used one-way method.

Jitter is measured by sending messages with a period of 5 ms between two SS20's. The code is ourlined in Figure 10. The message size is 8192 bytes on the ATM-network and 1 kbytes on ethernet to compensate for Ethernets lower bandwidth. The measurement processes were scheduled in real-time mode, and the sender process busy waited to get more accurate timing than the scheduler can produce. The influence of a loaded network is determined by letting two processes, scheduled in timeshare mode, send available bit rate traffic at their maximum speed. An added traffic load can be either cross traffic (U1-U1), or parallel (SS20-SS20), and both experiments are conducted, see Figure 9. The last experiment is meaningful because the SS20 machines are equipped with 2 processors each.

[3] A sufficiently large *random* subset would suffice.

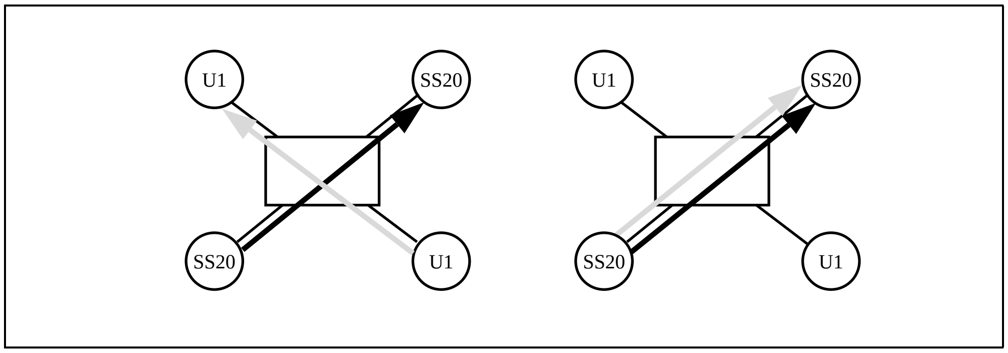

Figure 9: Two experiments with traffic load. Left: cross load. Right: parallel load. Black arrows denote thea jitter sensitive traffic. Gray arrows denote the added background load.

```
Sender                                      Receiver

for(number_packs) {                         for(no_packets) {
   start=gethrtime();                          receive(packsize)
   send(packsz);                               stop=gethrtime();
   stop=gethrtime();                           //stop=reception time of packet i
   //verify stop-start is insignificant     }
   while(gethrtime()<nextPeriod);
   //busy wait until next period
}
```

Figure 10: Code sketch for jitter measurements

Expectations
We expect to see lower jitter values on the ATM-network because ATM is designed to be able to carry time sensitive traffic, and it is therefore likely that the implementors have considered issues like how long time a transfer unit (cell) can be stored at a switch. Ethernet is based on an older technology, primarily intended to carry time insensitive data traffic. We also expect that UDP on ATM has slightly more jitter than AAL-5, primarily because AAL-5 is ATM-native. In general, adding load on the network should cause more jitter, because network resources (buffers, and switches) are now shared, and thus less likely to be available at a given time. However, we in general expect to see small jitter values because our testbed is a local area network with only one HUB/switch. Measurements on a wide area network will likely give different results.

Results
The frequency distribution for the 3 configurations without additional load is shown in the histograms in figures 12, 13, and 14. All three histograms are very

similar, and are centered around 0.

Most jitter values lie within +/- 200μs., which are so small numbers that scheduling disturbance can have a visible influence. UDP seems to have a small top at the left of its center. Hence, without load all three protocols produce insignificant jitter for video communication.

A few very extreme values have been omitted from the figure to produce a clear graph. These values are tabulated in Figure 11.

Configuration	Jitter (μs)	count
AAL-5-noload	1492	2
UDP-ATM-noload	3039	1
UDP-ATM-load	6570	1
UDP-ETH-noload	10143	1

Figure 11: Unplotted Jitter Values

Adding a cross load at the HUB/switch have no visible impact at all, so no graphs of this are shown. The missing effect can be explained by the capacities of the switch/HUB: The ATM switch has a switching capacity of 2 Gbit/s, which is an order-of-magnitude more than the load which can be produced by the two U1 hosts. Similarly the ethernet HUB is designed to feed 24 hosts with 10 Mbit/s each. More machines must be used to produce a significant load.

Figures 15, 16, and 17 show the effect of adding a parallel load. The jitter values increase significantly and they are spread out more. Most jitter values on AAL-5 and UDP on ATM lie between +/- 500 μs but many values are beyond these limits. UDP jitter on ATM appear more focused than AAL-5. We do not know whether the extra delay variation is generated by the switch or (more plausible) at the sending/receiving queues.

The jitter on a loaded ethernet is affected extremely. Most jitter values lye between +- 60 ms (*mili-seconds*), some even beyond. This amount is, as will be discussed later, very significant for video communication and affects implementation strategies.

We have identified a probable cause of this amount of jitter. The plot in Figure 18 of the actual receive times of each sent message reveal that the receiver makes sudden jumps in receiving time. An ideal plot should be a straight line, like the one between message 235 and 245. A possible explanation can be found in differences in the queuing strategies used by the two network types in hosts and/or HUBS. Our ATM network uses per virtual connection queuing, i.e., cells belonging to one connection is queued up in its own private queue, see Figure 19a. Our hypothesis is that the UDP on the Ethernet configuration uses a common FIFO queue for storing packets to all receivers, see Figure 19b. The consequence of FIFO queuing at the sender side is that the multiplexing is done without consideration to the fact that one process is sending time sen-

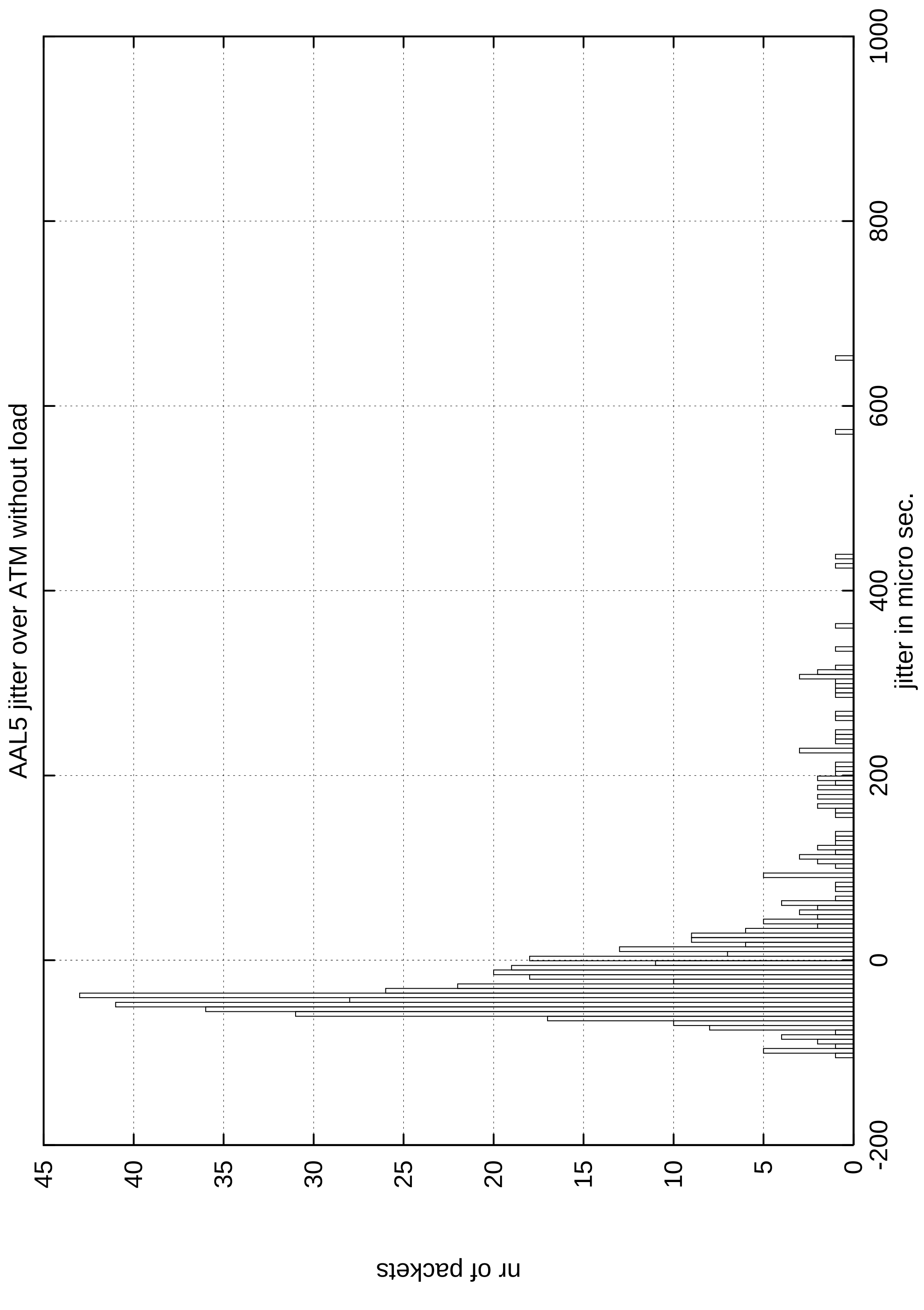

Figure 12: Histogram for jitter on AAL-5 without extra load.

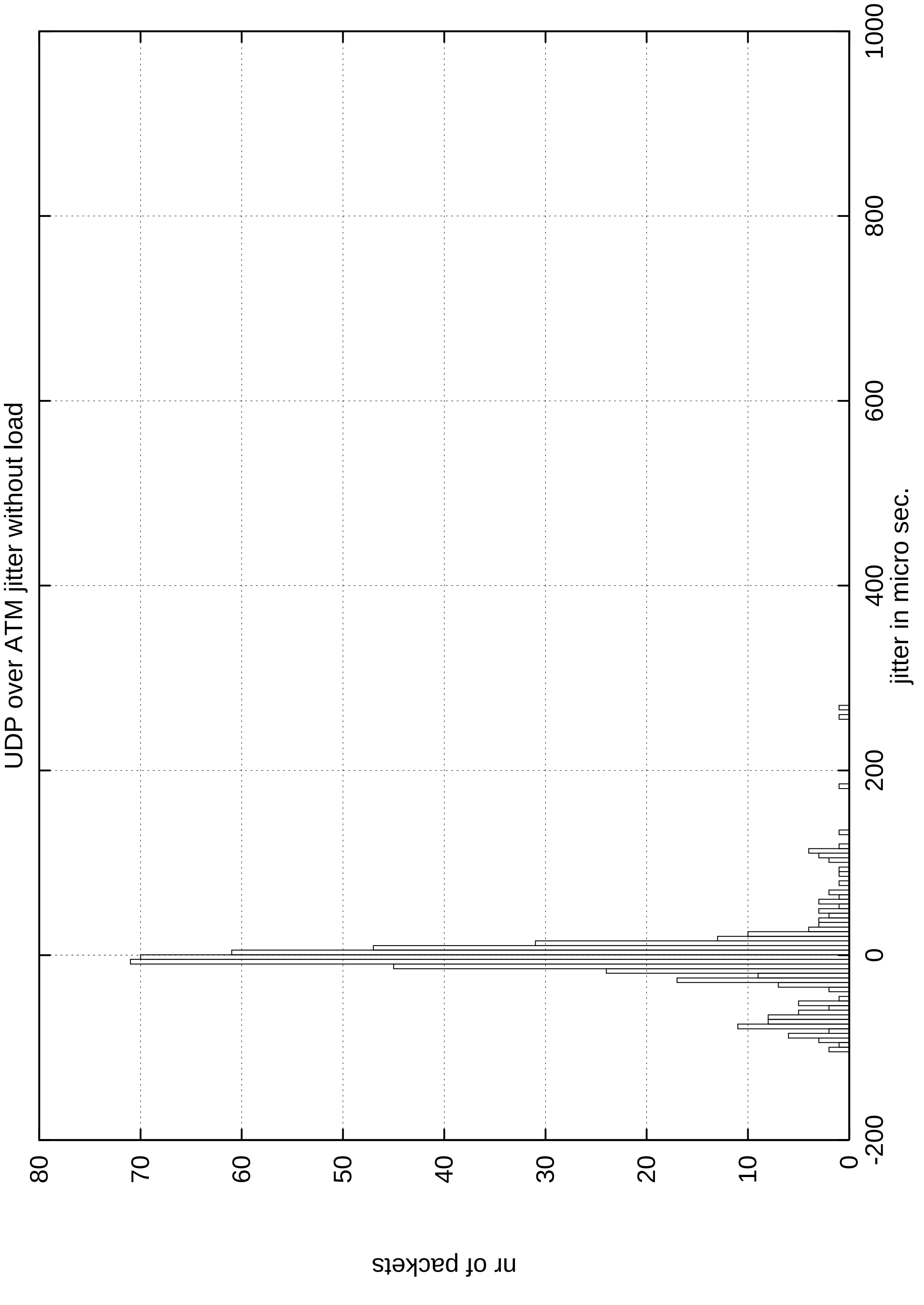

Figure 13: Histogram for jitter on UDP on ATM without extra load.

