1996

Multi-pass Transmission Policy: An Effective method of transmitting large multimedia objects in the wide-area network

Sheng-Yih Wang

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
96-057

MULTI-PASS TRANSMISSION POLICY: AN EFFECTIVE METHOD OF TRANSMITTING LARGE MULTIMEDIA OBJECTS IN THE WIDE-AREA NETWORK

Sheng-Yih Wang
Bharat Bhargava

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR 96-057
September 1996
(Revised April 1997)

Multi-pass Transmission Policy: An effective method of transmitting large multimedia objects in the wide-area network *

Sheng-Yih Wang and *Bharat Bhargava*

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
E-mail: {swang,bb}@cs.purdue.edu

Technical Report CSD 96-057 (Revised April 7, 1997)

April 20, 1997

Abstract

Multimedia objects such as audio, video and images are usually very large in size and used in time-critical applications. The traditional method of transmitting these large objects over a WAN is to use TCP because of the high loss rate of IP datagrams over a WAN. In this paper, we propose a new method called Multi-pass Transmission Policy (MpTP). The basic idea of MpTP consists of three things: sending small packets, selective retransmission requested by the receiver, and multi-pass transmission. MpTP function by sending small packets and packets not received on the first pass are retransmitted on the second pass and so on till the required reliability is reached. We conduct experiments on both MpTP and TCP to compare the efficiency of the two protocols. Experimental results indicate that when transmitting large objects, MpTP can, on average, be as much as 4 times faster than TCP. In addition to the efficiency, MpTP can support multi-resolution, degrees of reliable transmission and

*This research is partly supported by a grant from NSF under NCR-9405931

real-time transmission. We also conduct experiments to decide the optimal datagram size for MpTP. Based on the experimental result we provide some guidelines for achieving optimal performance using MpTP.

1 Introduction

Multimedia objects such as audio, video and image are usually very large in size. For example, a typical MPEG movie which lasts 15 to 25 seconds has a size of 1 Mbytes. Similarly, an audio file whose playing time is 3.4 seconds can be as large as 27 Kbytes. The traditional way of transmitting these large objects over a WAN is to use TCP [Pos81] because the high loss rate of IP datagrams over a WAN. The loss rate of the WAN environment can easily be over 50%. When loss rate is high, however, TCP incorporates retransmission and congestion control algorithms which will slow down the transmission process. For multimedia data, it is not appropriate to apply TCP-style retransmission policies because the time-critical nature of the data. Multimedia data transmission is different in many respect from traditional data transmission. One significant difference is that multimedia data does not always require 100% reliability. Many researches exploit this fact and use UDP/IP [Pos80] to transmit large multimedia objects over the network. However, UDP is only successfully used in LAN environments because the loss rate in today's LAN is usually less than 1%. When used in a WAN, UDP can not provide sufficient reliability for multimedia data in most cases. There are some researches which try to overcome the problem of data loss in multimedia data transmission. Usually these researches fall into the following two categories:

- One approach try to ignore the lost of the data and recovery as much information as possible from the data received. Some pre-processings on the data are required to minimize the effect of lost packets on the data. This approach, generally called "open-loop method", includes *Forward Error Correction (FEC)* [KV89], *Channel Coding* [GV93] and other similar techniques.

- The other approach is to provide some error control mechanisms which try to reduce the loss rate to some acceptable degree. For example, new-generation protocols such as XTP [Str95] provide some error control mechanisms which can be used to provide different levels of reliabilities. However, these protocols didn't specify the details of how to efficiently transmit and retransmit the data. It is left for the data sender and receiver to decide.

In this paper, we propose a new method called Multi-pass Transmission Policy (MpTP). MpTP is a policy of how to decompose the large data object into small transmission units and how to retransmit data if it is lost. Some preliminary experimental results indicate that this method is a very effective way of transmitting large objects over a WAN.

The following sections of the paper are organized as follows. Section 2 discuss the issues related to multimedia data transmission over a WAN. Section 3 discuss the basic concept and implementations of MpTP. In section 4 we describe the result of some experiments on MpTP. Section 5 describe the features of MpTP. In section 6 we will discuss some related researches. In section 7 we conclude our work with discussions and future works.

2 Issues in Multimedia Data Transmission over a WAN

- **Network Resources:** The most important problem in multimedia data transmission over a WAN is insufficient bandwidth. The maximal bandwidth of a WAN is usually less than 50 KBytes/sec which are too small for typical multimedia applications such as video-on-demand or video-conferencing. One technique to tackle the problem of insufficient bandwidth is to incorporate adaptability into the system. The changes of the network infrastructure such as ATM network and the introduction of the next-generation IP (IPv6) are also promising solutions.

- **Quality of Service:** The time-critical nature of multimedia data transmission is another big problem in multimedia data transmission over a WAN. Current network technology in WAN tend to produce very long delay (for example, 1000 ms) data packets. These kinds of network traffics are not suitable for interactive multimedia applications. Current WANs can only generate data traffics which can not guarantee a fixed delay for each individual packets. The delay jitter, which is defined to be the variations of the network delay for each packets, is a very important parameter to continuous media applications. To guarantee the smoothness when playing back a continuous media streams, some mechanism of jitter control is required.

- **Transport Protocols:** Since multimedia data transmissions consume a lot of network bandwidths, rate control is very important to keep the network functions well. Current protocols for data transmission such as TCP don't have built-in rate control mechanism which is essential to multimedia data transmissions. Error control mechanism is also a very important part of a multimedia data transport protocol. Since multimedia data does not always require 100% reliability but usually constraints by the time, it need error control mechanisms which are different from the TCP-style (100% reliable) or UDP-style (no guarantee or upper bounds on error at all).

