1. INTRODUCTION

With the growth of computer networks and their specific uses in networked audio and music, the desire for sharing real-time audio has inspired research by both musicians and network engineers.

To measure the performance of real-time network streaming systems, some parameters, such as bandwidth, latency, jitter and packet loss, can be used.

Latency corresponds to how long a message takes to travel from one end of a network path to the other [12], and in the case of audio streaming, latency is perceived as delay. Jitter is defined as variation in latency over time and it is calculated as the standard deviation of latency measurements. Packet loss is the percentage of packets lost in transmission, received corrupted or received out of order and therefore rejected.

A point raised frequently in the computer networking literature ([5], [19]) is that multimedia streaming requires high bandwidth and low latency. It is also a common view in the same literature that data loss is not a crucial impairment to audio streaming, because lost packets might only result in small glitches in the resulting audio/video streams, without compromising intelligibility.

Different audio application scenarios may not agree with the aforementioned views, and according to specific requirements they may allow the trade-off between latency and packet loss to tilt to one side or the other. For instance, in a distributed music rehearsal one would not worry so much about packet loss but would with latency, whereas in a distributed musical recording there should be no packet loss, but high latency values might be overlooked [14].

From a low-level networking point of view, latency and packet loss are features directly related to the protocol implementation and to the concept of protocol reliability. Thus, a reliable network protocol uses techniques to achieve reliability that would normally increase the delivery time of a network packet, resulting in increased latency. Since an unreliable protocol like UDP is usually faster than a reliable protocol, UDP is suggested by many computer networking textbooks as the best protocol choice for audio streaming. In fact we do see the use of the UDP protocol in many existing real-time audio streaming applications, such as NetJack [3], SoundJack [4] and JackTrip [2].

As we examine different usage scenarios, like rehearsal and recording, we are invited to review the way audio is distributed over computer networks and to consider other transport protocols besides UDP [13]. TCP [10], for instance, is a reliable protocol and its use should have less packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.

In this paper we have explored certain aspects of network communication to support the development of Medusa [15], a distributed audio environment.

In this paper we have considered the specific case of audio streaming within local computer networks, as a first testbed for future discussion of audio streaming over wide-area networks. The remainder of this paper is organized as follows: Section 2 presents the network architecture and the four transport layer protocols we studied, Section 3 and packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.

In this paper we have explored certain aspects of network communication to support the development of Medusa [15], a distributed audio environment.

In this paper we have considered the specific case of audio streaming within local computer networks, as a first testbed for future discussion of audio streaming over wide-area networks. The remainder of this paper is organized as follows: Section 2 presents the network architecture and the four transport layer protocols we studied, Section 3 and packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.

In this paper we have explored certain aspects of network communication to support the development of Medusa [15], a distributed audio environment.

In this paper we have considered the specific case of audio streaming within local computer networks, as a first testbed for future discussion of audio streaming over wide-area networks. The remainder of this paper is organized as follows: Section 2 presents the network architecture and the four transport layer protocols we studied, Section 3 and packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.

In this paper we have explored certain aspects of network communication to support the development of Medusa [15], a distributed audio environment.

In this paper we have considered the specific case of audio streaming within local computer networks, as a first testbed for future discussion of audio streaming over wide-area networks. The remainder of this paper is organized as follows: Section 2 presents the network architecture and the four transport layer protocols we studied, Section 3 and packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.
works, IP has two important features: MTU measurement, the maximum size of an IP header is 60 bytes [11]. Although many books suggest also the RTP protocol for sending multimedia streams, it should be noted that RTP is actually not a transport layer protocol but an application layer protocol, created as an addendum to UDP, with a timestamp and sequence number added to each UDP datagram. RTP always appears together with RTCP (Real Time Control Protocol) [20], a control protocol used for providing reliability to RTP. Another application layer protocol used for streaming multimedia media is RTSP (Real Time Streaming Protocol) [16]. RTSP was designed for streaming and also features commands like START and PAUSE. Since our research is focused in transport protocols, applications protocols like RTP, RTCP and RTSP will not be covered in this paper.

The IP protocol will be presented first, and then the UDP, TCP, SCTP and DCCP transport protocols will follow.

2.2. Internet Protocol (IP)

IP is the network layer protocol in the TCP/IP protocol suite [17] that is used to carry almost all user data in TCP/IP networks. IP packets are called datagrams, and user data is sent as a packet with an IP header. The normal size of an IP header is 20 bytes, unless other options are present, as shown in Fig. 2. With extra options and IPv6 information, the maximum size of an IP header is 60 bytes [11]. Besides encapsulating data to be sent over the network, IP has two important features: MTU measurement and TTL management.

MTU is the Maximum Transmission Unit of the Link layer, and represents the upper limit on the size of data packets that can be sent. Since IP is implemented independently of the Link layer, the MTU can be of any size; if an IP datagram is larger than MTU, it will be broken into fragments smaller than MTU [17], increasing the transmission overhead, since each fragment requires a new IP header. Meanwhile, the ethernet payload size, adopted as a standard packet length, is defined as 1500 bytes, and thus defines the upper bound on any packet size [19]. TTL (Time to Live) is the life time of a network packet, which is used to avoid packets left wandering around indefinitely. Although the name suggest a timestamp, this unit refers to the number of hops that a packet can reach between the packet sender and the receiver. Every hop that a packet reaches should decrease its TTL and then forward it. If a datagram's TTL value is zero, devices on the packet path can ignore it and simply remove it from the sending queue. Thus TTL management is directly linked to packet loss, because a network device can discard a packet if TTL reaches zero.