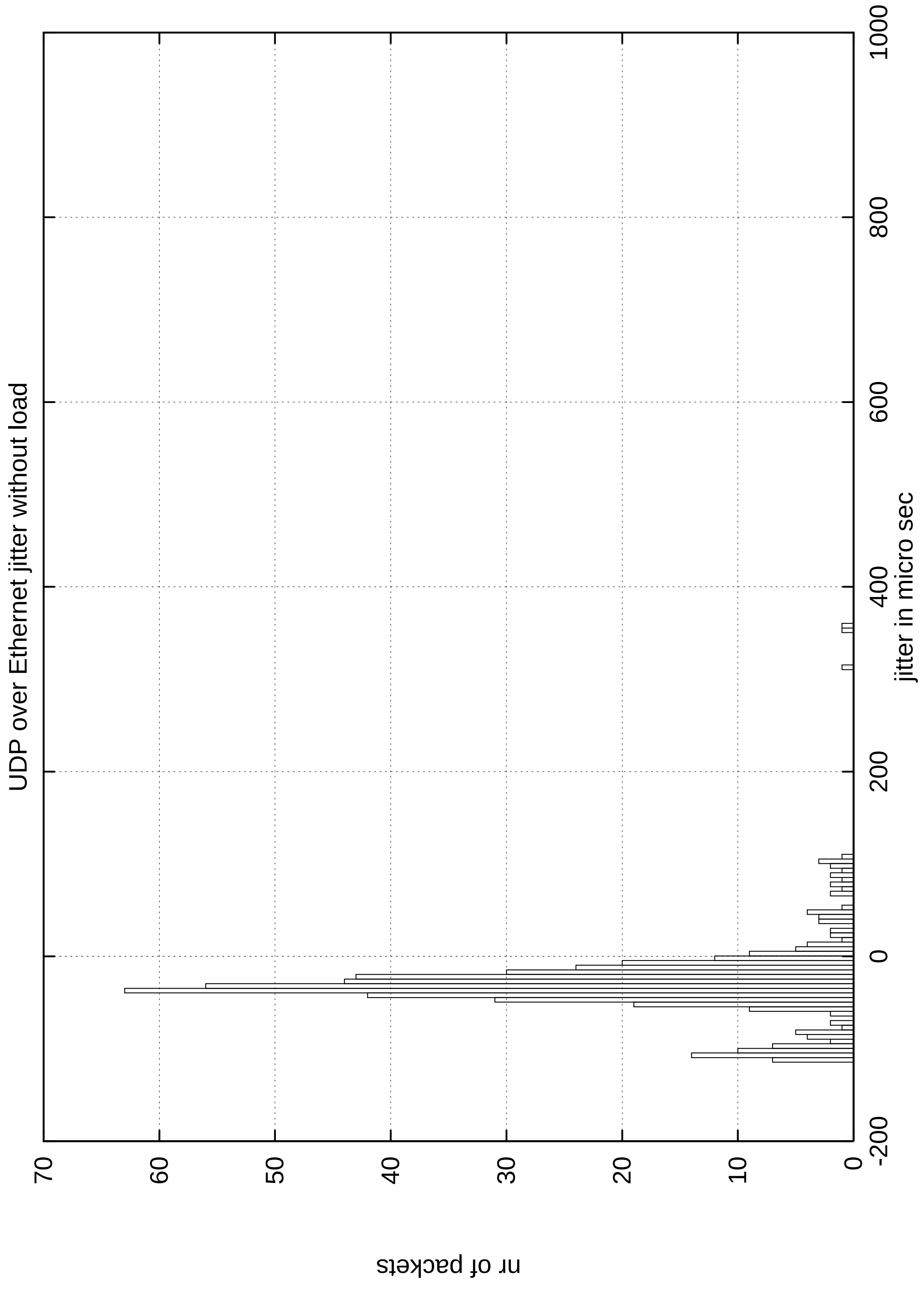

Figure 14: Histogram for jitter on UDP on ethernet without extra load.

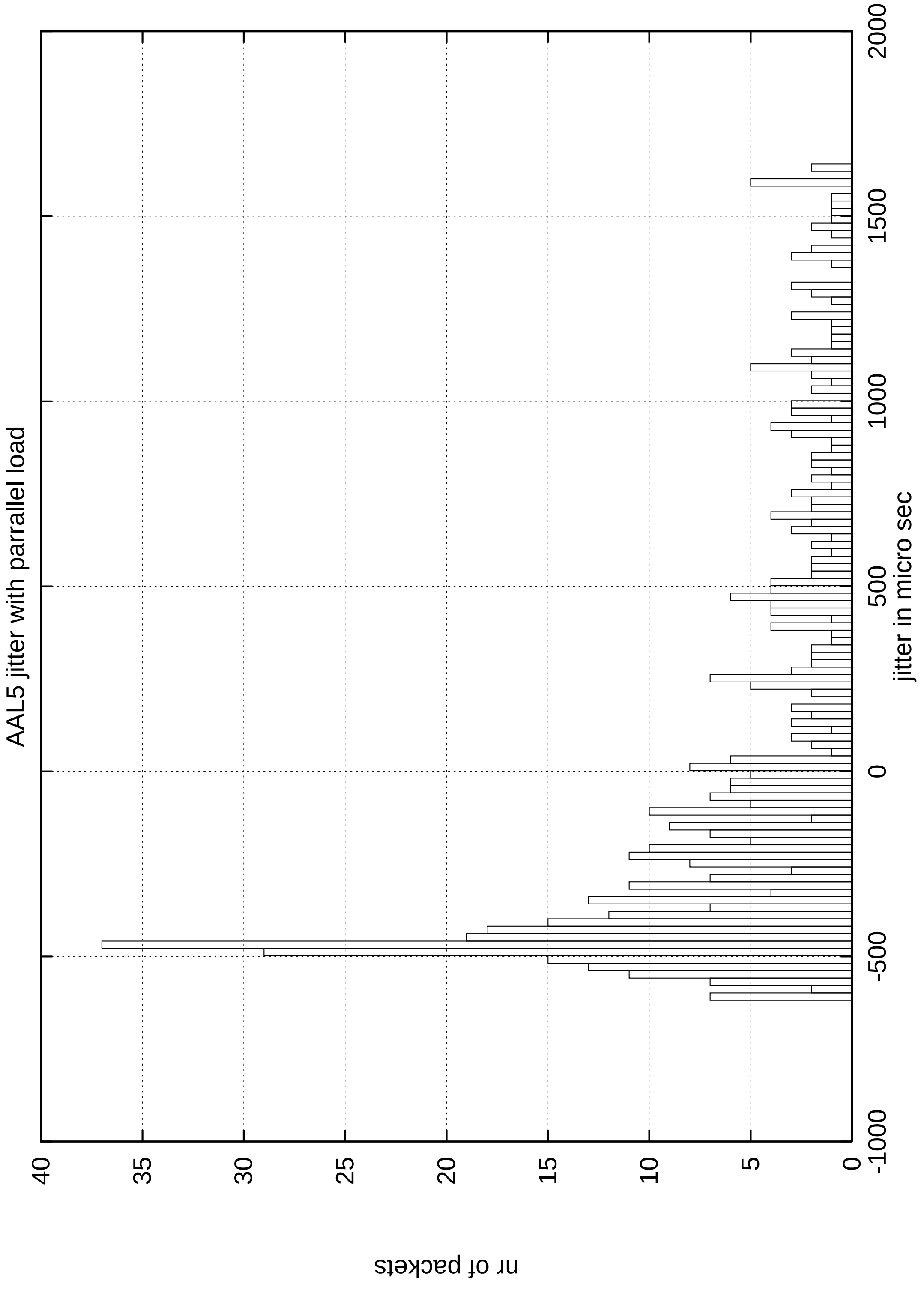

Figure 15: Histogram for jitter on AAL-5 with parallel load.

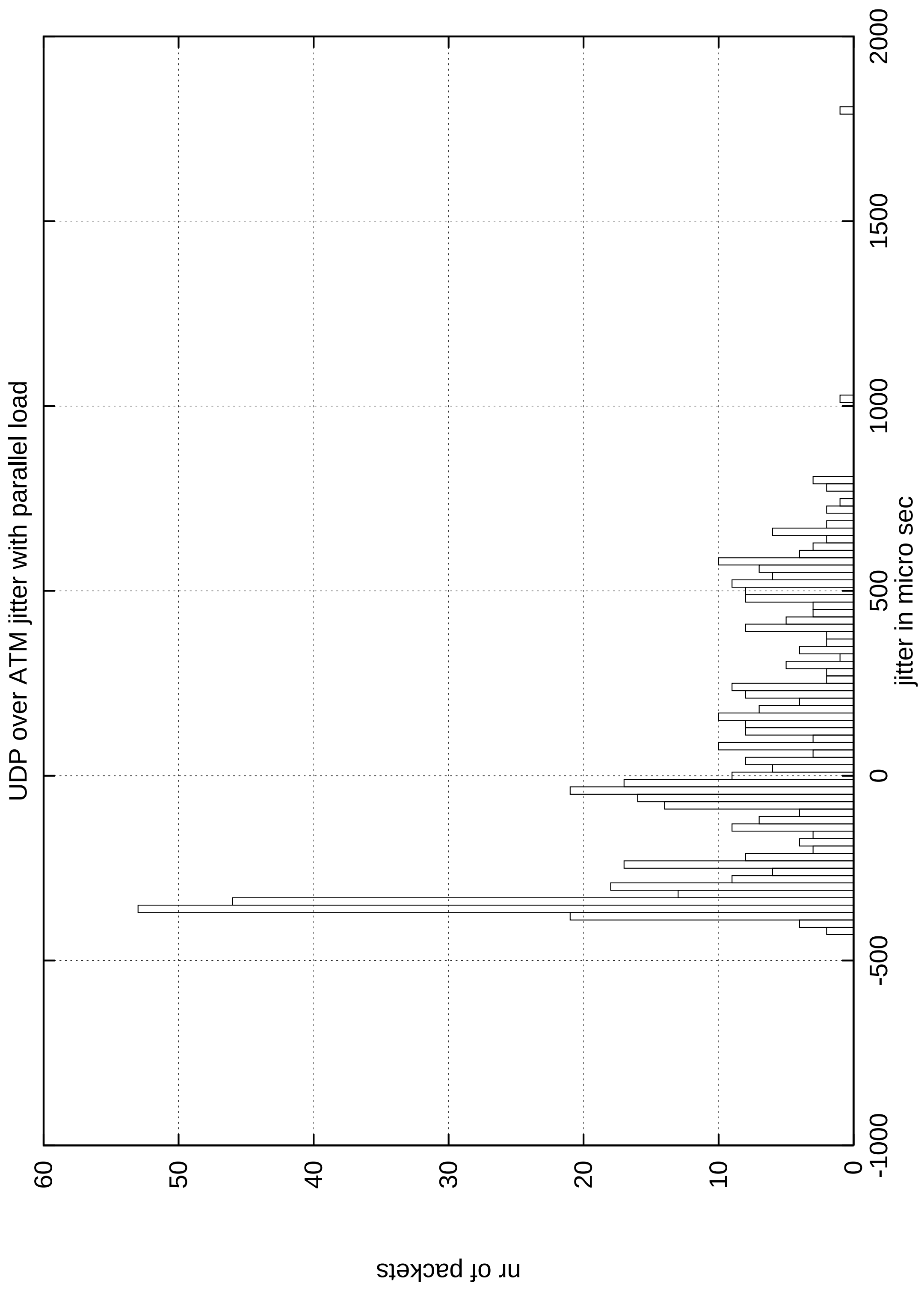

Figure 16: Histogram for jitter on UDP on ATM with parallel load.