- **Heterogeneity** A WAN is usually a collection of heterogeneous nodes. The heterogeneous factors among systems can be: different networks (from high-speed network to slow modem), different platforms (different CPU power and operating systems), different video display (24-bit full-color, 8 bit-color or monochrome), different audio devices (speech quality 8 KHz sampling rate or CD-quality 44.1 KHz sampling rate), or different quality of service each application require. Even with the same system, different nature of the data can pose different problems. For example, the characteristic of audio data is completely different for speech and music. An algorithm optimized for speech data can run very poorly on music data, and vice versa. Similarly video data from video conferencing and motion pictures are inherently different. A multimedia system have to deal with all these heterogeneity issues and take into consideration that sites in a WAN come with different capabilities and requirements. The sender and the receiver of multimedia data transmission should have some mechanism to negotiate between parameters such as sending rate, payload types, etc to achieve optimal performance.

3 Multi-pass Transmission Policy (MpTP)

3.1 Basic ideas of MpTP

The basic idea of Multi-pass Transmission Policy (MpTP) is: instead of transmitting large chunks of data over the WAN, the algorithm tries to send small packets (usually less than or equal to 4 Kbytes) and retransmit the data only after the whole object was transmitted and the retransmission is performed only by the request of the receiver. The receiver send an error report as a feedback to the sender, and the sender retransmit only those packets reported as lost by the receiver. It should be noted that even if the loss rate is quite high, MpTP works very well in WAN environment. For example, if the upper bound of the loss rate is 30%, then the first round of data transmission will leave behind 30% of the data, and the second round of data transmission will leave only 9% behind, and the third round will leave behind only 2.7% of data.

3.2 A formal model of MpTP

To formally model the behavior of MpTP, the following parameters are needed:

- S: Size of object to be transmitted (KBytes).
- R: Sending rate (KB/sec).
- C: Chunk size (KBytes).
- E: Reliability. $0 \leq E \leq 1$. For example, $E = 0.9$ means that the reliability is 90%.
- **F**: Random variable which describe the distribution of the network delay from sender to receiver.
- **B**: Random variable which describe the distribution of the network delay from receiver to sender.
- **L**: Random variable which describe the distribution of the network loss rate for each pass.

From the above definitions we can calculate the number $t = \frac{C}{R}$ which is the interval between two successive packets sending out from the sender. Figure 1 illustrates the formal model of MpTP.

The total time required to transmit an object can be calculated as follows. For pass j,

$$\max_{i=1}^{N_j}((i-1)t + \mathbf{F}_{N_1+\ldots+N_{j-1}+i})$$

time is required to send all data from sender to receiver. Here $N_1 = \frac{S}{R}$, $N_j = \frac{S}{R} * L_1 * \ldots * L_{j-1}$ for $j > 1$. Between pass j and pass $j+1$, an error report which takes B_j time is sent from the receiver to the sender. Therefore the total transmission time $T(k)$ of a k-pass transmission is

$$T(k) = \sum_{j=1}^{k} \max_{i=1}^{N_j}((i-1)t + \mathbf{F}_{N_1+\ldots+N_{j-1}+i}) + \sum_{j=1}^{k-1} \mathbf{B}_j$$

We can derive some useful results from the above formula. First let's make observations on how many passes are required to achieve the desired reliability E. Let $\mathbf{L}^i = \mathbf{L}_1 \times \mathbf{L}_2 \cdots \mathbf{L}_i$. Let μ_i be the mean of $\mathbf{L}^i$, σ_i^2 be the variance of $\mathbf{L}^i$. According to *Chebyshev's inequality,*

$$P(|\mathbf{L}^i - \mu_i| \geq t) \leq \frac{\sigma_i^2}{t^2}$$

We can transform the above equation into

$$P(\mathbf{L}^i - \mu_i \geq t \quad \vee \quad \mathbf{L}^i - \mu_i \leq -t) \leq \frac{\sigma_i^2}{t^2}$$

or alternatively,

$$P(\mathbf{L}^i \geq \mu_i + t \quad \vee \quad \mathbf{L}^i \leq \mu_i - t) \leq \frac{\sigma_i^2}{t^2}$$

Let $\mu_i + t = 1 - E$. Then $t = 1 - E - \mu_i$, $\mu_i - t = 2\mu_i - 1 + E$. The above formula becomes

$$P(\mathbf{L}^i \geq 1 - E \quad \vee \quad \mathbf{L}^i \leq 2\mu_i - 1 + E) \leq \frac{\sigma_i^2}{(1 - E - \mu_i)^2}$$

or

$$P(\mathbf{L}^i \geq 1 - E) + P(\mathbf{L}^i \leq 2\mu_i - 1 + E) \leq \frac{\sigma_i^2}{(1 - E - \mu_i)^2}$$

Since it is difficult to simplify the above formula without further assumptions, let's consider some cases which are simple enough to calculate. Although the calculations below