2.3. User Datagram Protocol (UDP)

UDP is the most simple transport protocol that runs over IP. Like an IP packet, a UDP message is also called a datagram, and each datagram is handled independently of all others. The UDP protocol merely encapsulates application data with its 8-byte header [17], shown in Fig. 3. Thus, UDP is a lightweight transport protocol with a minimalistic service model, that basically demultiplexes messages to the application layer [8, 12].

Just as with normal IP packets, UDP datagrams can be lost, duplicated or arrive out of order. That is why UDP is referred to as an unreliable transport protocol. The only guarantee that a UDP datagram has is its checksum; through it a datagram’s contents can be checked for consistency. However, UDP does not specify what will be done if a checksum does not match a datagram’s contents; this protocol does not entail package retransmission. For this reason, applications that use UDP must be prepared to deal with error recovery, packet loss, packet reordering, flow control, congestion control and so on [5, 11]. At the application level, each transmission request issued by a process produces exactly one UDP datagram, which causes one or more IP packets to be sent, depending on the network physical MTU. Theoretically, the maximum size of a UDP datagram is 65535 bytes, including data and headers [17]. Finally, UDP has no acknowledgement messages, neither initial or final handshaking messages. There is no way to know if a packet was correctly sent or if a client is still connected to a server [8].

2.4. Transmission Control Protocol (TCP)

TCP is the classical reliable transport protocol from the TCP/IP suite. Since TCP is a byte-stream protocol, TCP messages are called “segments” of the data stream. Each segment is packaged with a 20-byte TCP header as depicted in Fig. 4. TCP provides a connection-oriented, reliable byte-stream service [17] consisting of the following:

1. MSS: The largest chunk of data that TCP is allowed to send is called Maximum Segment Size (MSS). When a connection is established, each side can inform its MSS [17], in order to prevent overflows.

2. ACK: When TCP receives a segment it sends an acknowledgement packet (ACK), to inform the sender that this segment was correctly received.

3. ACK timeout: TCP maintains a timer for each segment sent, waiting for the other end to acknowledge reception. If an ACK is not received in due time the segment is retransmitted.

4. Checksum: Similarly to UDP, TCP has a checksum header field to check a message for consistency. TCP will discard a segment with an invalid checksum, and will not ACK receiving it; it expects the sender to retransmit it when the ACK timeout expires.

5. Buffer: Independently of MSS, a TCP segment must fit in the IP payload [19]. Since IP packets may arrive out-of-order, a TCP receiver has to sort data when necessary. Packet sorting and duplicate removing are done by the transport layer and every data received by the application layer is guaranteed to be in order and not duplicated.

6. Flow control: Since each end of a TCP connection has a finite buffer space, a TCP receiver informs the sender how much buffer space it has to receive data, preventing overrunning its capacity.

7. Congestion control: While flow control is an end-to-end agreement, congestion control cares about link capacity. This control will prevent TCP to inject more data than the link or switches can hold [12].

A TCP connection is always point-to-point, mainly due to the nature of its reliability mechanisms. Multicas- sing communication, with one sender that sends data to many receivers, is not possible in TCP [8].

2.5. Stream Control Transmission Protocol (SCTP)

SCTP is a connection-oriented transport protocol that provides a reliable full-duplex association [18]. This protocol was originally developed to transport voice over IP (VoIP). An SCTP packet contains a fixed 12-byte header, and is divided into chunks, as depicted in Fig. 5. Each chunk is preceded by a 4-byte header containing its type, a set of flags and the chunk length [18].

Reliability of SCTP is provided through the following features:

1. Multihoming: Instead of regular point-to-point connections, SCTP provides general associations between clients and servers. This feature provides increased robustness against network failure, since an endpoint may have multiple redundant network connections. SCTP can use a workaround in case of a faulty path across the Internet, by switching to...
and TTL management.

MTU is the Maximum Transmission Unit of the Link layer, and represents the upper limit on the size of packets that can be sent. Since IP is implemented independently of the Link layer, the MTU can be of any size; if an IP datagram is larger than MTU, it will be broken into fragments smaller than MTU [17], increasing the transmission overhead, since each fragment requires a new IP header. Meanwhile, the ethernet payload size, adopted as a standard packet length, is defined as 1500 bytes, and thus defines the upper bound on any packet size [19]. TTL (Time to Live) is the life time of a network packet, which is used to avoid packets left wandering indefinitely. Although the name suggests a timestamp, this unit refers to the number of hops that a packet can reach between the packet sender and the receiver. Every hop that a packet reaches should decrease its TTL and then forward it. If a datagram’s time to live is zero, devices on the packet path can ignore it and simply remove it from the sending queue. Thus TTL management is directly linked to packet loss, because a network device can discard a packet if TTL reaches zero.

2.3. User Datagram Protocol (UDP)

UDP is the most simple transport protocol that runs over IP. Like an IP packet, a UDP message is also called a datagram, and each datagram is handled independently of all others. The UDP protocol merely encapsulates application data with its 8-byte header [17], shown in Fig. 3. Thus, UDP is a lightweight transport protocol with a minimalistic service model, that basically demultiplexes messages to the application layer [8, 12].