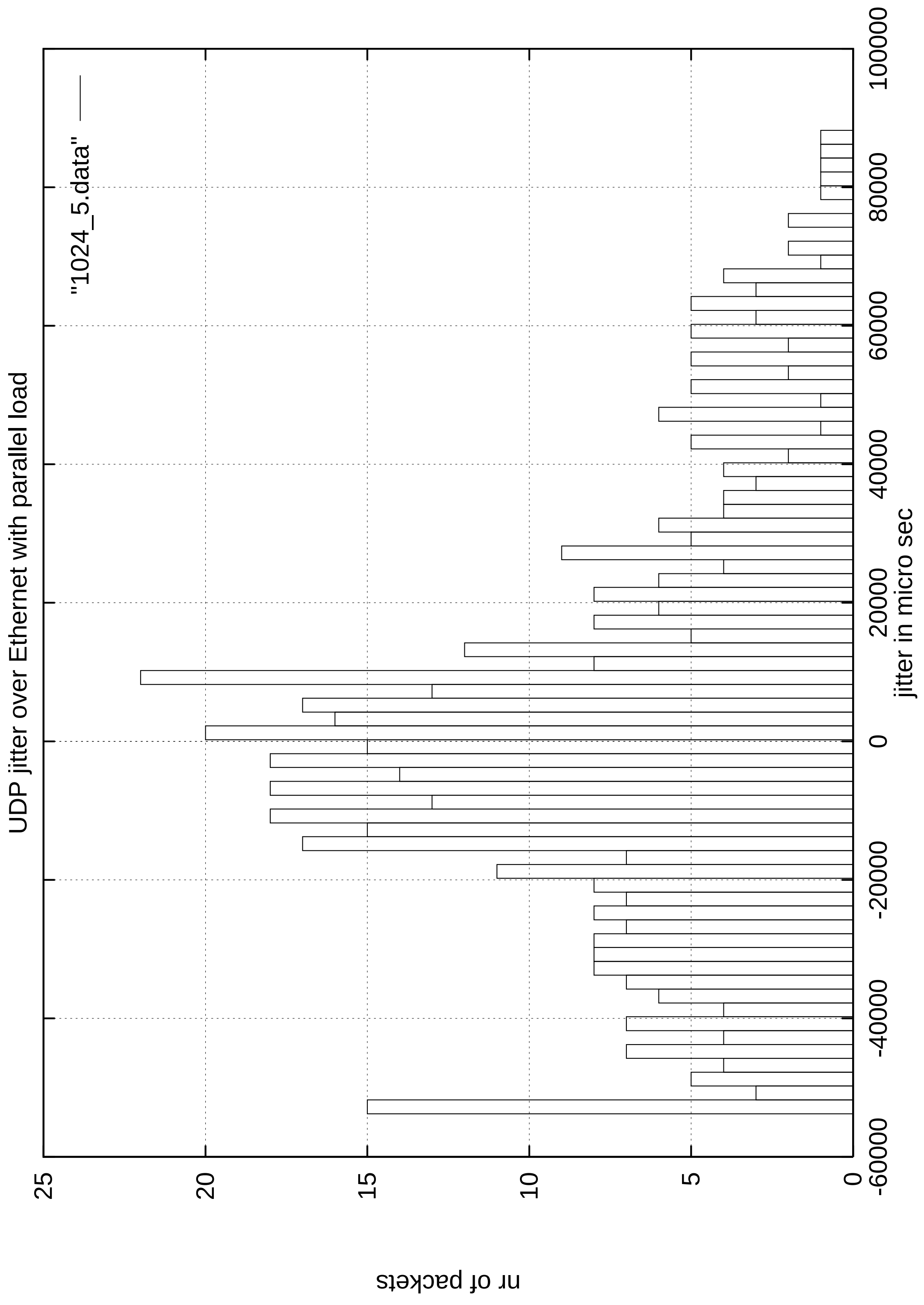

Figure 17: Histogram for jitter on UDP on ethernet with parallel load.

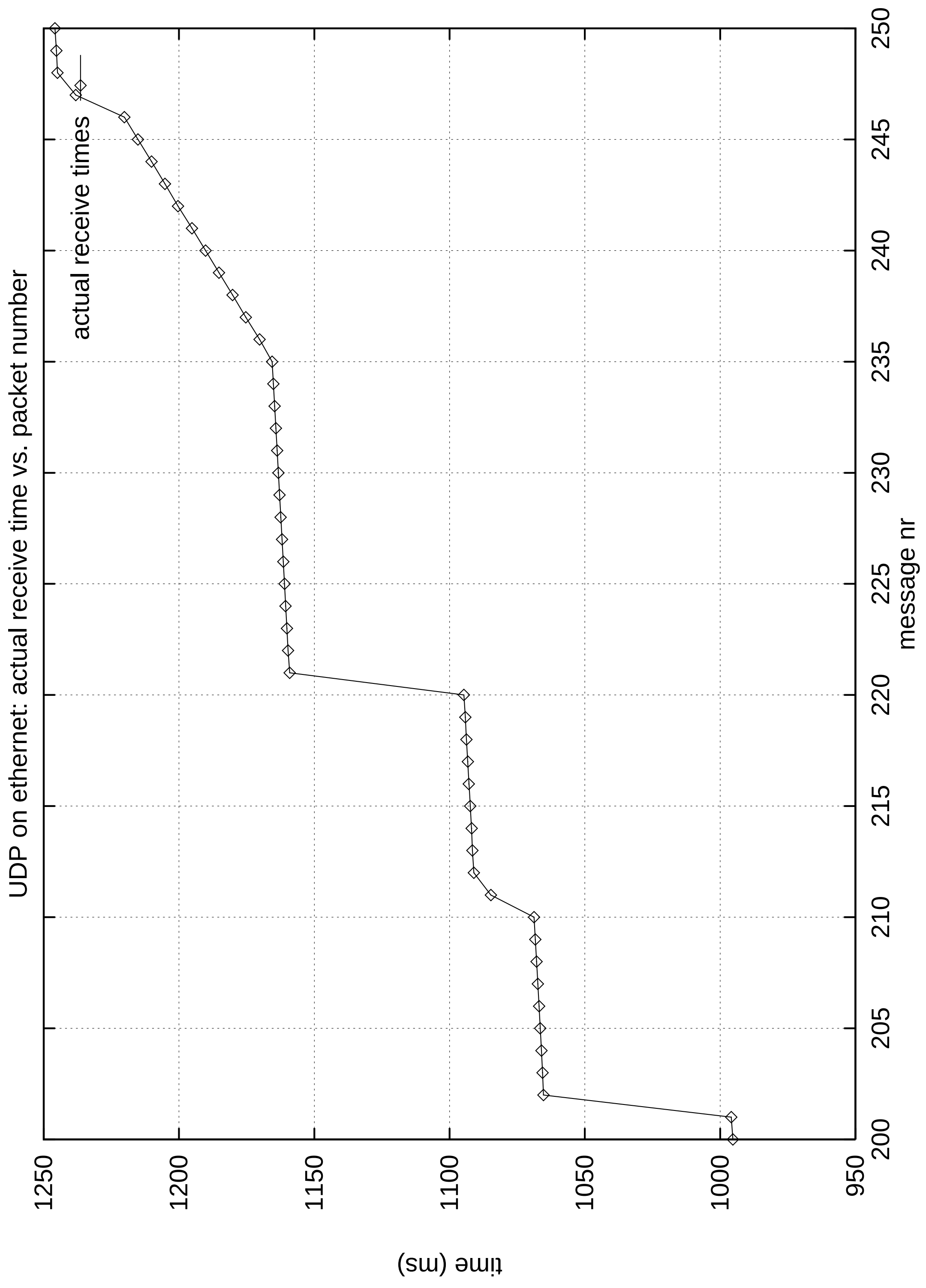

Figure 18: Actual receiving time for each message confirms the large amount of jitter found on a loaded ethernet

sitive traffic. This traffic is delayed unpredictably by the non-realtime traffic produced by other senders. A similar problem occur at the receiver side: The real-time traffic is sent upstream and demultiplexed and in arrival order, not in “priority” order.

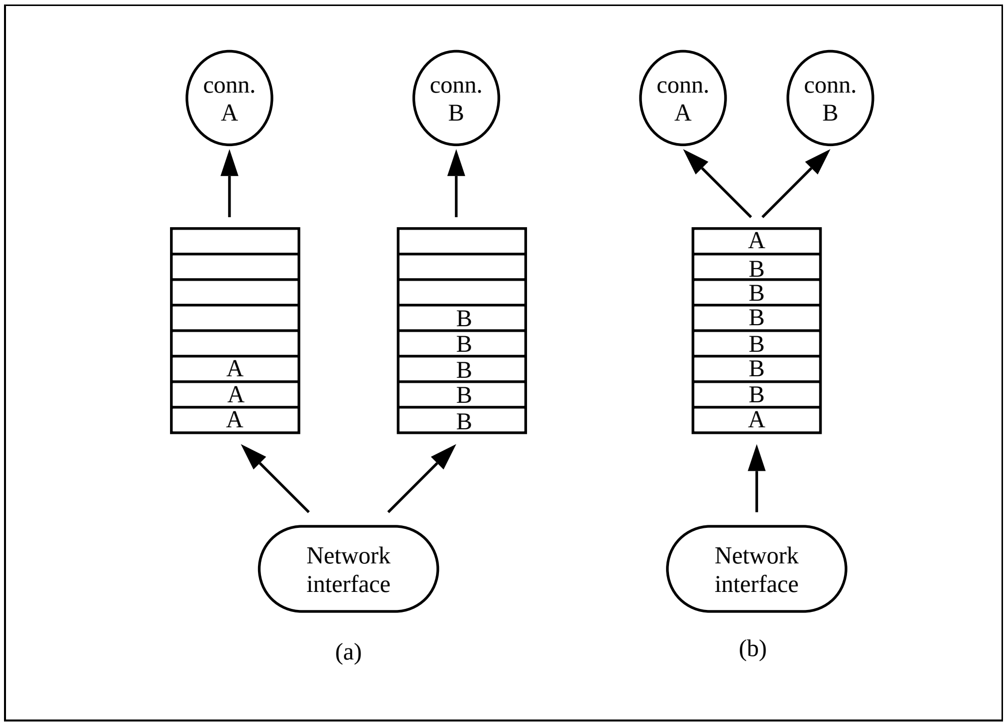

Figure 19: (a) Per connection (de) multiplexing. (b) FIFO (de)multiplexing.

Another problem that may occur at the receiver side is *unbounded priority inversion*[19]. Suppose a receiving process P_h scheduled with high priority blocks because a lower priority process P_l has to consume messages from the queue head. Now a third process P_m with medium priority wakes up and due to its higher priority it preempts the low priority process. As long P_m has work to do it prevents P_l (and thereby P_h) from running by an unbounded amount of time. It should be noted that the Solaris OS supports real-time scheduling and implements a priority inheritance mechanism to avoid unbounded priority inversion due to blocking on locks to kernel data structures, but does not support real-time i/o[14]. However, the implementation of the ATM interface seem more adequate at supporting real-time traffic than the Ethernet interface.

A further source of variation in communication delay on an ethernet is collisions, and the following recovery through exponential backoff. However, we meassured the amount on collisions on a host in the testbed and a heavily loaded departmental server to be around 3% of the communicated packets on their interfaces (using the `netstat` tool). This seems to infrequent to blame collisions as a primary delay variation source.

A disadvantage of using UDP on ATM is that UDP is transferred as available bandwidth traffic, and therefore will be experience a penalty on a loaded network. It is currently impossible to provide QoS guarantees for UDP based applications, although work is in progress to allow qos to be provided for IP-based applications[26].

3 Compressed Video Experiments

In this section we analyze compressed video with respect to bandwidth and cpu usage. We compare two compression techniques defined by the Motion Picture Expert Group, MPEG-1 and MPEG-2. MPEG-1 is the older technique aimed at transmission and CD-ROM storage of sub TV-quality signals. MPEG-2 is a newer, optimized and extended version of MPEG-1. It is intended to support transmission various degrees of quality, up to high definition TV[20].