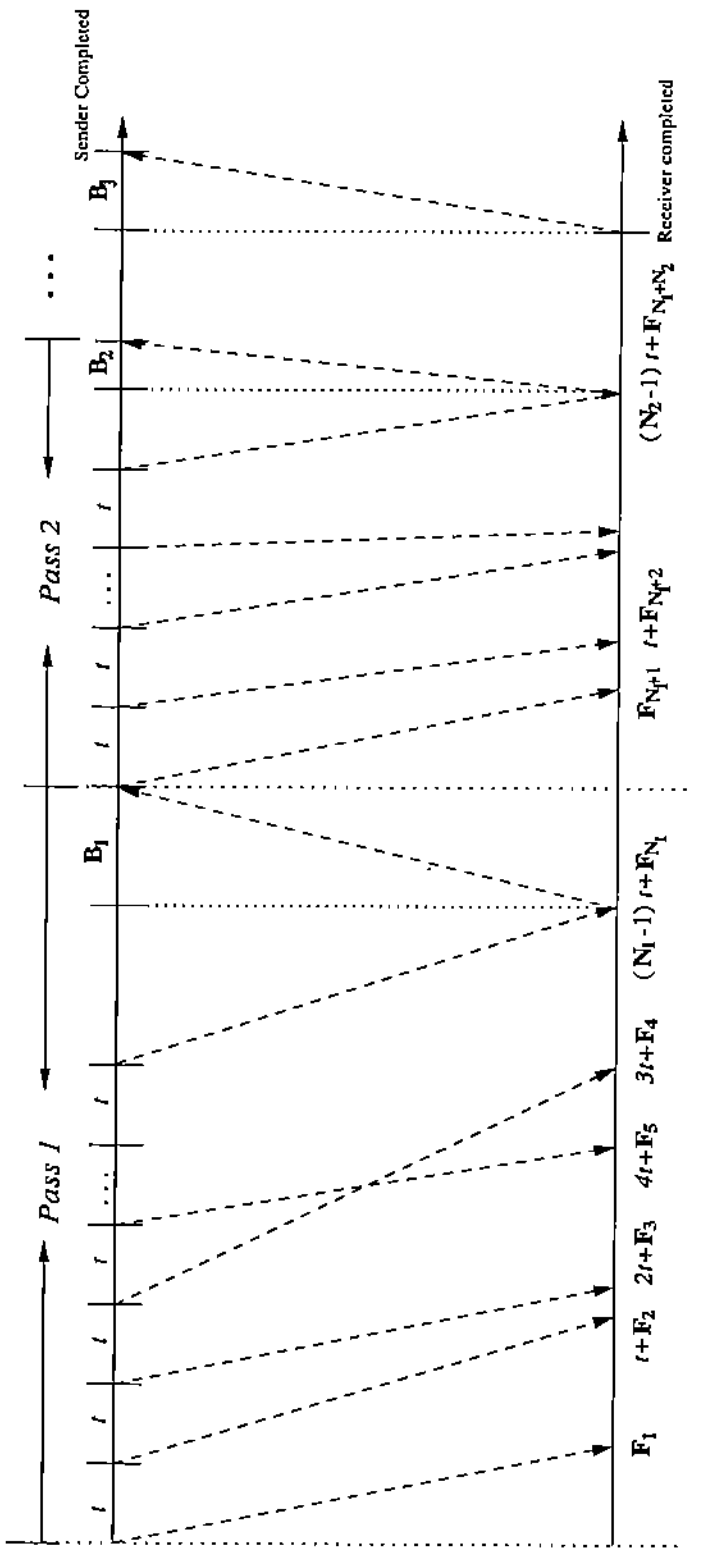

Figure 1: A Formal Model of MpTP

make some assumptions which may not be complex enough to match the real situations, they still offer some insights to the behavior of MpTP. The first simplification is to drop the term $P(\mathbf{L}^i \leq 2\mu_i - 1 + E)$ because a probability value is always positive. Then the formula becomes

$$P(\mathbf{L}^i \geq 1 - E) \leq \frac{\sigma_i^2}{(1 - E - \mu_i)^2}$$

We can also derive a inequality about $\mathbf{L}^i$ from *Markov's inequality.* The Markov's inequality said that

$$P(\mathbf{X} \geq a) \leq \frac{E(\mathbf{X})}{a}$$

for any non-negative random variable $\mathbf{X}$. Therefore

$$P(\mathbf{L}^i \geq 1 - E) \leq \frac{\mu_i}{1 - E}.$$

Below are some observations we can made from the above formulas:

- Let's assume that the loss rate $\mathbf{L}_i$ is uniformly distributed. That is, the p.d.f. of $\mathbf{L}_i$ is

  $$L_i(x_i) = 1 \qquad\qquad 0 \leq x_i \leq 1, i = 1, 2, \cdots, j$$

  Then the p.d.f. of $\mathbf{L}^i$ is [Spr79]

  $$L^i(y) = \frac{\ln(\frac{1}{y})^{(n-1)}}{(n-1)!} \qquad\qquad 0 < y \leq 1$$

  We can obtain the numerical results of the above two inequalities in this case. Figure 2 shows the graph of the probability $P(\mathbf{L}^i \geq 1 - E)$ v.s. the reliability parameter E. From the graph we can observe that although the the assumption about the loss rate distribution is very harsh, the probability of not achieving the required reliability in very few passes is still low. In practice, as we observed, the loss rate distribution is something similar to Gamma distribution (e.g. $f(x) = 100xe^{(-10x)}$) which are quite heavy-headed. The result will be even better if heavy-headed distributions are used in the calculations.

- Let's regress a bit and assume that the loss rate $\mathbf{L}_i$ is bounded from above by some constant L. We can calculate the number of passes required to achieve the reliability