![Figure 3: Format of an UDP datagram [17]](image)

Just as with normal IP packets, UDP datagrams can be lost, duplicated or arrive out of order, which is why UDP is referred to as an unreliable transport protocol. The only guarantee that a UDP datagram has is its checksum, through it a datagram’s contents can be checked for consistency. However, UDP does not specify what will be done if a checksum does not match a datagram’s contents; this protocol does not entail package retransmission. For this reason, applications that use UDP must be prepared to deal with error recovery, packet loss, packet reordering, flow control, congestion control and so on [5, 11]. At the application level, each transmission request issued by a process produces exactly one UDP datagram, which causes one or more IP packets to be sent, depending on the network physical MTU. Theoretically, the maximum size of a UDP datagram is 65535 bytes, including data and headers [17].

Finally, UDP has no acknowledgement messages, neither initial or final handshake messages. There is no way to know if a packet was correctly sent or if a client is still connected to a server [8].

2.4. Transmission Control Protocol (TCP)

TCP is the classical reliable transport protocol from the TCP/IP suite. Since TCP is a byte-stream protocol, TCP messages are called “segments” of the data stream. Each segment is packaged with a 20-byte TCP header as depicted in Fig. 4.

![Figure 4: Format of a TCP segment [17]](image)

TCP provides a connection-oriented, reliable byte-stream service [17] consisting of the following:

1. MSS: The largest chunk of data that TCP is allowed to send is called Maximum Segment Size (MSS). When a connection is established, each side can inform its MSS [17], in order to prevent overflows.

2. ACK: When TCP receives a segment it sends an acknowledgement packet (ACK), to inform the sender that this segment was correctly received.

3. ACK timeout: TCP maintains a timer for each segment sent, waiting for the other end to acknowledge receipt. If an ACK is not received in due time the segment is retransmitted.

4. Checksum: Similarly to UDP, TCP has a checksum header field to check a message for consistency. TCP will discard a segment with an invalid checksum, and will not ACK receiving it; it expects the sender to retransmit it when the ACK timeout expires.

5. Buffer: Independently of MSS, a TCP segment must fit in the IP payload [19]. Since IP packets may arrive out-of-order, a TCP receiver has to sort data when necessary. Packet sorting and duplicate removing are done by the transport layer, and every data received by the application layer is guaranteed to be in order and not duplicated.

6. Flow control: Since each end of a TCP connection has a finite buffer space, a TCP receiver informs the sender how much buffer space it has to receive data, preventing overrunning its capacity.

7. Congestion control: While flow control is an end-to-end agreement, congestion control cares about link capacity. This control will prevent TCP to inject more data than the link or switches can hold [12]. A TCP connection is always point-to-point, mainly due to the nature of its reliability mechanisms. Multicasting communication, with one sender that sends data to many receivers, is not possible in TCP [8].

2.5. Stream Control Transmission Protocol (SCTP)

SCTP is a connection-oriented transport protocol that provides a reliable full-duplex association [18]. This protocol was originally developed to transport voice over IP (VoIP). An SCTP packet contains a fixed 12-byte header, and is divided into chunks, as depicted in Fig. 5. Each chunk is preceded by a 4-byte header containing its type, a set of flags and the chunk length [18].

![Figure 5: Format of an SCTP packet [9]](image)

Reliability of SCTP is provided through the following features:

1. Multihoming: Instead of regular point-to-point connections, SCTP provides general associations between clients and servers. This feature provides increased robustness against network failure, since an endpoint may have multiple redundant network connections. SCTP can use a workaround in case of a faulty path across the Internet, by switching to...
another address defined in the same SCTP association [18].

2. Multiple streams: SCTP can provide multiple streams between connected endpoints, each with its own reliably sequenced messages. If one message is lost in one of the streams, SCTP does not block messages in any of the other streams [18].

3. Transport-layer fragmentation: SCTP keeps track of a fragmentation point based on the smallest MTU in the path to all peer addresses, and this smallest MTU size is used to split large user messages into smaller pieces that can be sent using a single IP packet [18]. Like TCP and unlike UDP, this fragmentation occurs at the transport-layer instead of the IP layer.

4. Fast retransmission: SCTP, like TCP, uses ACK to allow detection of packet losses. But instead of a normal ACK, it uses selective acknowledgement (SACK) and a mechanism that sends SACK messages faster than usual when losses are detected [11].

5. Non-blocking head of the line: Unlike TCP, which is stream-based, SCTP is message-oriented or message-based like UDP. This allows SCTP to distinguish signaling messages at the transport layer, which are delivered to the application layer as soon as they arrive [11].

SCTP, like TCP, also provides reliability, sequencing, flow control, congestion control, and full-duplex data transfer, through message chunking, checksumming, packet validation and path management [18]. There are two types of SCTP socket implementations: one-to-one sockets and one-to-many sockets.

2.6. Datagram Congestion Control Protocol (DCCP)

DCCP is a transport protocol that combines TCP-friendly congestion control with unreliable datagram semantics, for applications that transfer fairly large amounts of data [6]. It uses a lightweight 12-byte header, as shown in Fig. 6, to avoid network overhead [1].