Figure 20: A frame from the test video

To perform our analysis we have recorded a short typical video conferencing test video. In the beginning it shows a person talking to the camera. Later, another person moves into the background from the right, and immediately thereafter, a third person moves in from the left while the second person moves back out. The video has a resolution of 400×320 pixels with 24-bit color depth, and is recorded with 18 frames per second (fps), i.e. a new frame is grabbed/displayed each 56 ms. The video contains a total of 175 frames, giving a playback time of approximately 9.7 sec. Figure 20 shows a single frame of the video. Without compression this video would require a bandwidth of (400*320*24*18 bits/s) approximately 55 Mbit/s. Using compression this can be reduced to about 1 Mbit/s without severe degradation of quality. The test video was then compressed with an using MPEG-1 and MPEG-2. The compressors and decompressors were available in the public domain [2, 8, 9]. The results reported here depends on the performance of these tools, and may not be exactly identical with other implementations; your milage may vary. We expect the general trend to be clear, though.

3.1 MPEG Background

We here summarize certain background information necessary to get a full understanding of the following analysis. Motion pictures can be compressed in two ways: First, each frame can be compressed by reducing the amount of spatial color redundancy information in a frame, e.g., areas with the same color. A second important observation is that a movie rarely consists of a sequence of unrelated pictures, but more often consists of the same objects moved in some direction. The leads to reduction of temporal redundancy, i.e., only store differences between frames.

In MPEG a frame is divided into areas called macro blocks of 8×8 and 16×16 pixels. These sizes turn out to suit motion prediction. A frame can be coded in one of three ways[4][20]:

I-frame: An intra-coded frame is a self-contained picture (a still picture). Each macro block is Discrete Cosine Transformed (DCT), and then differences between successive blocks are computed and represented using variable-length encoding. Consequently, areas with little color difference can be compressed a lot, whereas areas with large difference (such as sharp edges) can be less compressed.

P-frame: A predictive-coded frame contains motion prediction information, and its compression uses information from the previous I-frame or all previous P-frames. A prediction error is calculated between macro blocks in the current picture and the past references I or P picture.

B-frame: A bidirectional predictive coded frame contains motion prediction from the previous and next I or P-frame, that is, compression of a B-frame uses information from both a past and a future I/P frame. A prediction error is calculated for a macro block between the two pictures. Two motion vectors are calculated. The first determines the value and direction of the forward prediction referencing the future frame. The second determines the value and direction of the backward prediction, referencing the past frame.

An application decides which order it uses these encoding techniques. The more B-frames it uses, the higher compression rate. But many successive B-frames

[4]MPEG describes in addition to I, P and B frames also a fourth type, D frames, which we will not use here. D frames are used to provide fast-forward facilities.

makes it hard to do random access (or recovery after a dropped frame due to transmission error). The pattern an application uses is called a GOP-pattern (Group of Pictures). Our MPEG-1 test movie uses a recurring "IBBPBB" pattern, and our MPEG-2 uses a recurring "IPPPPPPPPPPP" pattern. These patterns reflect the order inwhich these frames are displayed, the display order. Since B-frames in MPEG-1 references future pictures the order inwhich frames are compressed is different: A "IBBP" pattern is compressed in the order "IPBB".

3.2 Bandwidth Analysis

Purpose
The goal is to determine the bandwidth variation of compressed video, and to point out the differences between MPEG-1 and MPEG-2.

Method
We have constructed a tool that is able to parse MPEG files and extract the size of each frame. The bandwidth required to transmit a frame is the frame size divided by the frequency with which frames are transmitted. e.g., a frame is sent every $\frac{1}{18}$ second. We have applied the parser tool to the test movie.

Expectations
We expect to see that the frame size depends on frame type, and on the amount of movement in the movie. We expect MPEG-2 to produce smaller frames than MPEG-1.

Results
On Figure 22 we have plotted the frame size versus frame number (reflects the order in which frames are recorded) for the MPEG-1 compressed file. The figure shows the variation in frame size over time, and thus implicitly also the variation in bandwidth usage. It can be seen that the frame size varies hugely. The largest frame is 33Kbytes and the smallest is about 1.1 kbytes. The differences can be explained by two main factors, frame type, and the amount of motion in a movie clip.

To see how frame size depends on frame type consider Figure 23 where we have focused on frames 20-40 of Figure 22, and annotated each bar with the frame type. I-frames, storing complete compressed pictures, are the largest, P-frames the second largest, and B-frames the smallest. It can also noted that the movie's GOP pattern "IBBPBB" is clearly distinguishable.

The second main influence on frame size variation is motion. From frame 1 to around frame 80 on Figure 22 one person is sitting calmly talking to the camera. The plot is in this area relatively constant. Between frame 80 to 110 the second moves into the picture and between frame 110 to 140 the the third person moves in, both creating large fluctuations in the frame sizes. In area with movement frames generally becomes larger: P and B frames because they accommodate the extra movement, and I frames because the background now containing extra information, and consequently cannot be compressed as much.

Figure 24 shows the bandwidth profile for the test movie using MPEG-2 compression. First, it can be noted that in B-frames in MPEG-2 have been replaced by P-frames. Only I-frames are peaking up. The GOP pattern is clearly visible: an I frame succeeded by 11 P-frames. Although the B-frames that were the smallest in MPEG-1 have been replaced with larger P-frames, the overall frame size seems to be halved. Both I and P frames are generally half the size of the corresponding frames in MPEG-2. Thus, MPEG-2 appear to compress much better than MPEG-1.

However, a direct comparison is problematic because the visual quality of the decompressed movie has to be judged. The authors' subjective opinion is that the MPEG-2 compressed moved has the best quality of the two. On the other hand trying to compress the MPEG-2 movie additionally introduces a visible quality degradation. We have been unable to adjust the MPEG-1 compression rate with the used compression tool, and were thus unable to determine if the MPEG-1 movie could have been compressed further without quality degradation (or be compressed less with quality gain).

Finally, it should be noted that the effect of movement appear different in MPEG-2. In the movement area the P-frames seem a little larger than usual. This increase is smaller than one would expect by looking at the corresponding change in the MPEG-1 movie. It is surprising however, to see that I-frames actually becomes smaller. Our explanation of this is that the MPEG-2 compressor tries to maintain a fixed output bandwidth by compressing the outlined frames harder than usual (possibly degrading the quality or using extra time for compression). Thus, a MPEG-2 appear much more suitable for transmission on a network, because it is easier predict bandwidth usage, and consequently to determine appropriate quality of service settings.

	Framesize (kbytes)				
	MPEG-1			MPEG-2	
	I-frame	P-frame	B-frame	I-frame	P-frame
average	23.3	13.2	2.6	11.1	2.4
minimum	17.5	8.7	1.1	7.8	1.4
maximum	33.0	22.2	7.4	13.5	3.5
Total	1379.3			547.1	

Figure 21: Bandwidth statistics

Figure 21 summarizes some statistical information about the frame sizes produced my the two compression techniques. This information can be used to calculate an estimate for the QoS parameters used in ATM-transmission, see Section 4.1.

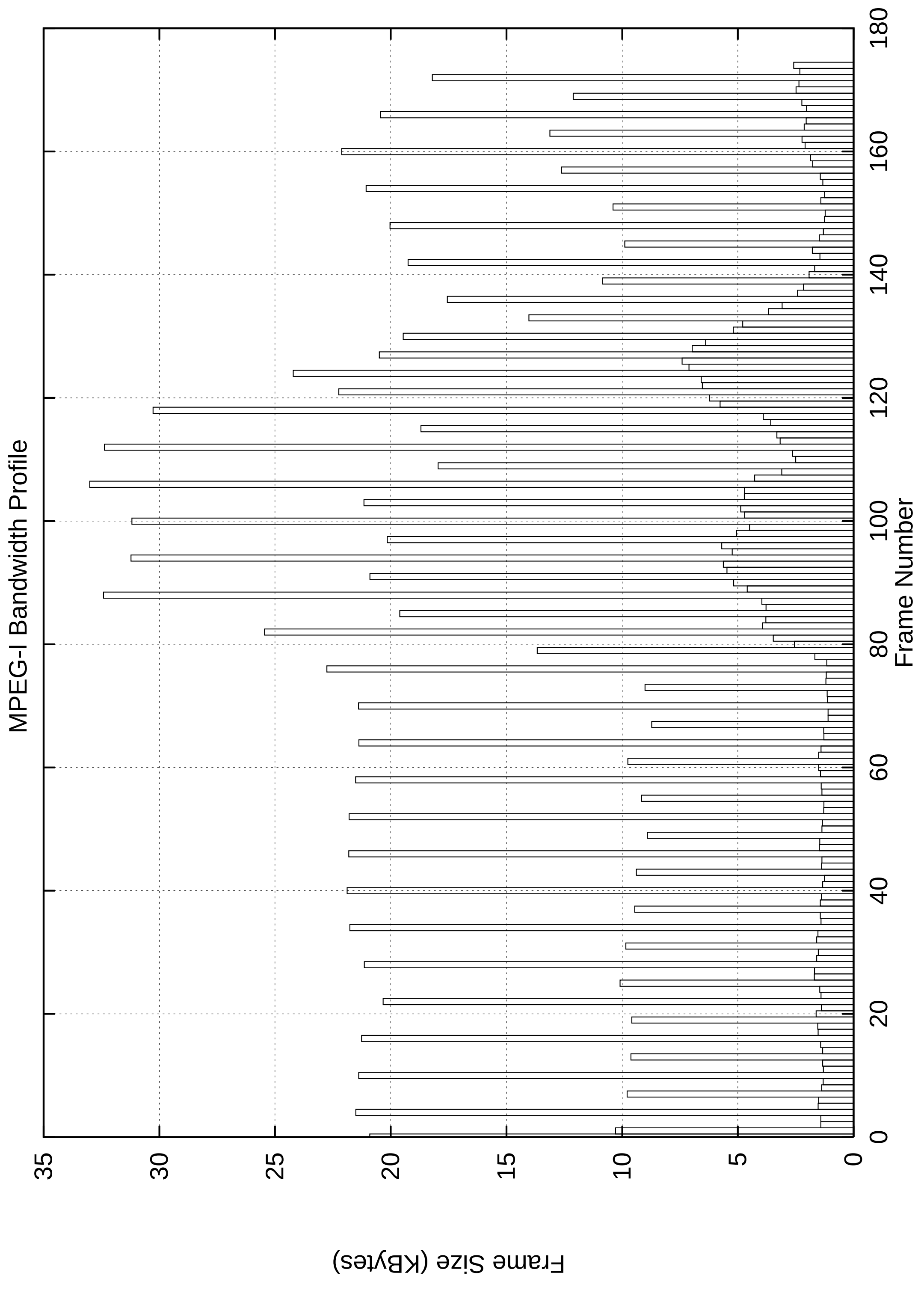

Figure 22: Bandwidth profile for MPEG-1: frame size plotted against frame number (display order).