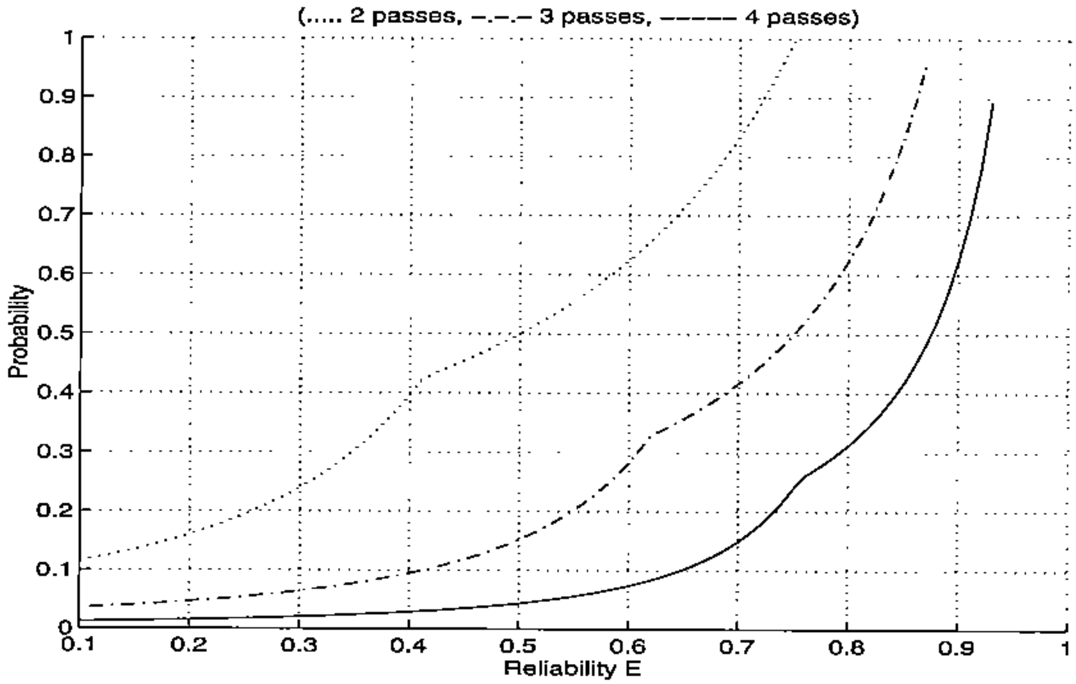

Figure 2: $P(\mathbf{L}^i \geq 1 - E)$ on different E and i

E. The condition to achieve the reliability E is $L^i \leq 1 - E$. Solve for i we have

$$i \geq \frac{\ln(1-E)}{\ln L}$$

A plot of the above formula is in figure 3. We can see that even though the loss rate is very high, the passes of retransmission required are still very few. For example, we can achieve 90% reliability for only 4 retransmissions at the loss rate upper bounded by 60%.

3.3 Implementation Considerations

MpTP is a general concept which can be implemented in many ways. It can be implemented in different network architectures such as IP, ATM or any other network architecture which supports unreliable packet transmission. Currently our implementation is based on UDP/IP [Pos80]. We implemented a simple mechanism of handshaking control for re-

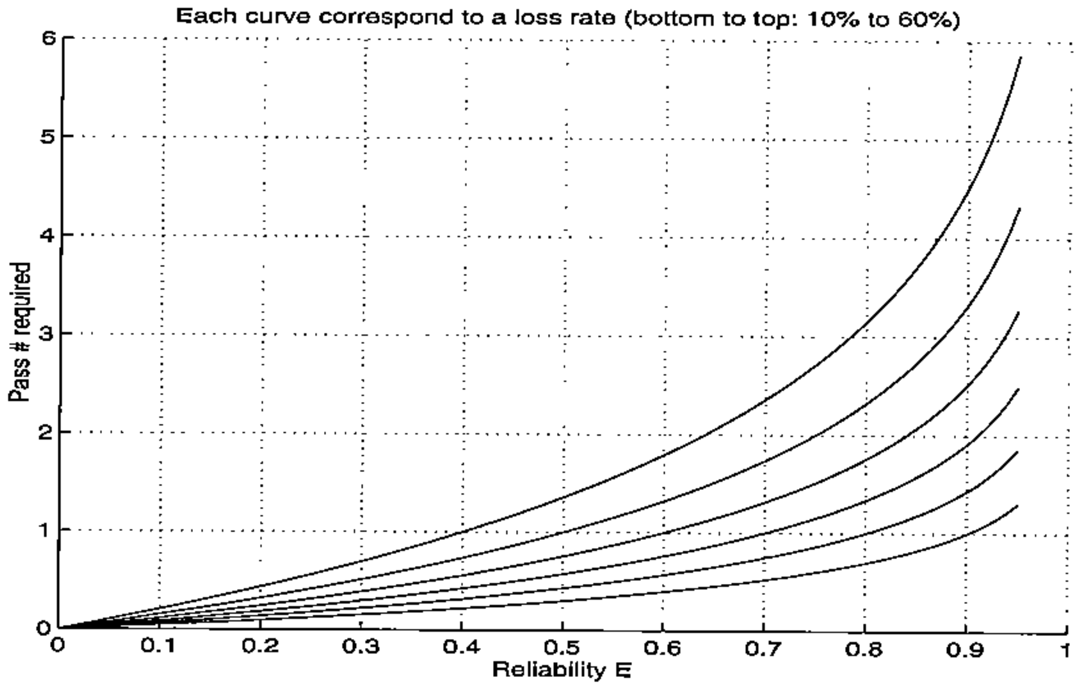

Figure 3: graph of $y(E, L) = \frac{\ln(1-E)}{\ln L}$ for some Es and$L$s

liable file transmission to demonstrate its effectiveness. The handshaking control works as follows (refer to Figure 4):

> The sender tries to establish a connection by sending a packet with information about the file it wants to send to the receiver. Upon the reception of this packet by the receiver, the receiver sends back an acknowledgment and a session is established. After receiving the acknowledgment, the sender begins to send identically-sized packets using UDP (the last packet may not be full-sized.). The sender also sends out a magic packet at the end of each data transmission round as an indication that the data transmission round is completed. The receiver receives the data sent by the sender and put it into an appropriate buffer. This buffer is determined according to its relative offset from the beginning of the file. (This information is carried in the data packet.) The receiver maintains a table of flags to keep a record on missing packets. At the end of the round of data transmission (as indicated by the reception of the magic packet),