![Figure 6. Format of a DCCP packet](#)

This protocol was developed for delay-sensitive applications that favor timeliness over reliability [7]. DCCP may use ACK messages, and does not interpret checksum errors as network congestion problems. Since DCCP does not use retransmissions, TCP-like fast-recovery mechanisms are not available. Unlike UDP, DCCP will avoid congestion collapse. It aims to add to UDP a minimum mechanism to support congestion control, such as reliable transmission of ACK information [7].

DCCP, like TCP, provides a single bidirectional uni-directional connection: both data and acknowledgments flow in both directions. At the startup of the connection the endpoints must agree on a set of parameters, such as which congestion control mechanism will be used [7].

2.7. Comparison Draft

To show the main differences between these protocols, a draft table is provided below.

![Table 1. Feature draft table](#)

3. IMPLEMENTATION

For the purpose of this protocol comparison, a front-end for multichannel audio communication was implemented using the Jack audio server and the Linux socket API. Despite all these protocols being able to support full-duplex communication, the implementation distinguishes the roles of sender and receiver. The sender takes audio from the Jack server, resamples it to the desired representation, packs it and the resulting packet into a ring buffer to be sent by another thread. This data will be fragmented into blocks of 1024 bytes before sending to the network. Because of this implementation, some fragmentation can happen in the middle of a packet. Thus, the receiver must identify the beginning of a packet, and then separate header and data. The header is logged for measurement and the data is then resampled and put into a ring buffer to be consumed by the Audio API.

To measure the performance in the following tests, we created a packet header as depicted in Fig. 4. This packet header is used to calculate packet loss (using the seq number), the latency (using the timestamp) and the amount of data transmitted (using the data size). The key is used by the receiver to identify the beginning of a package.

To perform tests over the network, we also developed a loopback application that uses the same network implementation of sender and receiver. The loopback application does not have an Audio API and does not open the packets, but only receives a packet from the sender and sends it back to the receiver, as presented in Fig. 8.

![Figure 7. Medusa Application Protocol](#)

![Figure 8. RTT latency calculation with a loopback.](#)

Since the sender and receiver are in the same machine, the packet timestamp can be used to calculate latency per packet. The latency was logged to a text file for post-mortem jitter calculation. Because of this configuration, all measurements are round-trip time.

Some packet flags option were used in this implementation. In all cases, SO_SNDTIMEO was configured to value 6 (max value without blocking). Unlike UDP, DCCP will avoid packet loss and does not use retransmissions, TCP-like fast-recovery mechanisms are not available.

As in localhost connection, no packets were lost in this experiment. As in localhost connection, no packets were lost in this experiment. We call SCTP 1 the one-to-one implementation and SCTP 2 the one-to-many implementation.

We calculated latency as the average of all packages round trip time (1) and jitter as the latency deviation average (2).

3.1. LOCALHOST

The first test was made using a localhost connection. The localhost is a virtual interface, presented by the special IP address 127.0.0.1, used for system management. The preliminary test using localhost presents the system latency independently of network delay, i.e. the time that the system spends to pack the audio data, fragment it, copy the data to the kernel space, copy it back to user space and finally unpack the data.

The time to copy data from the audio API and to sample the audio was ignored in this measurement. Altogether, this time can be considered the JACK block processing time.

Table 2 presents the localhost measurements. There was no packet loss with any protocols in this connection.

3.1. CROSSOVER

A crossover cable is a special network cable with some transmission and reception wires inverted to directly connect one computer to another, bypassing any intervening device. This connection is an efficient way to connect two computers because it does not demand any network devices like hubs, routers or switches.

Table 3 presents the measurements in this connection. As in localhost connection, no packets were lost in this configuration.
SNDTIMEO was configured to 196 to avoid network overhead [1]. DCCP, like TCP, provides a single bidirectional uni-

3. Transport-layer fragmentation: SCTP keeps track of a fragmentation point based on the smallest MTU in the path to all peer addresses, and this smallest MTU size is used to split large user messages into smaller pieces that can be sent using a single IP packet [18]. Like TCP and unlike UDP, this fragmentation occurs at the transport-layer instead of the IP layer.

4. Fast retransmission: SCTP, like TCP, uses ACK to allow detection of packet losses. But instead of a normal ACK, it uses selective acknowledgement (SACK) and a mechanism that sends SACK mes-

5. Non-blocking head of the line: Unlike TCP, which is stream-based, SCTP is message-oriented or message-based like UDP. This allows SCTP to distinguish signaling messages at the transport layer, which are delivered to the application layer as soon as they ar-

SCTP, like TCP, also provides reliability, sequenc-

flow control, congestion control, and full-duplex data transfer, through message chunk building, check-

summing, packet validation and path management [18]. There are two types of SCTP socket implementations: one-to-one sockets and one-to-many sockets.

2.6. Datagram Congestion Control Protocol (DCCP)

DCCP is a transport protocol that combines TCP-friendly congestion control with unreliable datagram semantics, and a congestion control mechanism will be used [7].

2.7. Comparison Draft

To show the main differences between these protocols, a draft table is provided below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>UDP</th>
<th>TCP</th>
<th>SCTP</th>
<th>DCCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message-oriented</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Connection oriented</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Full duplex</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Reliable data transfer</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ordered data delivery</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unordered data delivery</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow control</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Congestion control</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multicasting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Broadcasting</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTU path discovery</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragmentation</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. IMPLEMENTATION

For the purpose of this protocol comparison, a front-end for multichannel audio communication was implemented using the Jack audio Server and the Linux socket API. Despite all these protocols being able to support full duplex communication, the implementation distinguishes the roles of sender and receiver.