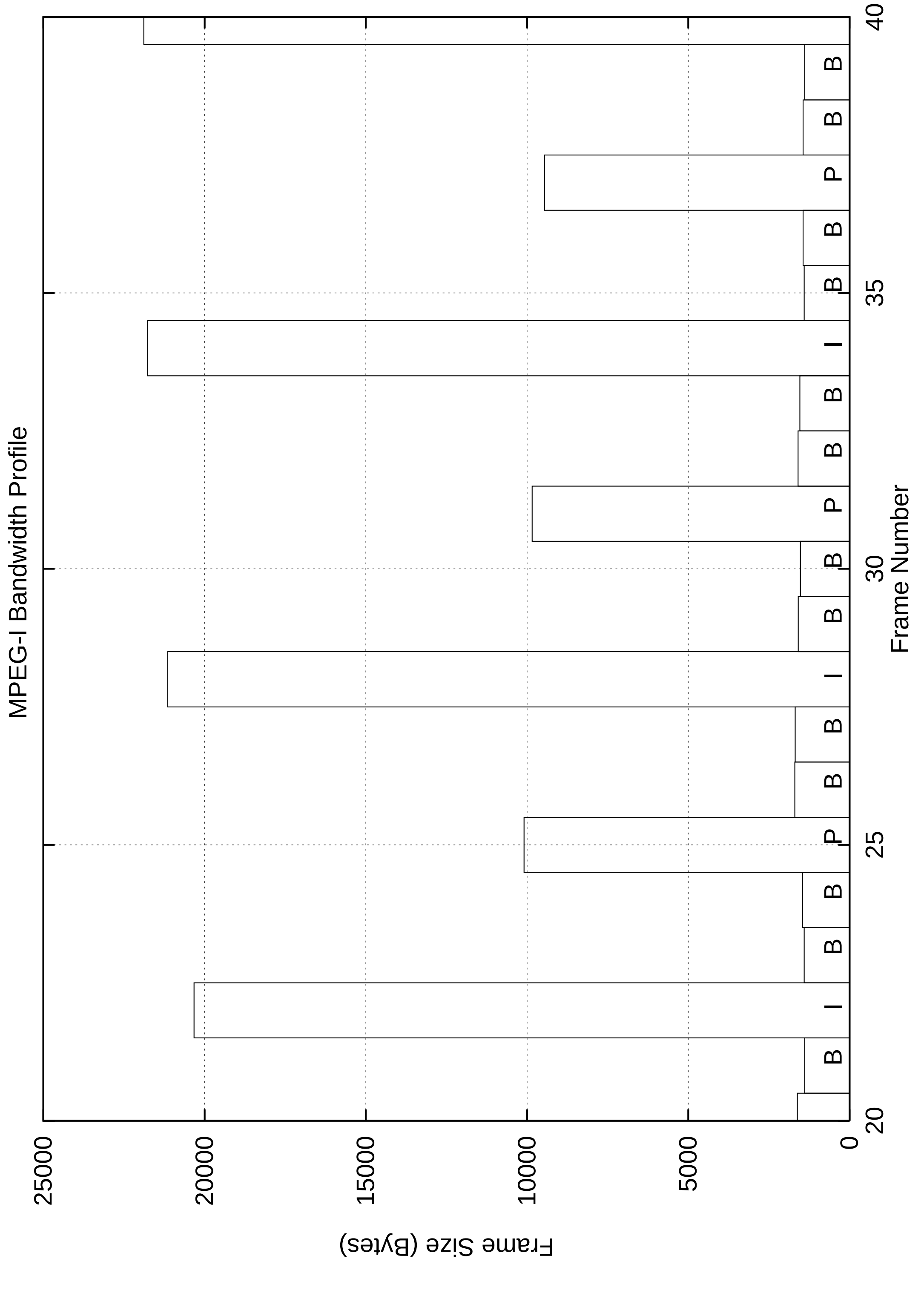

Figure 23: Correlation between frame type and frame size (display order).

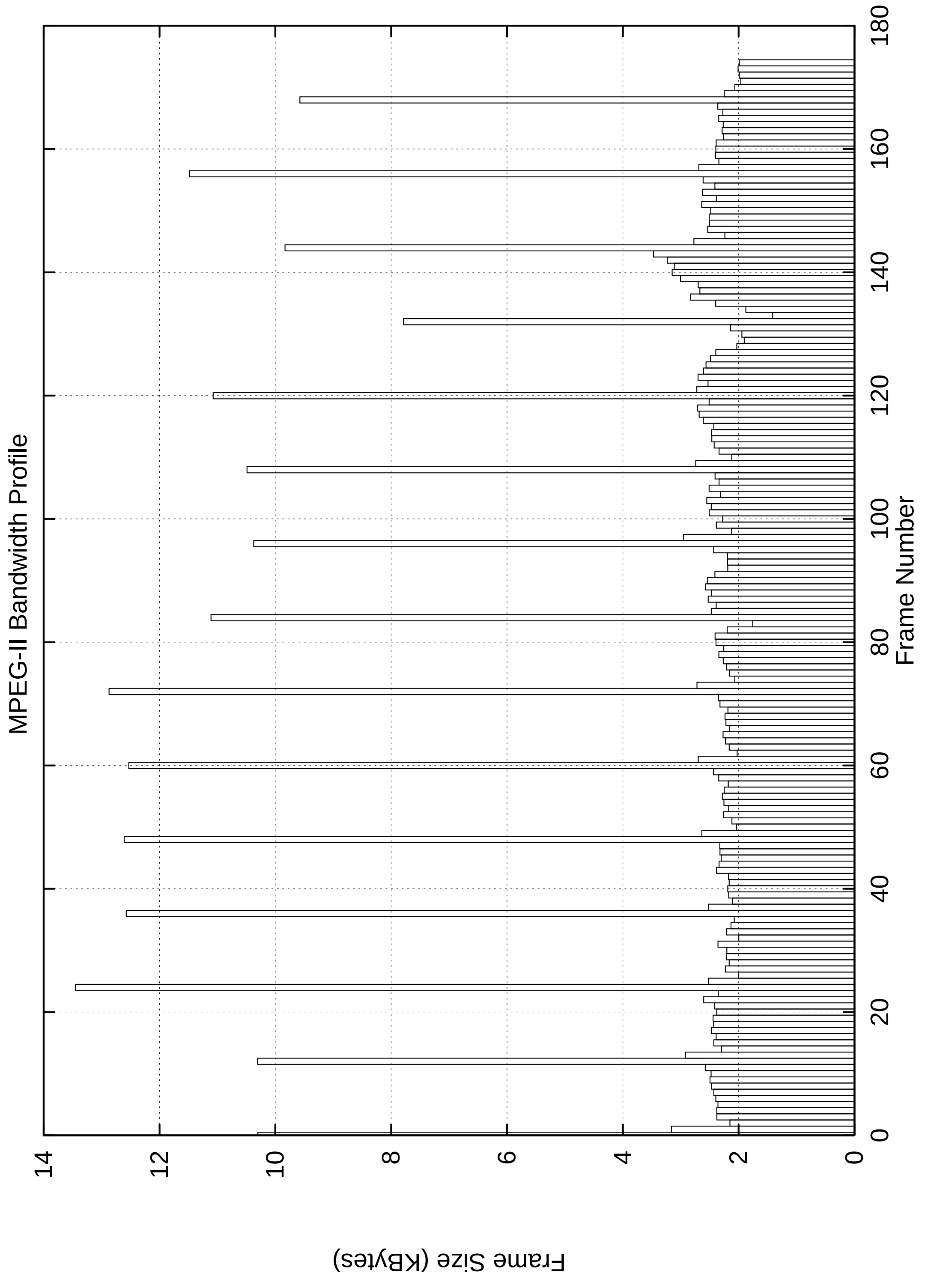

Figure 24: Bandwidth profile for MPEG-2: frame size plotted against frame number (display order).

3.3 CPU-usage: Decompression

Purpose
The purpose is to measure the variation in CPU-usage during decompression of a movie, and compare MPEG-1 and MPEG-2 wrt. CPU-usage.

Method
We have instrumented existing shareware MPEG-1 and MPEG-2 software compressors with code that measures the time spent on decompressing each frame. Our movie was then compressed with these tools on U1 hosts.

Expectations
We should see a dependence on frame type and on the movement in the movie, and that MPEG-2 is faster than MPEG-1.

Results
On Figures 25 and 26 we have plotted the amount of time used to decompress a frame versus frame number for the MPEG-1 and MPEG-2 files respectively. The figure shows the variation in CPU usage frame size over time. In MPEG-1, there is a large variation: the smallest frame took 40.8 ms to decompress while the largest 183.9ms, i.e., exceeding 400%. This difference in job size must be taken into account when scheduling the decompressor: With 18 fps, decompression must begin at least 4 frames earlier. The variation in MPEG-2 is much less, about 10%.

There is an evident relation between the frame type and decompression time: I frames consume the longest time, P frames second longest, and B frames the least. It is interesting to note that B-frames are relatively hard to decompress considering their relative small sizes. Further, the decompression profile appears to follow the bandwidth profile exactly, thus there seem to be a correlation between frame size (for each frame type) and the time used to decode it.

	Decompression time (ms)				
	MPEG-1			MPEG-2	
	I-frame	P-frame	B-frame	I-frame	P-frame
average	103.1	80.7	42.9	80.1	79.9
minimum	81.8	40.8	35.0	78.3	77.0
maximum	183.9	156.3	149.8	82.3	86.7
Total	10414.8			13745.2	

Figure 27: CPU usage statistics: decompression

Figure 27 summarizes the decompression statistic.

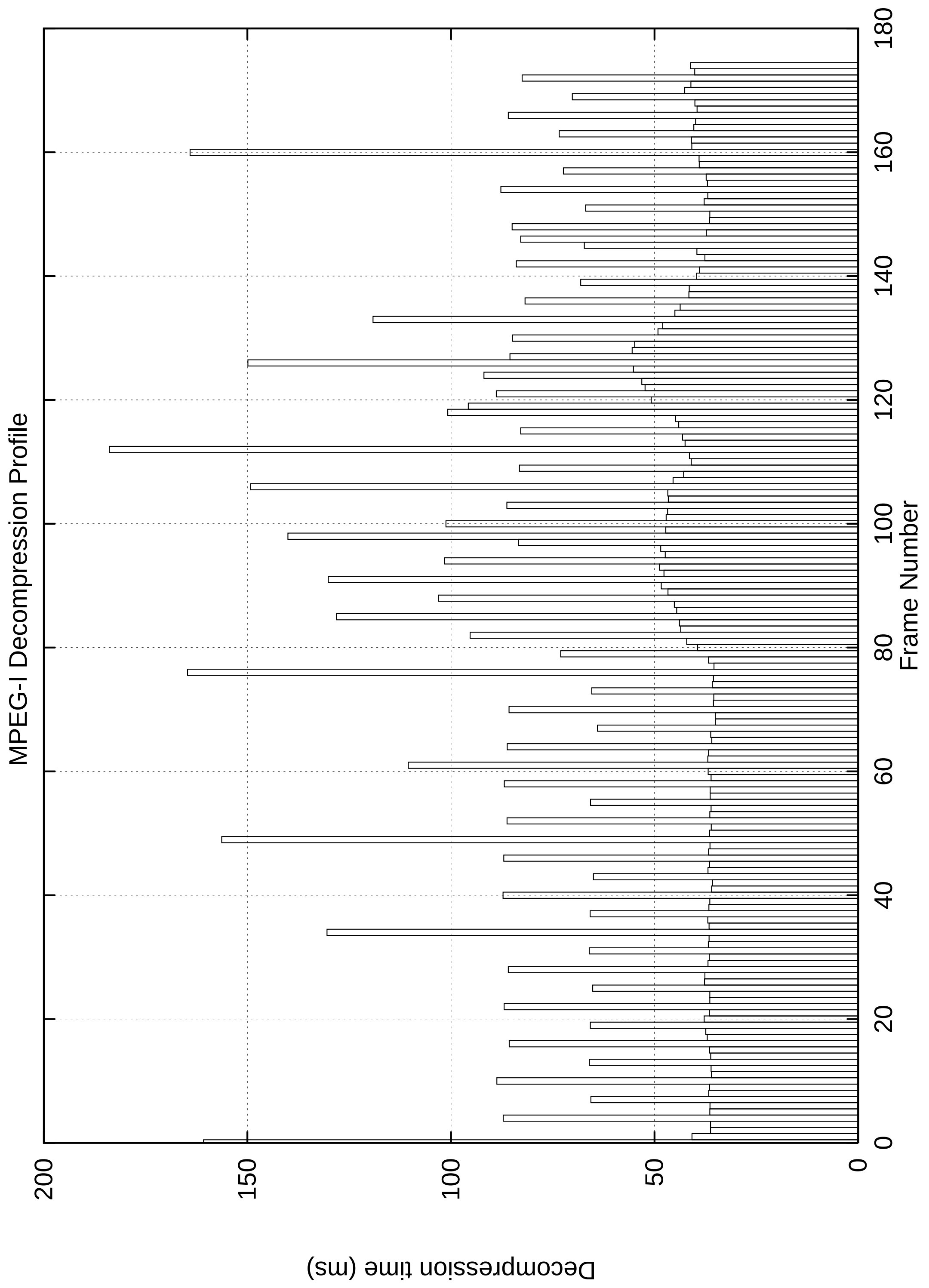

Figure 25: Decompression time profile for MPEG-1: decompression time plotted against frame number (display order).