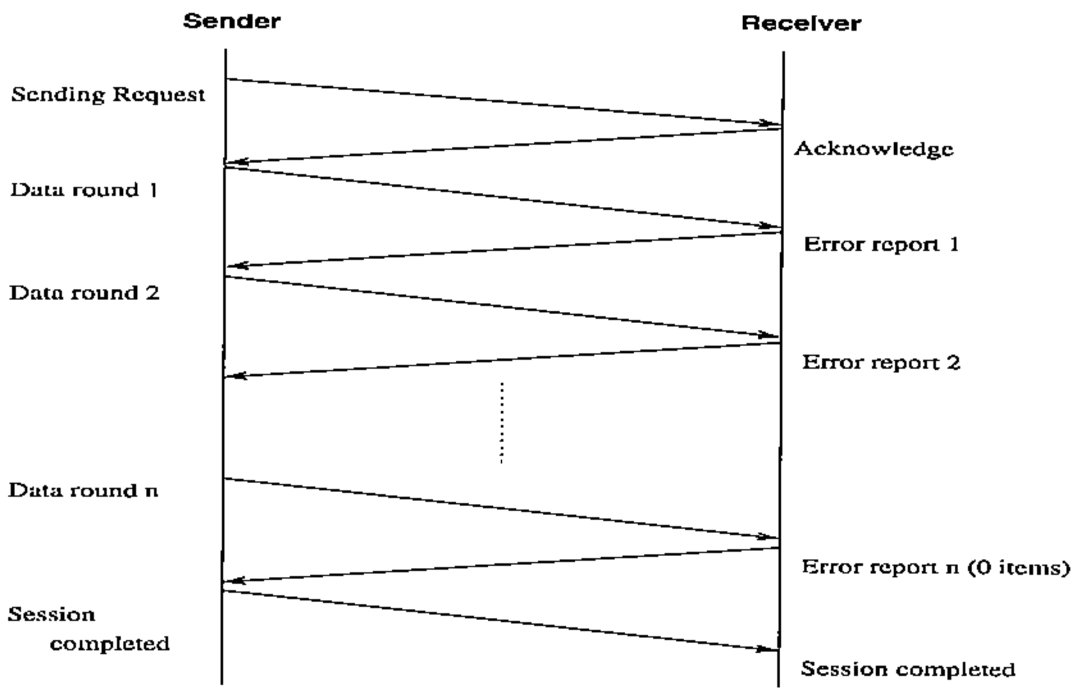

Figure 4: A simple handshaking control for reliable transmission

the receiver sends a message reporting the lost packets to the sender. If any packets were reported lost, the sender will then initiate a second round of data transmission, otherwise the transmission is completed. Of course it is possible that in the second round some of the retransmitted packets are again lost. If this is the case, there will be a third, forth, ... round of data transmission until all the data is received.

Note that the above scenario assumes that the magic (control) packets are not lost during transmission. In reality this may not be true. In our implementation we use retransmission to ensure the arrival of control packets.

Another possible implementation is to integrate MpTP into real-time transport protocol (RTP). In RTP specifications, a separate channel carrying control information is used to send RR (receiver report) packets to the sender. We can just use RR packets to report the lost packets, and let the protocol handle the control of session initialization and control packet transmitting.

MpTP can also be implemented on top of Xpress Transport Protocol (XTP) [Str95]. XTP separates the communication paradigm and error control policy to provide a set of mechanisms which can cover a wide range of transmission requirements. XTP provides a selective retransmission option which can be used to implement MpTP. However, there are some problems if MpTP is implemented on top of XTP. In section 6 we will discuss these problems in more detail.

4 Experiments

4.1 Experimental set up

- **Purpose of experiments**

  The purpose of the experiments is to validate the feasibility of the MpTP approach on multimedia data transmission.

- **Problem statement**

  We want to compare the performance of TCP and MpTP on multimedia data by transmitting files using both TCP and MpTP. We also want to experimentally decide a chunk size for MpTP sessions to reduce the loss rate of UDP packets and achieve optimal throughput.

- **Input parameters**

  In the following experiments, a set of satellite images from NASA are used as the benchmark. The size of the images files range from 6988 bytes to 486430 bytes. The remote site for the experiments is in Academia Sinica, Taiwan. The maximal bandwidth between Purdue University and Academia Sinica, Taiwan is 1.54 Mb/sec (T1 line). We conduct the experiments on two different sending rates (24 KB/sec and 32 KB/sec). For each sending rate, we vary the chunk size (1KB, 2KB, 4KB and 8KB) to study the effect of chunk size on loss rate.

- **Output parameters**

  We measure the packet loss rate for each different chunk size and different sending rate. Total transmission times for each different settings are also measured and compared to the numbers obtained using TCP. We also calculate the standard deviation of both MpTP and TCP sessions.

- **Methods used for experiments**

  To compare the performance of TCP and MpTP on multimedia data transmission, we transmit the image files using both TCP and MpTP and measures the transmission times for each session and standard deviations from the transmission times. We vary the chunk size of MpTP sessions for each sending rate to experimentally decide an chunk size which can reduce the loss rate of UDP packets and achieve best throughput.

  The TCP result is obtained by an FTP program which has been slightly modified to print out accurate transmission times. The MpTP result is obtained by a set of client/server programs which employ the handshaking algorithm described in section 3. The results below are the average of 15 sets of experiments. Two sets of the experiments are done during business hours, so the network is more congested and the performances of MpTP are much better than TCP.

4.2 Results

The results of our experiments are described as follows:

- **TCP v.s. MpTP**

  Figure 5 shows the results of both TCP and MpTP transmission times on NASA image files. It is observed that MpTP is much faster than TCP in transmitting large files and is a viable and effective method to transmit large objects over a WAN environment.

- **Chunk size and loss rate**

  Figure 6 and 7 shows the loss rate of sending NASA image files using UDP in 24 KBytes/sec and 32 KBytes/sec respectively. From the results it is clear that the chunk size has a definite influence on the loss rate. In both graphs, the loss rate is very high for 8K packets. Therefore sending large UDP packets over a WAN should