The sender takes audio from the Jack server, resam-

bles it to the desired representation, packs it and puts the resulting packet into a ring buffer to be sent by another thread. This data will be fragmented into blocks of 1024 bytes before sending to the network. Because of this implementation, some fragmentation can happen in the mid-

dle of a packet. Thus, the receiver must identify the begin-

ning of a packet, and then separate header and data. The

header is logged for measurement and the data is then re-

sampled and put into a ring buffer to be consumed by the Audio API.

To measure the performance in the following tests, we

created a packet header as depicted in Fig. 4.

This protocol was developed for delay-sensitive appli-
cations that favor timeliness over reliability [7]. DCCP may use ACK messages, and does not interpret checksum errors as network congestion problems. Since DCCP does not use retransmissions, TCP-like fast-recovery mecha-

nisms are not available. Unlike UDP, DCCP will avoid congested collapse. It aims to add to UDP a minimum mechanism to support congestion control, such as reliable transmission of ACK information [7].

DCCP, like TCP, provides a single bidirectional uni-

Fig. 7. Medusa Application Protocol

Fig. 8. RTT latency calculation with a loopback.

Since the sender and receiver are in the same machine, the packet timestamp can be used to calculate latency per packet. The latency was logged to a text file for poste-

rior jitter calculation. Because of this configuration, all measurements results are round-trip time.

Some socket flags option were used in this implementa-

tion. In all cases, SO_SNDBUF was configured to 40ms and SO_PRIORITY to a value 6 (max value without root privileges). The former controls the maximum time-

out for sending packets before an error is reported, while the latter controls the priority ordering of outgoing pack-

ets in the send queue. The protocols TCP and SCTP also

used a NODELAY flag to disable the Nagle algorithm.

4. NETWORK PERFORMANCE MEASUREMENT ENVIRONMENT

For the performance test we used an HP Notebook Pavilion DV6 (Computer A) to send and receive data, and an ASUS netbook Eee PC Seashell Series (Computer B) as the loopback machine, both running Ubuntu 12.04. We configured these machines for real-time priority by setting kernel parameters

rtensio = 99 and nice = -19 in /etc/security/limits.d/audio.conf.

Streaming performance was measured under different

connections, such as localhost, crossover cable, through a switch, direct wireless connection and wireless connec-

tion with an access point (AP). With the exception of loc-

alhost, in all settings Computer A was sending, receiving and measuring, and Computer B was merely looping back the received data. In the localhost scenario both machines ran sender, loopback and receiver.

In all tests, Jack was configured for a 48 kHz sam-

pling rate and 32-bit sample size, with 512 samples per

1The Nagle algorithm is responsible for grouping together as much data as it can between ACK packets from the other end of the connection.

2Intel(R) Core(TM)2 Duo CPU T6600 2.20GHz 4GB Mem RAM

3AMD Athlon(tm) II N Processor 2GB Mem RAM

Table 1. Feature draft table

Table 3 presents the measurements in this connection.

In localhost connection, no packets were lost in this

connection. As in localhost connection, no packets were lost in this

connection.

Table 3 presents the measurements in this connection.

As in localhost connection, no packets were lost in this

connection.
4.1.3. Cable switch.

In this scenario, a router was used to connect the computers by cable. We used a D-Link DIR-615 switch with standard CAT5E network cables. The results for this connection are presented in Table 4. Again, there was no packet loss.

Table 4. Crossover connection

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>0.54ms (0.13ms)</td>
<td>0.62ms (0.15ms)</td>
</tr>
<tr>
<td>TCP</td>
<td>0.50ms (0.14ms)</td>
<td>0.67ms (0.16ms)</td>
</tr>
<tr>
<td>DCCP</td>
<td>0.50ms (0.14ms)</td>
<td>0.67ms (0.18ms)</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>0.65ms (0.17ms)</td>
<td>0.77ms (0.20ms)</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>0.60ms (0.17ms)</td>
<td>0.78ms (0.21ms)</td>
</tr>
</tbody>
</table>

4.1.4. Direct Wireless.

For the direct wireless connection, we created an 802.11 G wireless network (“Wi-Fi”) in computer A and connected computer B in this network. Thus, the Wi-Fi connection is direct between the two machines and there is no access point to connect them. The 802.11 G standard has a theoretical maximum transmission rate of 54 Mbps. The measurement of direct Wi-Fi connection is presented in Table 5.

Table 5. Direct wireless connection

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>18.76ms (27.98ms)</td>
<td>17 (17.3%)</td>
</tr>
<tr>
<td>TCP</td>
<td>20.71ms (33.41ms)</td>
<td>1 (0.01%)</td>
</tr>
<tr>
<td>DCCP</td>
<td>3.41ms (1.90ms)</td>
<td>0</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>21.00ms (34.85ms)</td>
<td>0</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>20.94ms (35.11ms)</td>
<td>0</td>
</tr>
</tbody>
</table>

4.1.5. Access Point Wireless.

Our last scenario is a domestic Wi-Fi connection. A D-Link DSL 2640T router was used as the access point to connect both computers. The result of these measurements is presented in Table 6.