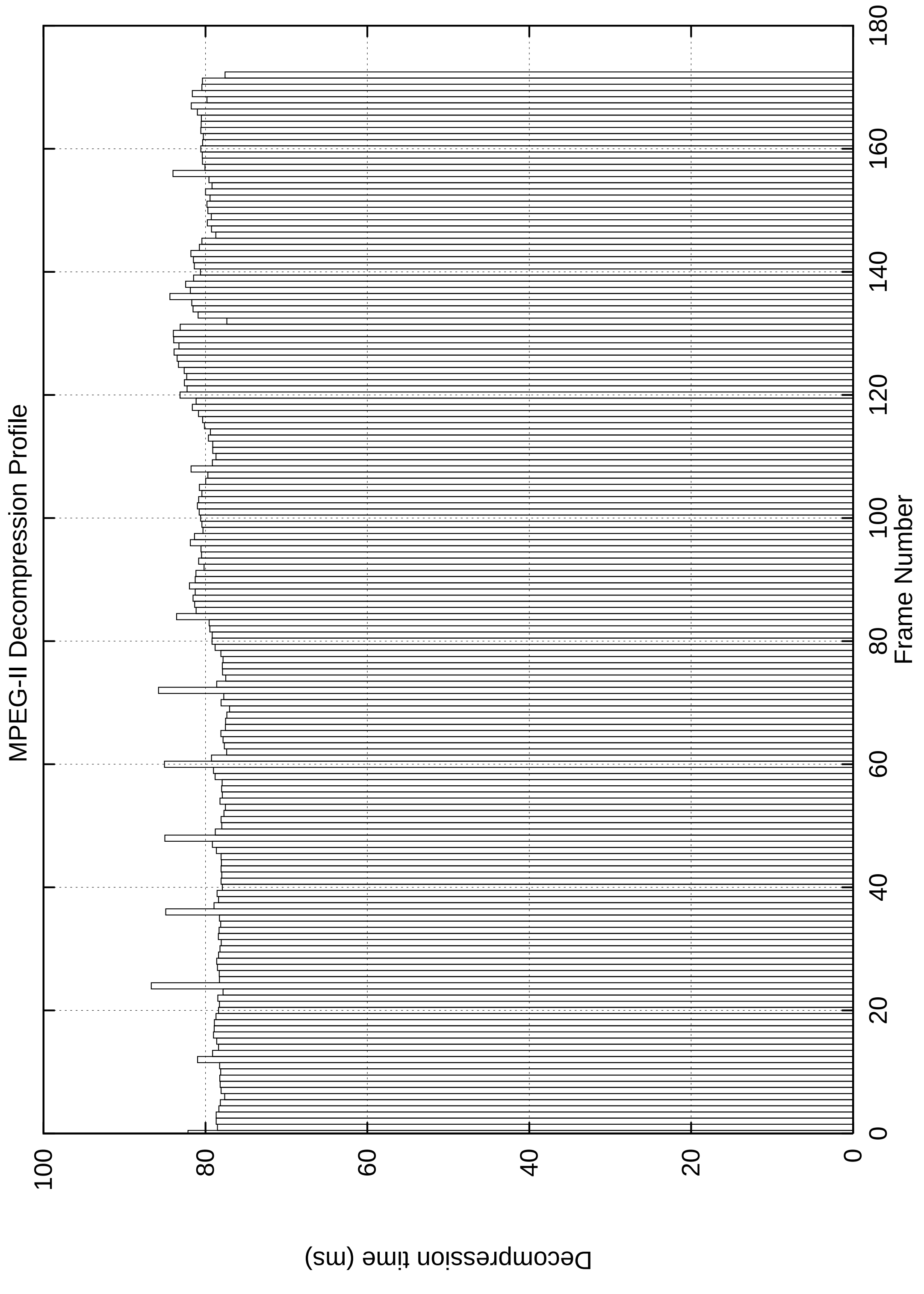

Figure 26: Decompression time profile for MPEG-2: decompression time plotted against frame number (display order).

Observe that the MPEG-2 video is slower to decompress (about 13.7 sec.) than the MPEG-1 video (about 10.4 sec.). Thus, neither decompressors are able to decompress the 9.7 second video in real-time, although MPEG-1 is close.

3.4 CPU-usage: Compression

Purpose
We like to determine the variation in CPU-usage during compression of a movie, and compare MPEG-1 and MPEG-2 wrt. CPU-usage.

Method
We have instrumented existing shareware MPEG-1 and MPEG-2 software compressors with code that measures the time spent on compressing each frame. Our test movie was then compressed with these tools on U1 hosts.

Expectations
We should see a dependency on frame type and on the movement in the movie.

Results
Figure 28 shows the compression time profile for the MPEG-1 compressed test movie. There is a clear relation between frame type and compression time, but now the situation is the inverse of decompression: I-frames are the fastest to compress, P-frames the second fastest, and B-frames the slowest (See also statistics summary in Figure 30). It can also be observed, that the movement between frame 80 to 140 causes longer compression times, although not significantly. This is not surprising since, intuitively, a lot of movement should be more difficult to predict than little movement.

Figure 29 shows the compression time profile for the MPEG-2 compressed test movie. Again, I-frames are compressed faster than P-frames, about 2 times as fast. In contrast to MPEG-1 there appear to much more variation in the compression times of P-frames.

The most important observation that should be made from the compression experiment, is that it appears infeasible to do real-time software compression. In MPEG-1 measurement, it takes over half a second to compress a frame. MPEG-2 is even worse, over 2 seconds. The compression used in these experiments are thus asymmetrical: it takes longer time to compress than to decompress. This is suitable for a system where compression is done once and playback is done many times, as is done in video on demand applications. Both MPEG-1 and MPEG-2 should be suitable for symmetrical compression [20]. This cannot be directly concluded from our experiment. Observe however that while MPEG-1 is faster in compression than MPEG-2, it also compresses less (requires more bandwidth). Thus, it may be possible to get symmetrical compression by configuring the tools to provide a higher output bandwidth, or

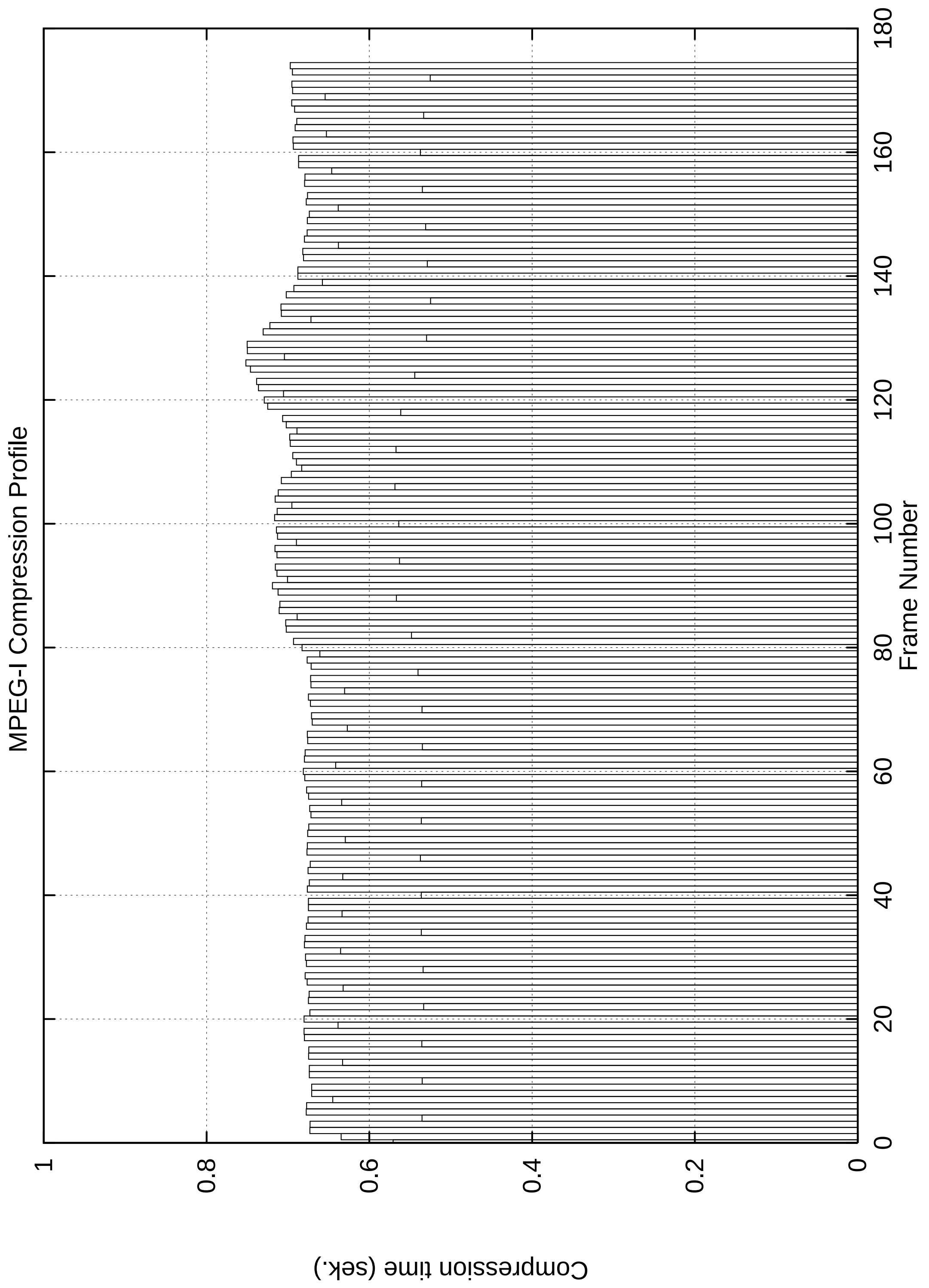

Figure 28: Compression time profile for MPEG-1: compression time plotted against frame number (display order).

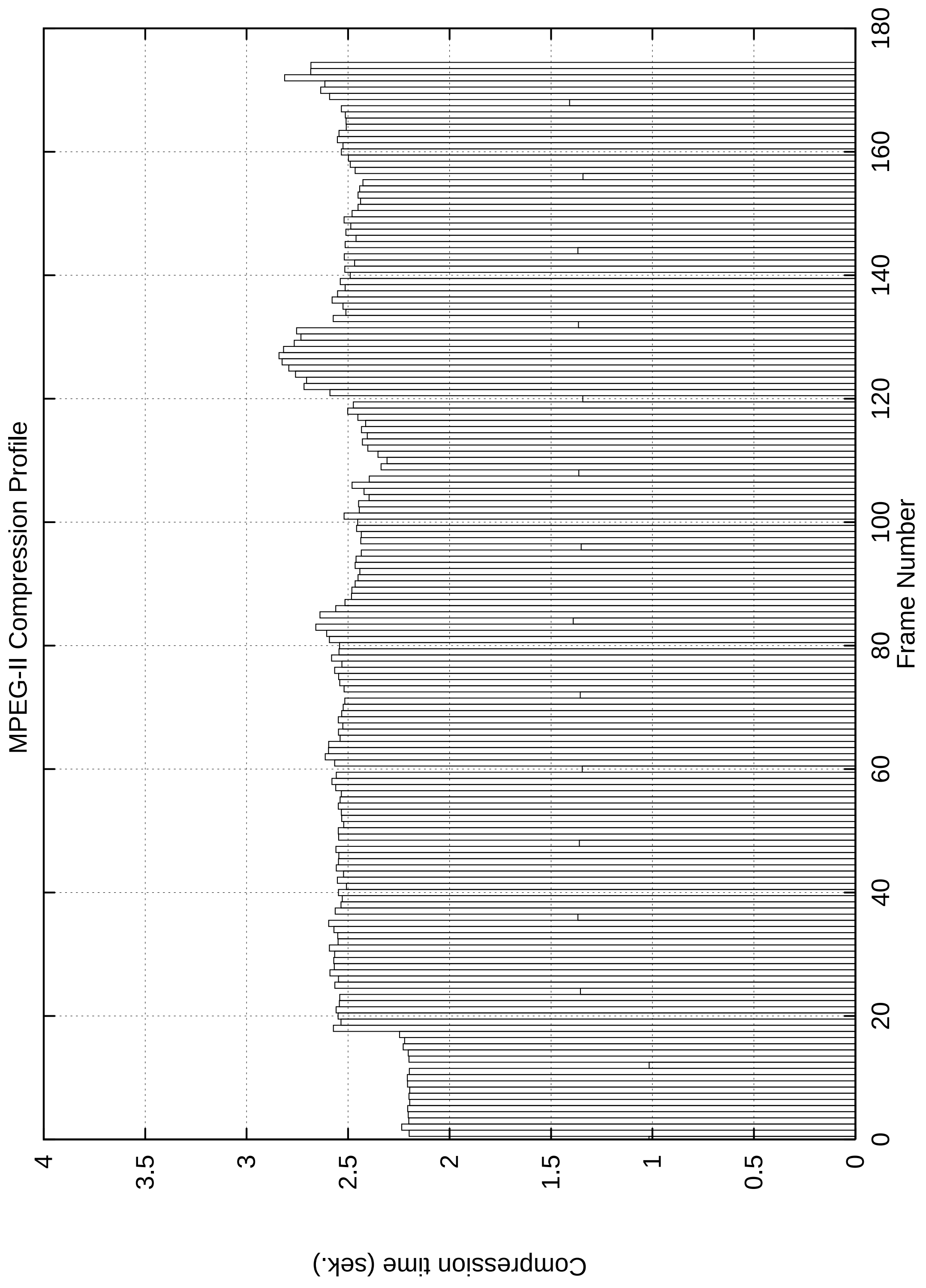

Figure 29: Compression time profile for MPEG-2: compression time plotted against frame number (display order).

compress less. We conclude that further experiments are needed to uncover this possibility.