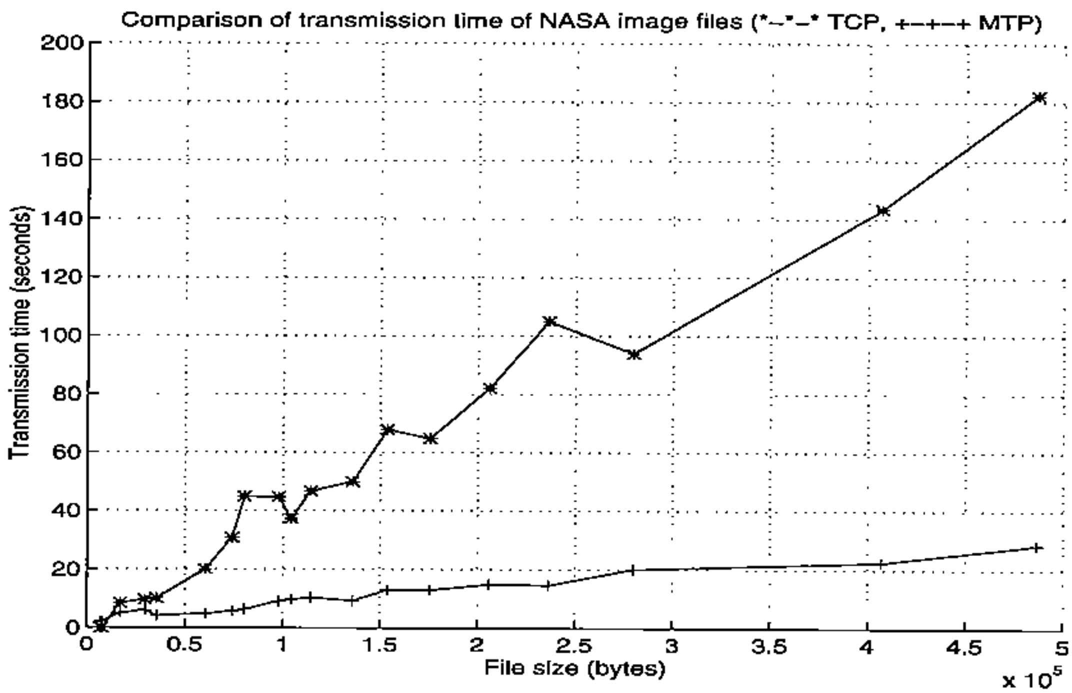

Figure 5: Comparison of transmission times of TCP and MpTP

be avoided. It is also clear that the loss rates are influenced by the sending rates. It can be seen clearly that the loss rates of 24 KBytes/sec data is much smaller than the loss rates of 32 KBytes/sec data. This is true no matter what the chunk size is. This result can be interpreted as an validation of UDP cooling method [LCB95] (in a generalized sense for WAN environment).

In addition to the loss rate comparison, we also compared the transmission times for different chunk sizes and different sending rates. Figures 8, 9, 10 and 11 compare the transmission times for sending rates at 24 KB/sec and 32 KB/sec and chunk sizes of 1, 2, 4 and 8 KBytes respectively. It is observed that sending data at a higher rate did not always improve the throughput. In fact, for the 8 KBytes chunk size case, the result is worse at the 32 KB/sec sending rate due to the high packet loss rate.

Figure 12 and 13 compare the transmission times of different chunk sizes at the sending rates of 24 KBytes/sec and 32 KBytes/sec respectively. It is observed that the difference of transmission times for chunk sizes 4 KBytes and 2 KBytes is not