Table 6. Wireless connection with AP

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>20.10ms (32.08ms)</td>
<td>30 (0.15%)</td>
</tr>
<tr>
<td>TCP</td>
<td>31.21ms (51.29ms)</td>
<td>0.001%</td>
</tr>
<tr>
<td>DCCP</td>
<td>5.02ms (3.16ms)</td>
<td>140.7%</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>27.09ms (44.40ms)</td>
<td>0</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>26.22ms (42.40ms)</td>
<td>0</td>
</tr>
</tbody>
</table>

4.2. Data analysis.

The previous data can be organized in two diagrams to give a better overview of each protocol performance. Fig. 9 presents the latency and Fig. 10 presents the packet loss for all protocols using 2 audio channels.

Figure 9. Latency draft - 2 channels.

Considering each localhost connection as one connection, in Fig. 9 we can note that the UDP protocol presented the lowest latency in 4 out of 6 connections. DCCP presented the lowest latency in Wi-Fi connections. TCP latency varied been the highest latency in direct Wi-Fi and the second lower using an AP. SCTP, both implementations, presented the highest latency in 4 out of 5 connections.

The congestion control mechanism present in DCCP seems to be more compatible with 802.11.11 rules and it led to better performance of this protocol in this scenario. When the transmission channel is occupied DCCP fails to send a package, the DCCP sender will discard data instead of retrying forever. This data discard appears as packet loss in Fig. 10.

Analyzing packet loss as presented in Fig. 10, we may observe that SCTP lost no packets in all connections, DCCP, as presented before, lost more packages in connections where it was faster. TCP also lost a few packets in Wi-Fi connections. The TCP packet loss can be understood as a result of sender timeout flag set for this protocol. In this case, the (send) function returns an error that is discarded by the sender application.

A summary of the data analysis can be done by dividing the protocols into groups: UDP and DCCP being faster and unreliable, TCP in a middle term and SCTP being slower but reliable.

Grouping the protocols this way, we have a protocol evaluation that allows us to choose between faster protocols or reliable protocols, according to the specific application scenario considered.

4.3. Discussion.

Besides the actual data analysis, the tests made allow some further discussion related to audio streaming performance measurements.

A first observation derived from our data regards the localhost connection, which can be used to calculate system latency in a single machine. Since computer B is less powerful than computer A and its results are inferior, we may observe that the machine configuration indeed influences network audio streaming performance. Because network communication is affected by latency in both nodes, we might expect that in no scenarios would we obtain better latency results than the ones with localhost connection on computer B.

A second observation regards the results on crossover connection as compared to cable switch connection. This comparison indicates the impact of adding other network devices in the middle of the network. To add other network devices means adding another set of receiving and sending buffers, and also introducing a new set of packet routing policies. For the specific equipment used, these buffers and routing policies increased the latency about 0.16 ms in switch connection compared to a crossover connection.

Despite the transport protocol choice, our data shows that the physical layer is also an important factor to be considered in a network music performance. For all protocols, the latency with a Wi-Fi connection with an AP is at least 20 times larger than with a crossover connection, considering the same transport protocols. This huge performance difference is explained in the physical connection. While network cables have different pair of wires to send and receive data, a Wi-Fi connection uses the same channel for both sending and receiving. Therefore the network channel is a shared resource and there are some race conditions to access it. For this reason, the Wi-Fi protocol has to have a workaround to avoid system starvation, which is included in the physical layer protocol (802.11 family). This includes a congestion control mechanism to avoid channel flooding by one sender, resulting in a better shared environment.

Since UDP and DCCP have no ACK or packet loss recovery, these protocols have less communication overhead. Less communication overhead implies a faster transfer rate but also more packet losses.

5. CONCLUSION.

This paper presented a group of transport protocols available to implement real-time connections for audio transmission in local computer networks. We chose to focus on these transport protocols because they run in kernel space, are present in most commonly-used operating systems and do not need special privileges to be deployed. Because of our focus on the transmission part of the streaming problem, this paper did not consider a plethora of post-processing techniques that would be available at the receiving end, such as buffering or glitch removal for instance.

We presented each protocol’s features from a theoretical point of view, to assess their performance measurements in different connection conditions using the same audio networking tool. Measurements were made using five different connection types: localhost, crossover, switch, direct wireless and wireless with access point. These connections are commonly-employed means to connect computers in a local network.

The results of these measurements showed that the protocol choice may emphasize the connection speed or the stream integrity, according to the application’s needs. A protocol that makes no confirmation of packet arrival will be faster and lose more data, whereas a reliable protocol will try to ensure stream integrity at the cost of increased latency. Thus, we have sorted the investigated protocols between fastest and most unreliable to slowest and most reliable, presenting alternatives for different network music scenarios.

Besides, the experiments also endorse the importance of having different protocol implementations and offering
### 4.1.3. Cable switch

In this scenario, a router was used to connect the computers by cable. We used a D-Link DIR-615 switch with standard CAT5E network cables. The results for this connection are presented in Table 4. Again, there was no packet loss.