It should be noted however, that due to the high cost of software compression most frame grabbers come with built in hardware compression.

	Compression time (sec)				
	MPEG-1			MPEG-2	
	I-frame	P-frame	B-frame	I-frame	P-frame
average	0.54	0.66	0.69	1.32	2.51
minimum	0.52	0.63	0.67	1.02	2.20
maximum	0.57	0.71	0.75	1.41	2.84
Total	115			421	

Figure 30: CPU-usage: compression statistics

3.5 Display performance

Purpose
The purpose is to determine the maximum frame rate a workstation can display.

Method
An arbitrary decompressed frame is selected and displayed using the X11 library. The time required to display the frame on an U1 host is measured. The experiment is repeated 2000 times.

Results
The average display time, along with observed minimum and maximum values, are tabulated in Figure 31. It takes about 12.7 ms to display a single frame. This gives an upper bound on frame rate of 79 fps. Thus, the workstation is, without decompression load, clearly able to display the movie in real-time (with 18 frames per second).

	Display time (ms)
average	12.7
minimum	7.2
maximum	13.3

Figure 31: 400×320 (8bit color) display time

4 Discussion

4.1 Calculation of QoS-settings

ATM-networks provide QoS guarantees for its connections. The desired and minimum acceptable QoS values must be specified when the connection is opened. In our testbed the required parameters are mean bandwidth, peak bandwidth, and mean burst length, see Section 1.4. Thus the application must provide a reliable estimate of the bandwidth to be used during the connection. If the estimate is less than what will actually be used the connection will be policed, possibly leading to many cell losses, and ultimatively to a poorer play-back quality. If the estimate is overly conservative, the network will be under utilized because the network must reserve resources according to the specified QoS demands.

One way to estimate QoS values is to calculate them from the bandwidth statistics given in Section 3.2 of the video clip compressed with MPEG-1 and MPEG-2. The total MPEG-1 video size is 1379.3 kbyte, and an easy calculation shows that sending this movie with 18 fps requires an average bandwidth of about 1.14 Mbit/s. The same calculation for MPEG-2 gives an average bandwidth requirement of 450 kbit/s. The largest MPEG-1 frame is 33 kbyte. Sending this in $\frac{1}{18}$ second requires a peak bandwidth of about 4.7 Mbit/s. The MPEG-2 peak bandwidth is 2 Mbit/s. The mean burst length can in each case be set to the average I-frame size, 23 kbyte for MPEG-1 and 11.2 kbyte for MPEG-2.

However, this technique applies only to a particular video, and additionally presumes that the entire video is available for analysis before a connection is established. This technique is applicable to applications like video on demand where the connection parameters can be pre-determined and stored along with the video. It is clearly inapplicable to live video services. In this case it is highly desirable to use a compression technology/tool whose output stream conforms to a specified target data rate (average bandwidth) and worstcase burstiness. Our MPEG-2 compressor is indeed capable of adjusting compression to a specified target rate, however, it appears impossible obtain information about or limit the burstiness of the produced data stream. No bandwidth parameters could be specified for our MPEG-1 compressor.

A technique for managing an unpredictable stream is to drop frames occasionally to limit the produced bandwidth/burstiness. In [11] we conducted a number of experiments which showed that decoders are very resilient to frame drops, although the quality can be visibly degraded in extreme cases. A further technique is to adopt the connection's QoS values to the produced stream, either by re-establishing the connection when a discrepancy have been detected, or by using the proposed concept of QoS renegotiation [18](not employed in current ATM-networks).

4.2 Bandwidth Utilization

In Section 2.1 we measured the achievable throughput for each of our host/protocol configurations, and in Section 3.2 we determined the bandwidth usage of MPEG-1 and MPEG-2 compressed video. Given these numbers we can make a statement about the number of test video streams that in principle could be communicated in each of our host/protocol combinations.

	Number of streams					
	SS20-SS20			U1-U1		
	AAL-5	$\mathrm{UDP}_{\mathrm{atm}}$	$\mathrm{UDP}_{\mathrm{eth}}$	AAL-5	$\mathrm{UDP}_{\mathrm{atm}}$	$\mathrm{UDP}_{\mathrm{eth}}$
MPEG-1	105	50	8	55	105	8
MPEG-2	266	128	21	140	266	21

Figure 32: Number of transferable test-video streams in each configuration

The number of transferable streams is calculated as the integer part of the maximum throughput divided by the average bandwidth usage by the MPEG compressed video. The result is shown in Figur 32. As expected the lowest number of streams is obtained when the MPEG-1 stream is communicated using UDP on an Ethernet (8 streams). The maximum number of streams is the MPEG-2 video communicated between SS20 hosts using AAL-5, or between U1 hosts using UDP (both 266 streams). Obviously the hosts would be unable to process and display these—as previously noted in Section 3.3 decompression is very CPU intensive, and neither of the streams can be decompressed in real-time, although it is close with the MPEG-1 video.

Beware of another limiting factor: network buffer space at the sender and receiver. If all streams send with their peak-rate simultaneously buffer overflow and consequent cell-loss is likely. This may occur when too many video streams are transmitted with I-frames coinciding. A solution is to send the streams *phase-shifted* such that a large frames in one stream coincide with a small frame in another stream.

4.3 End-to-End Delay

As stated in the introduction certain real-time requirements govern the transmission of video. A frame in an interactive video stream must be delayed less than 250 ms (the maximum delay between frame capture and display). This implies that the grab, compression, communication, decompression and display of a frame totally must take less than 250 ms.

Three factors influence the end-to-end delay: compression technology, speed of compression/decompression, and network delay. A particular combination of compression technology, host speed, and network protocol therefore gives different end-to-end delay. In Figure 33 we have tabulated four configurations

(MPEG-1 on Ethernet,MPEG-1 on AAL-5, MPEG-2 on Ethernet, MPEG-2 on AAL5; all with U1 hosts), versus the various delay contributing factors.

	End-to-end delay (ms)			
	MPEG-1 +ETH	MPEG-1 +AAL-5	MPEG-2 +ETH	MPEG-2 +AAL-5
SW compression	750	750	2840	2840
B-frame penalty	167	167	0	0
Transfer time	>29	56	>12	56
Latency	<1	<1	<1	<1
Jitter	60	<1	60	<1
Decompression	184	184	87	87
Display time	13	13	13	13
total (excl. SW comr)	454	422	173	158

Figure 33: End-to-end delay in four configurations

The factors are determined as follows: Software compression and decompression time found by the worstcase processing time in the software compression/decompression experiments in Figures 27 and 30. The B-frame penalty is caused by MPEG-1's use of B-frames that refers to future frames; with our GOP pattern "IBBP" the first B frame cannot be compressed until the P frame has been digitized and compressed. This causes a delay of 3 frames, equal to 167 ms. The transfer time is the time required to communicate the largest frame (from Figure 21) on the network. In the AAL-5 case frames are transfered with a rate determined by the QoS settings: If these values are set as defined in Section 4.1 it will take at most $\frac{1}{18}$ second to transfer one frame. In the Ethernet case frames are transferred with anything between 0 and 9 Mbit/s, depending on load. 9 Mbit/s is used in the above calculations; consequently the Ethernet transfer times are best case values. The latency and jitter contributions are measured in Section 2.2 and 2.3.

The total end-to-end delay is computed by adding all contributions except software compression. With current computers this is infeasible and should be done in dedicated hardware. However, since we have no numbers on hardware compression we have decided to omit this contribution altogether.

From Figure 33 it can be seen that neither MPEG-1 configuration satisfies the required 250 ms. This is primarily due to the B-frame penalty and the slow decompression of the worst case frame. The situation could be improved by avoiding the use of B-frames. This reduction (to 255 ms) is in fact nearly sufficient to enable MPEG-1 data to be communicated on ATM within the time limit. Jitter makes MPEG-1 communication on an Ethernet infeasible. Also, the disadvantage of avoiding B-frames is the much lower compression rate.

Both MPEG-2 configurations are feasible; primarily because there is no delay due to B-frames, and because the worst case decompression time is much less than MPEG-1. Note the difference between Ethernet and ATM (173 ms vs 158 ms) is larger than illustrated because the transfer time on Ethernet is based on a best case value.

In conclusion, the configuration with the lowest end-to-end delay is achieved when MPEG-2 is combined with ATM.

4.4 Future Experiments

During our experiments we found a number of relevant supplementary experiments that should be done:

- The causes of the poor performance of AAL-5 on the fast U1 machines for large messages should be found. We suggest experiments that examine the buffer space allocation for the AAL-5 protocol. Also differences in software drivers and hardware should be examined more thoroughly.
- We observed that the networks started dropping messages when very large messages were sent with maximum rate. An experiment should be performed, that shows the actual achievable throughput for very large messages (sustained throughput), i.e., the sending rate should be regulated such that message loss is avoided. In addition we suspect that the operating system's available buffer space influence drop rate. Consequently, it may be necessary add more memory and to adjust parameters in the operating systems buffer allocation schemes.
- Our experiments were conducted on a local area network. It would be interesting to see how the performance parameters change on a wide area network.
- Our jitter measurements with parallel load were subject to an uncontrolled load (as much as possible). A future experiment should throttle the load process to make the load known and predictable.
- The Ethernet measurements was performed on a 10 Mbit/s Ethernet. However, it would be interesting to determine to which extent faster versions of the Ethernet technology (100 Mbit/s FastEthernet or even 1 Gbit/s Ethernet) has the same profound problems. A simulation study [27] of Ethernet performance for multi-media traffic suggests that 10 times as large bursts of background traffic is required on a 100 Mbit/s Ethernet to give the same amount disturbances as on a 10 Mbit/s Ethernet. An open question is whether hosts can/will produce frequent large bursts on a 100 Mbit/s Ethernet. If not, multi-media may possibly be supported adequately by a high bandwidth Ethernet.
- We found that to compare MPEG-1 and MPEG-2 genuinely their produced bitrates should be related to their visual quality.
- We need to determine if symmetrical compression/decompression is possible at the expense of lower compression rate.

4.5 Related Work

Our experiments were designed to give insight in resource management problematics for video transmission on our local testbed. Others have made similar types of experiments, and we review a few of these and other related work next. These experiments are conducted with different aim than ours and in different testbeds and cannot replace the benchmark of our testbed.

Measurements of throughput and latency is part of nearly all network/protocol performance benchmarks. Measurements of jitter however, is almost exclusively found in work related to network/protocols for real-time or multi-media transmission. In the study reported in [16] jitter is measured in a network consisting of a combination of Ethernet and FDDI. The measured jitter values are in the same magnitude as found in our experiments, i.e., several mili-seconds, and therefore seem to confirm our results. The study also found a corelation between packet size and jitter: The larger packets the more jitter. This observation is significant because audio tend to be transmitted as frequent small packets (therefore subject to less jitter) whereas video tend to be transmitted as fewer large packets (subject to more jitter). Also the study finds that the amount of background load influence jitter.