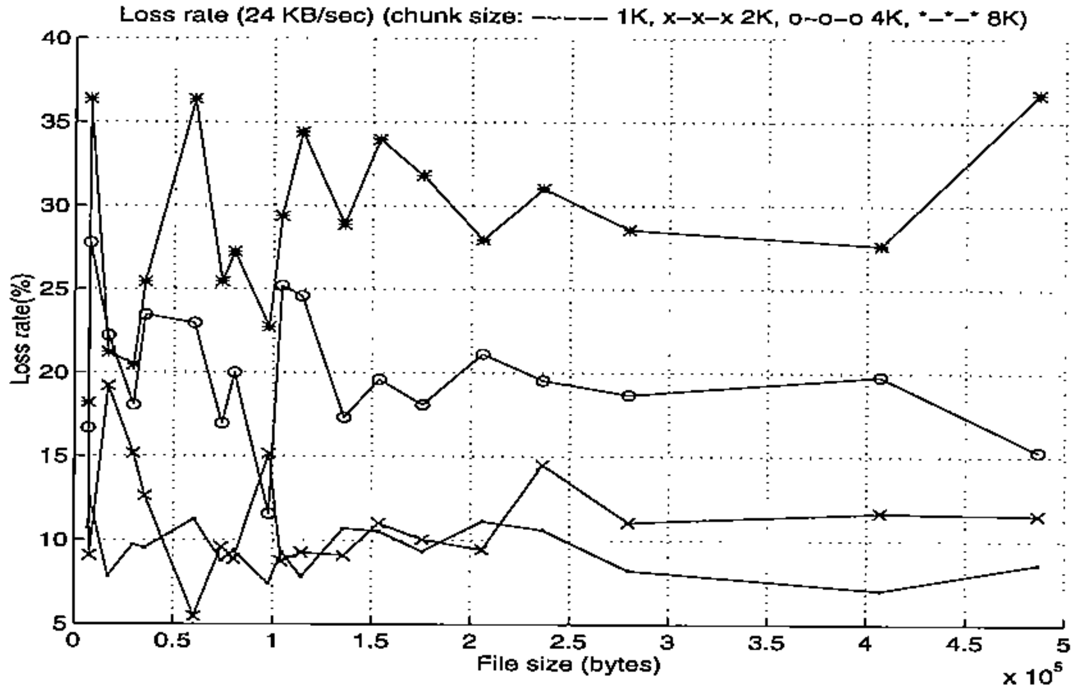

Figure 6: Loss rate of UDP packets when transmitted in 24KB/sec

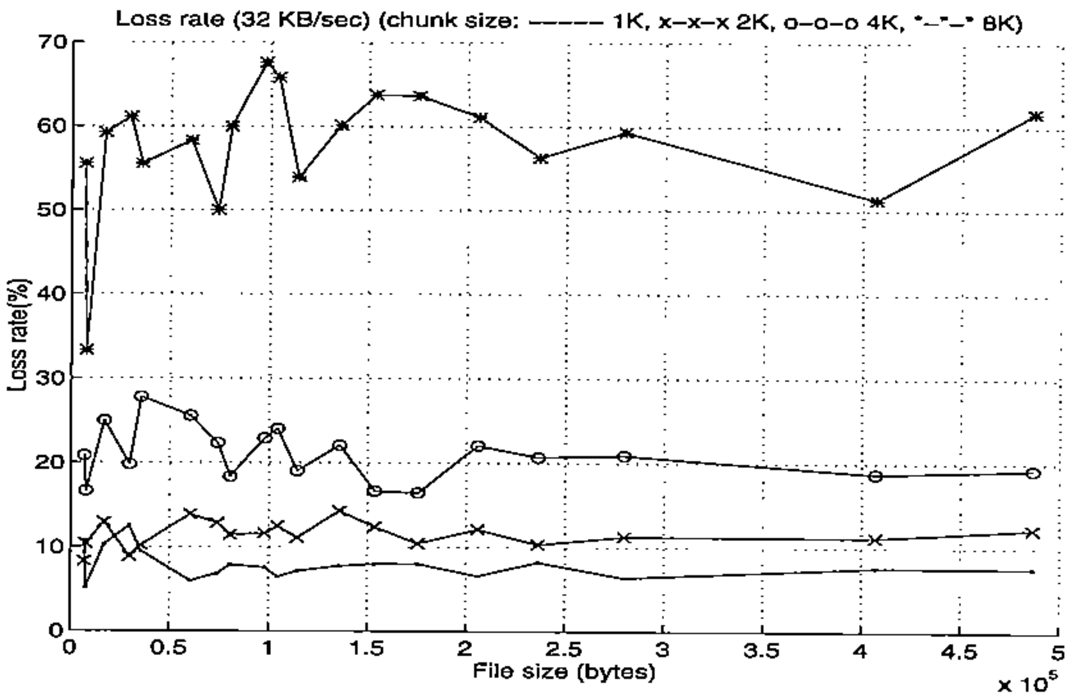

Figure 7: Loss rate of UDP packets when transmitted in 32KB/sec

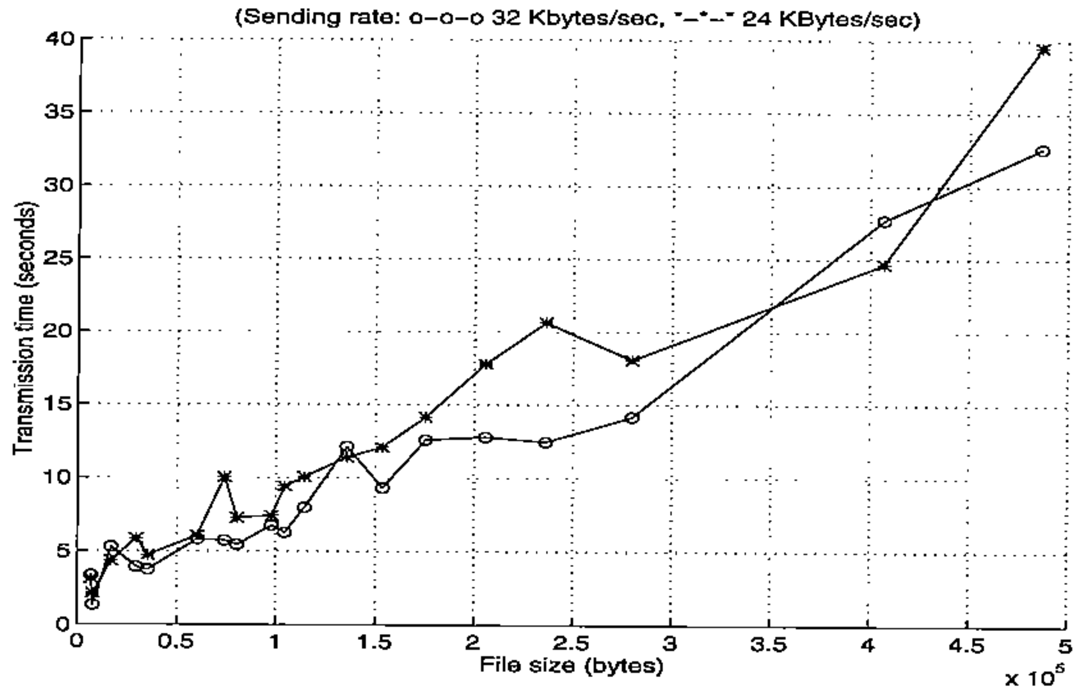

Figure 8: Transmission time of chunk size 1 Kbytes