**Table 4. Crossover connection**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>0.54ms 0.11ms</td>
<td>0.62ms 0.21ms</td>
</tr>
<tr>
<td>TCP</td>
<td>0.58ms 0.14ms</td>
<td>0.69ms 0.15ms</td>
</tr>
<tr>
<td>DCCP</td>
<td>0.60ms 0.14ms</td>
<td>0.67ms 0.20ms</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>0.65ms 0.17ms</td>
<td>0.77ms 0.19ms</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>0.60ms 0.17ms</td>
<td>0.78ms 0.21ms</td>
</tr>
</tbody>
</table>

### 4.1.4. Direct Wireless

For the direct wireless connection, we created an 802.11 G wireless network (“Wi-Fi”) in computer A and connected computer B in this network. Thus, the Wi-Fi connection is direct between the two machines and there is no access points to connect them. The 802.11 G standard has a theoretical maximum transmission rate of 54 Mb/s. The measurement of direct Wi-Fi connection is presented in Table 5.

**Table 5. Direct wireless connection**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>14.75ms 17.67ms</td>
<td>137 (1.37%)</td>
</tr>
<tr>
<td>TCP</td>
<td>16.76ms 20.52ms</td>
<td>0</td>
</tr>
<tr>
<td>DCCP</td>
<td>15.65ms 18.58ms</td>
<td>552 (5.52%)</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>26.39ms 35.64ms</td>
<td>0</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>26.56ms 37.33ms</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 6. Wireless connection with AP**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Latency Jitter</th>
<th>Latency Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>19.99ms 24.48ms</td>
<td>324 (4.63%)</td>
</tr>
<tr>
<td>TCP</td>
<td>19.98ms 24.45ms</td>
<td>31 (4.55%)</td>
</tr>
<tr>
<td>DCCP</td>
<td>14.70ms 13.26ms</td>
<td>1413 (7.06%)</td>
</tr>
<tr>
<td>SCTP 1</td>
<td>67.49ms 106.79ms</td>
<td>0</td>
</tr>
<tr>
<td>SCTP 2</td>
<td>72.06ms 166.40ms</td>
<td>0</td>
</tr>
</tbody>
</table>

### 4.2. Data analysis

The previous data can be organized in two diagrams to give a better overview of each protocol performance. Fig. 9 presents the latency and Fig. 10 presents the packet loss for all protocols using 2 audio channels.

**Figure 9. Latency draft - 2 channels.**

**Figure 10. Packet loss draft - 2 channels.**

Considering each localhost connection as one connection, in Fig. 9 we can note that the UDP protocol presented the lowest latency in 4 out of 6 connections. DCCP presented the lowest latency in Wi-Fi connections. TCP latency varied between the highest latency in direct Wi-Fi and the second lower using an AP. SCTP, both implementa-

tions, presented the highest latency in 4 out of 5 connections.

The congestion control mechanism present in DCCP seems to be more compatible with 802.11 rules and it led to better performance of this protocol in this scenario. Since when the transmission channel is occupied DCCP fails to send a package, the DCCP sender will discard data instead of retrying forever. This data discard appears as packet loss in Fig. 10.

Analyzing packet loss as presented in Fig. 10, we may observe that SCTP lost no packets in all connections. DCCP, as presented before, lost more packages in connections where it was faster. TCP also lost a few packets in Wi-Fi connections. The TCP packet loss can be understood as a result of sender timeout flag set for this protocol. In this case, the send() function returns an error that is discarded by the sender application.

A summary of the data analysis can be done by dividing the protocols into groups: UDP and DCCP being faster and unreliable, TCP in a middle term and SCTP being slower but reliable.

Grouping the protocols this way, we have a protocol evaluation that allows us to choose between faster protocols or reliable protocols, according to the specific application scenario considered.

### 4.3. Discussion

Besides the actual data analysis, the tests made allow some further discussion related to audio streaming performance measurements.

A first observation derived from our data regards the localhost connection, which can be used to calculate system latency in a single machine. Once computer B is less powerful than computer A and its results are inferior, we may observe that the machine configuration indeed influences network audio streaming performance. Because network communication is affected by latency in both nodes, we might expect that in no scenarios would we obtain better latency results than the ones with localhost connection on computer B.

A second observation regards the results on crossover connection as compared to cable switch connection. This comparison indicates the impact of adding other network devices in the middle of the network. To add other network devices means adding another set of receiving and sending buffers, and also introducing a new set of packet routing policies. For the specific equipment used, these buffers and routing policies increased the latency about 0.16 ms in switch connection compared to a crossover connection.

Despite the transport protocol choice, our data shows that the physical layer is also an important factor to be considered in a network music performance. For all protocols, the latency with a Wi-Fi connection with an AP is at least 20 times larger than with a crossover connection, considering the same transport protocols. This huge performance difference is explained in the physical connection. While network cables have different pair of wires to send and receive data, a Wi-Fi connection uses the same channel for both sending and receiving data. Therefore the network channel is a shared resource and there are some race conditions to access it. For this reason, the Wi-Fi protocol has to have a workaround to avoid system starvation, which is included in the physical layer protocol (802.11 family). This includes a congestion control mechanism to avoid channel flooding by one sender, resulting in a better shared environment.

Since UDP and DCCP have no ACK or packet loss recovery, these protocols have less communication overhead. Less communication overhead implies a faster transfer rate but also more packet losses.