In contrast with our jitter experiments and [16] another study [6] finds much smaller jitter values, less than 1 ms for an Ethernet (assuming low collision rates, less than 3%) . The study uses a network simulator to determine jitter at the MAC (medium access layer). Jitter is thus measured at a much lower level in the protocol stack than in our experiments (UDP socket level). This difference in jitter at the two levels indicates that protocol implementation in operating systems plays a significant role. It would therefore be interesting to see if it is possible to device a more real-time friendly implementation of the UDP protocol.

There are numerous experiments that determine the bitrate of compressed video. Results from such experiments are used in many studies where a characterization of the traffic pattern is necessary, e.g., [18, 15]. These studies plots bandwidth profiles with roughly speaking identical shapes as ours. Thus, the dependency on frame type and on media contents seems to be confirmed by others.

The MPEG-2 standart [13] includes a description of how compressed video can be transported across a network. The output of the compressor is broken up into socalled transport stream packets of 188 bytes. These packets contain information that allows the receiver to select the appropriate decompresser (audio/video), and enables the decompressor to decompress the packet in isolation. i.e., without reference to other packets. A packet thus provide useful information even if other packets in the same frame are dropped by the network. A study [31] shows that MPEG requires advanced error concealment algorithms to give a satisfactory quality on very lossy networks, such as a wireless local network.

The ATM forum specification [3] defines how a constant packet rate MPEG-2 stream can be mapped to an ATM network using AAL-5. The document includes a specification of system configurations, interfaces, QoS parameters etc. However, transmission of compressed video with a constant rate requires more buffering at the hosts in order to smooth the variable bitrate produced by the compressor. In [11] we examined another strategy where entire frames were mapped to AAL-5 protocol data units which then are transmitted with a variable bitrate.

The problems of communicating multi-media traffic on networks without QoS support, which we have conformed here, are well accepted, and solutions have been proposed at various levels. An isochronous Ethernet [28] is an extension of the Ethernet technology that in addition to a normal 10 Mbit/s Ethernet channel also carries a channel with time sensitive data. The Resource Reservation Protocol (RSVP) [30] is a proposal for a protocol that enable connections that require quality of service guarantees to be supported on the internet. Application level solutions also include the use neural networks [29] or probabilistic [7] models to predict the network delay and use these predictions in the scheduling of a multi-media stream.

4.6 Conclusions

This report documents a number of experiments related to technical conditions of the realization of multi media communication. Multi-media applications require real-time communication and processing of large amounts of data. The consequence of not satisfying the real-time requirements is unsatisfactory quality of the presentation.

Our goal has been to identify how different protocols and compression technologies influence our ability to satisfy multi-media's real-time and bandwidth requirements. In addition, it has been important for us to get practical experience with network real-time performance and compression techniques. We therefore benchmarked the communication performance of our platform and two compression techniques.

In our network experiments we compared AAL-5 (ATM), UDP (ATM), and UDP (Ethernet) with respect to throughput, latency, and jitter performance. Also the influence fast and slow hosts were considered. This gives a total of six host/protocol configurations.

We found that ATM is superior to Ethernet in all three metrics. Of ATM's bandwidth of 155 Mbit/s (134 Mbit/s is available for user data) around 120 Mbit/sec. could be utilized at the hosts. This means in principle that a host is able to send or receive 120 Mbit/s of multi-media traffic. 9 Mbit/s out of the Ethernet's 10 Mbit/s could be utilized. Three factors were found to influence latency: host speed, protocol, and network type. As expected AAL-5 is the fastest with a latency of 201 μs. UDP on ATM is second fastest, and UDP on Ethernet is slowest. Fast hosts were about twice as fast as the slow hosts. Our jitter measurements revealed a significant difference between ATM and Ethernet. Both networks yielded low jitter values (less than 1 ms) on an unloaded network. When a load was added, the ATM were still within 1 ms, but Ethernet yielded extremely high values, often up to 60 ms and more. This level of jitter creates problems for an video communication implementation which must take extra care in its buffering and playback strategy. Thus, we have confirmed that the quality of service concept in ATM networks is indeed significant in practice.

We also found a few deficiencies of ATM. With its more than ten fold higher bandwidth its latency was only twice as fast as the Ethernet. Thus, valuable time is lost in operating system and network controller. The throughput on the fast hosts only reached 65 Mbit/s using the ATM native AAL-5, whereas UDP reached 120 Mbit/s. We have no definite explanation for this anomaly. In addition, QoS cannot be specified in the UDP protocol.

In our compression experiments we compared the MPEG-1 and MPEG-2 compression technologies with respect to their bandwidth usage, compression and decompression cpu time usage. We recorded a test movie and analyzed this with MPEG tools instrumented with measurement code.

We found that MPEG-2 was able to compress the test movie better (a data rate of 0.45 Mbit/s) than MPEG-1 (1.1 Mbit/s) at the same or better quality, subjectively judged. However, MPEG-2 seems to achieve its low bandwidth at a higher compression cost than MPEG-1. MPEG compressors produce variable bitrate traffic that depends on frame type (I, P or B) and frame contents, e.g., movement. Both were apparent in our measurements. Likewise, CPU compression and decompression time also varies with frame type. Moreover, the variation in bandwidth usage and decompression time is so significant (in particular for MPEG-1) that careful network and CPU scheduling is necessary in order to achieve optimal resource utilization and to satisfy real-time requirements. We found that software compression is infeasible with current cpu-speeds. It takes in the magnitude of seconds to compress a single frame. Thus, dedicated compression hardware is necessary. In contrast, decompression is feasible now for low quality video (small picture sizes, low frame rate) and will soon be feasible for higher quality videos.

The end-to-end transmission of multi-media data uses a combination of network protocol and compression technology. Based on our measurements we calculated the number of streams that a particular combination of protocol and compression technology configuration could support, and its end-to-end delay, including the delays related to communication and compression. The combination of AAL-5 and MPEG-2 stands out: it supports the most streams, and gives the lowest end-to-end delay.

In conclusion, our experiments has given us a usefull insight in the real-time performance characteristics of communication protocols for video communication and in compressed video. Also, many essential performance parameters of our local testbed has been uncovered. With these numbers we have laid a necessary foundation for design and construction of multi-media applications and support systems. Our results are employed in the design and construction of a multi-media support system [11, 12].

References

[1] 3Com. 3Com SuperStack II 1000 switch: Users Guide.

[2] Andy C. Hung (achung@cs.stanford.edu). MPEG-1 Compressor implementation. Available from ftp://havefun.stanford.edu.

[3] The ATM-Forum Technical Committee. Video on Demand Specification 1.0. Technical Report af-saa-0049.000, ATM Forum, 1995.

[4] The ATM-Forum Technical Committee. Traffic Management Specification 4.0. Technical Report af-tm-0056.000, ATM-Forum, April 1996.

[5] Standard Performance Evaluation Corporation. SpecINT benchmark. Results available at http://www.specbench.org/cgi-bin/osgresults.

[6] Shuang Deng, Alan R. Bugos, and Paul M. Hill. Design and Evaluation of an Etherned-Based Residential Network. *IEEE Journal on Selected Areas In Communications*, 14(6):1138–1150, August 1996.

[7] John F. Gibbon and Thomas D.C̃. Little. The Use of Network Delay Estimation for Multimedia Data Retrieval. *IEEE Journal on Selected Areas In Communications*, 14(7):1376–1387, September 1996.

[8] Greg Ward (greg@bic.mni.mcgill.ca). MPEG-1 Decompressor implementation. Available from http://www.mni.mcgill.ca/users/greg/mpeg.html.

[9] MPEG Software Simulation Group. MPEG-2 Compressor/Decompressor implementation. Available from http://www.mpeg.org/MSSG/ and ftp://ftp.mpeg.org/pub/mpeg/mssg/.

[10] Kaj Henriksen, Rolf J. Hillemann, Wladyslaw Pietraszek, Arne Skou, and Michael Aaen. Eksperiments with TCP/IP in ATM High-Speed Data Communications. Technical Report R-95-2026, Aalborg University, Department of Computer Science, December 1995.

[11] Thomas Husfeldt, Finn Norman Pedersen, and Dao Van The. Video Communication using ATM. Internal s9d semester project report, University of Aalborg, Institute for Electronic Systems, Department of Computer Science, January 1996.

[12] Thomas Husfeldt, Finn Norman Pedersen, and Dao Van The. Adaptive Multimedia Scheduling. Master's thesis, University of Aalborg, Institute for Electronic Systems, Department of Computer Science, June 1997.

[13] ISO/IEC. Generic Coding of Moving Pictures and Associated Audio Information: Video. ISO/IEC International Standard 13818-2, 1995.

[14] Sandeep Khanna, Michael Sebreé, and John Zolnowsky. Realtime Scheduling in SunOs 5.0. In *Proceedings of Usenix Winter Conference*, pages 375–390, San Francisco, CA, July 1992. USENIX.

[15] Soung C. Liew and Chi-Yin Tse. Video Aggregation: Adapting Video Traffic for Transport Over Broadband Networks by Integrating Data Compression and Statistical Multiplexing. *IEEE Journal on Selected Areas In Communications*, 14(6):1123–1137, August 1996.

[16] Changdong Liu, Yong Xie, Myung J. Lee, and Tarek N. Saadawi. Multipoint Multimedia Teleconference System with Adaptive Synchronization. *IEEE Journal on Selected Areas In Communications*, 14(7):1422–1435, September 1996.

[17] Daved E. McDysan and Darren L. Spohn. *ATM: Theory and Application.* McGraw Hill, Inc., 1994. ISBN 0-07-060362-6.

[18] Daniel J. Reininger, Dipankar Raychaudhuri, and Joseph Y. Hui. Bandwidth Renegotioation for VBR Video Over ATM Networks. *IEEE Journal on Selected Areas In Communications*, 14(6):1076–1086, August 1996.

[19] Lua Sha, Ragunathan Rajkumar, and John P. Lehoczky. Prioriry Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[20] Ralf Steinmetz and Klara Nahrstedt. *Multi-media: Computing, Communication, and Applications.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1995. ISBN 0-13-324435-0.

[21] Morten Stor. Asynchronous Transfer Mode. Technical report, Tele Danmark Research, 1994.

[22] FORE systems. Unix man pages to ATM application programming interface.

[23] FORE systems Whitepaper. ForeThought Bandwidth Management. Technical report, FORE Systems, 19?? Version 1.0, http://www.fore.com/atm-edu/whitep/index.html.

[24] FORE systems Whitepaper. ForeRunner ATM Switch Architecture. Technical report, FORE Systems, 199? Version 1.0, http://www.fore.com/atm-edu/whitep/index.html.

[25] FORE systems Whitepaper. Traffic Management Version and Congestion Control. Technical report, FORE Systems, 1994. Version 1.0, http://www.fore.com/atm-edu/whitep/index.html.

[26] FORE systems Whitepaper. Quality of Service Support for IP-based Applications. Technical report, FORE-systems, 1996. http://www.fore.com/atm-edu/whitep/index.html.

[27] Fouad A. Tobagi and Ismail Dalgiç. Performance Evaluation of 10Base-T and 100Base-T Ethernets Carrying Multimedia Traffic. *IEEE Journal on Selected Areas In Communications*, 14(7):1436–1455, September 1996.

[28] Debra J. Worsley and Tokunbo Ogunfunmi. Isochronous Ethernet—An ATM Bridge for Multimedia Networking. *IEEE Multimedia*, pages 58–67, Januar-March 1997.

[29] Maria C. Yuang, Po L. Tien, and Shih T. Liang. Intelligent Video Smoother for Multimedia Communications. *IEEE Journal on Selected Areas In Communications*, 15(2):136–146, February 1997.

[30] Zhang, Deering, Estrin, Shenker, and Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, September 1993.

[31] Jian Zhang, Michael R. Frater, John F. Arnold, and Terence M. Percival. MPEG2 Video Services for Wireless ATM Networks. *IEEE Journal on Selected Areas In Communications*, 15(1):119–128, January 1997.