### 5. Conclusion

This paper presented a group of transport protocols available to implement real-time communications for audio transmission in local computer networks. We chose to focus on these transport protocols because they run in kernel space, are present in most commonly-used operating systems and do not need special privileges to be deployed. Because of our focus on the transmission part of the streaming problem, this paper did not consider a plethora of post-processing techniques that would be available at the receiving end, such as buffering or glitch removal for instance.

We presented each protocol’s features from a theoretical point of view, to assess their performance measurements in different connection conditions using the same audio networking tool. Measurements were made using five different connection types: localhost, crossover, switch, direct wireless and wireless with access point. These connections are commonly-employed means to connect computers in a local network.

The results of these measurements showed that the protocol choice may emphasize the connection speed or the stream integrity, according to the application’s needs. A protocol that makes no confirmation of packet arrival is more flexible, but also more vulnerable to packet loss.

Besides the experiments also endorse the importance of having different protocol implementations and offering
the user a choice depending on his/her connection goals. One unforeseen result of this experiment is the observation that DCCP can be even faster than UDP in Wi-Fi connections, for example.

As observed in our results, other factors besides the transport protocol choice can influence latency, such as the machine setup, the number of network devices and the physical connection. Since repeating these tests is feasible in a short period of time, it would be interesting to add them as a performance measurement feature in a network music tool. With this feature users would be able to test their given connections and to choose the best protocol for the specific scenario at hand.

As future work we intend to test and compare these protocols over Internet connections.

6. ACKNOWLEDGMENTS

A special thanks to Beraldo Leal, Daniel Batista and Stephen Sinclair who helped with ideas, feedback and reviews. The authors would like also to thank the support of the funding agencies CNPq (grant no. 585955/2009-5), FAPESP (grant no. 2008/08632-8) and CAPES (grant no. BEX 1194/12-7).

7. REFERENCES


ANIMATED MUSIC NOTATION ON THE IPAD

(Or: Music stands just weren’t designed to support laptops.)

Aaron Wyatt and Cat Hope

School of Music
Western Australian Academy of Performing Arts

INTRODUCTION

Experimentation with graphic scores has enabled composers to free themselves from the constraints of traditional music notation. Avant garde composers in 1950s America and Europe began to develop an increasing array of new symbols to supplement the music that appeared on the stave, while those from the more experimental New York school often did away with traditional notation entirely, leaving performers to interpret a vast range of geometric shapes and patterns. But in spite of these developments, the essentially static nature of the printed score remained. With the advent of the micro computing era, however, this limitation could be trivially overcome, and graphic scores can now be set in motion in ways that are entirely predetermined or generated in real time. Recent developments in tablet computing have also meant that, for the first time, hardware capable of rendering such graphics now exists in a form factor that is portable and fits comfortably on any music stand. [8]

Decibel New Music Ensemble has been performing from such animated graphic scores for the past three years, with many of the works being written by the group’s members, commissioned and extant works being adapted to their ongoing development of a ‘score player’ for music featuring predominantly graphic scores. Initially, laptops were used for the task, with the scores being set in motion using networked Max/MSP patches created by ensemble members Lindsay Vickyery and Stuart James. It became apparent as time went on (and as my laptop went hurtling into the future) that there were limitations to this setup, and so the group embarked on a project to bring these scores to the iPad by developing their own software in house. The result is a flexible and modular application that aids not just the performance of these works, but their rehearsal and distribution as well. This app is the Decibel ScorePlayer. [6]

WHY ANIMATE?

Scores that ‘move’ on a computer screen provide opportunities for composers to engage in new paradigms of music creation. [15][16] They also enable new ways for composers to engage with such works. [11] Networking moving scores enables a level of synchronisation between performers that would be almost impossible otherwise, especially from open scores where proportion is key to the realisation of the work. [5] Music that requires a timeline (i.e. a stopwatch or other time measurement that does not involve tempo or bar lines) is facilitated, as are non-linear forms, animated scores and any score that requires real time generation. This can be realised by scrolling real time ordering and rendering of material, random processes, interactive processes and a range of animations.

Two extreme scores that lend themselves to such animation are Small Worlds (2004) by Werner Dafeldecker and In the Cut (2010) by Cat Hope. Small worlds is an improvised work for sextet that divides the players into two trios. Each trio has one person leading their particular improvisation, and as the work progresses, the membership of these two groups changes, with players swapping from one to the other and with new group leaders being appointed periodically. In the score, each player is represented by a labeled line that moves between the two trios, with a timeline indicating when transitions should occur. [4] By setting this in motion across a playground, it becomes much easier for the players to see how far through any given section or transition they are without the need to check an external timepiece. This type of score is referred to in the app (and in this paper) as a scrolling score.

Figure 1. Excerpt from Small Worlds (2004), by Werner Dafeldecker

In the Cut is also suited to being presented as a scrolling score, and it benefits particularly from the level of synchronisation offered by having the score players networked together. The work consists largely of long, droning sounds and glissandi, which would be difficult to represent with traditional notation. Because of this, the score has only ever existed in animated form, with the original version implemented in Max. As can be seen in the excerpt below, a line for each instrument represents pitch, proportionally over time. Displayed in this way, the performers are easily able to see when and how rapidly they should be altering their pitch, as their line moves across the playground. They can also see how their pitch should relate to that of the other instruments in the ensemble, as well as being able to quickly see and judge where entries or cut-offs should coincide. (Significant relationships between parts are marked on the score with a dotted vertical line.) The networked nature of the player