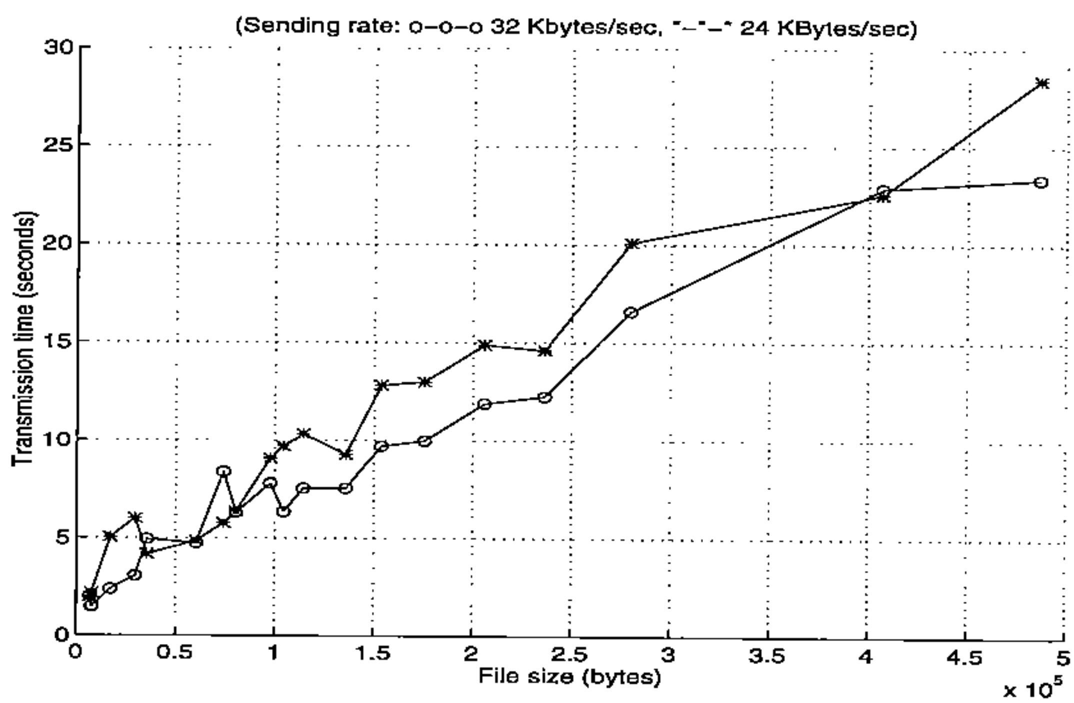

Figure 9: Transmission time of chunk size 2 Kbytes

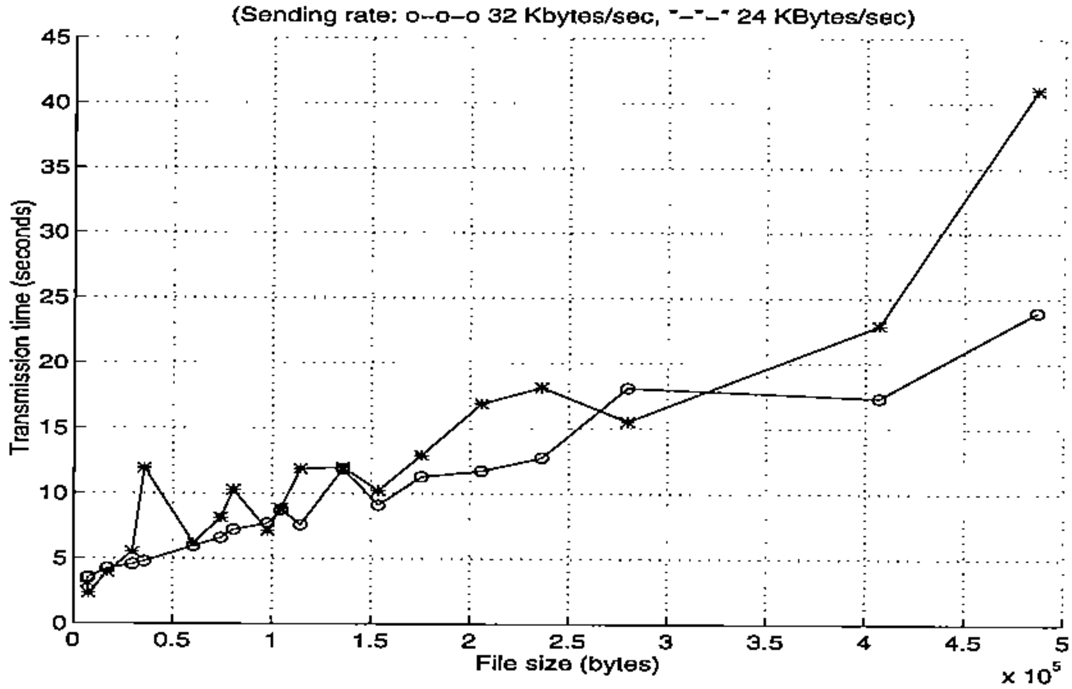

Figure 10: Transmission time of chunk size 4 Kbytes

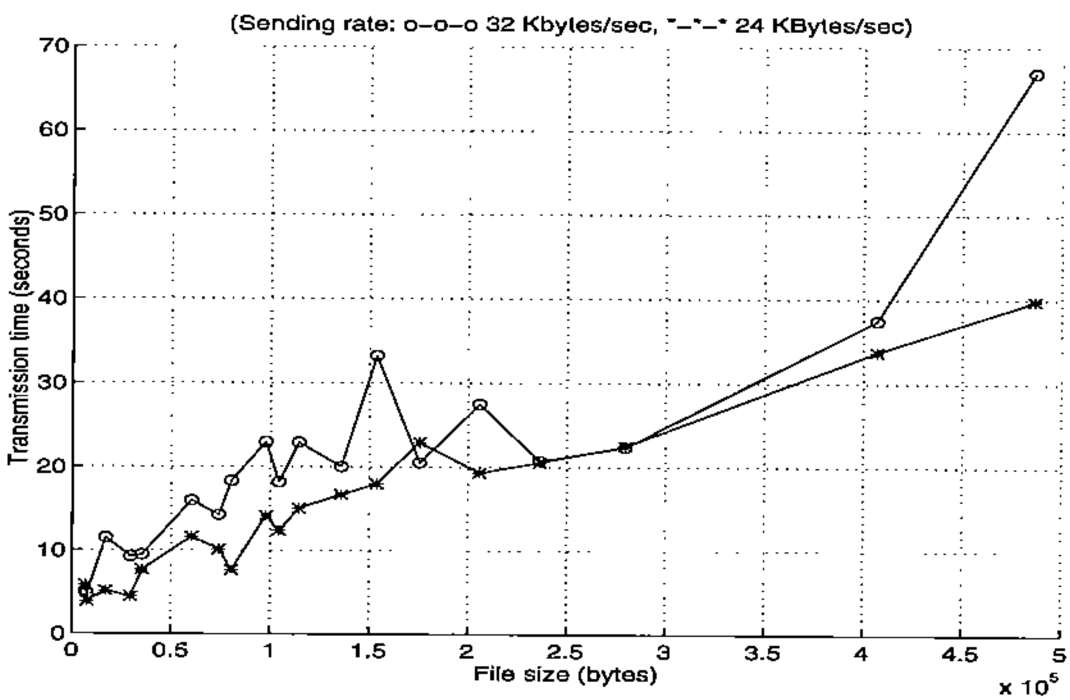

Figure 11: Transmission time of chunk size 8 Kbytes

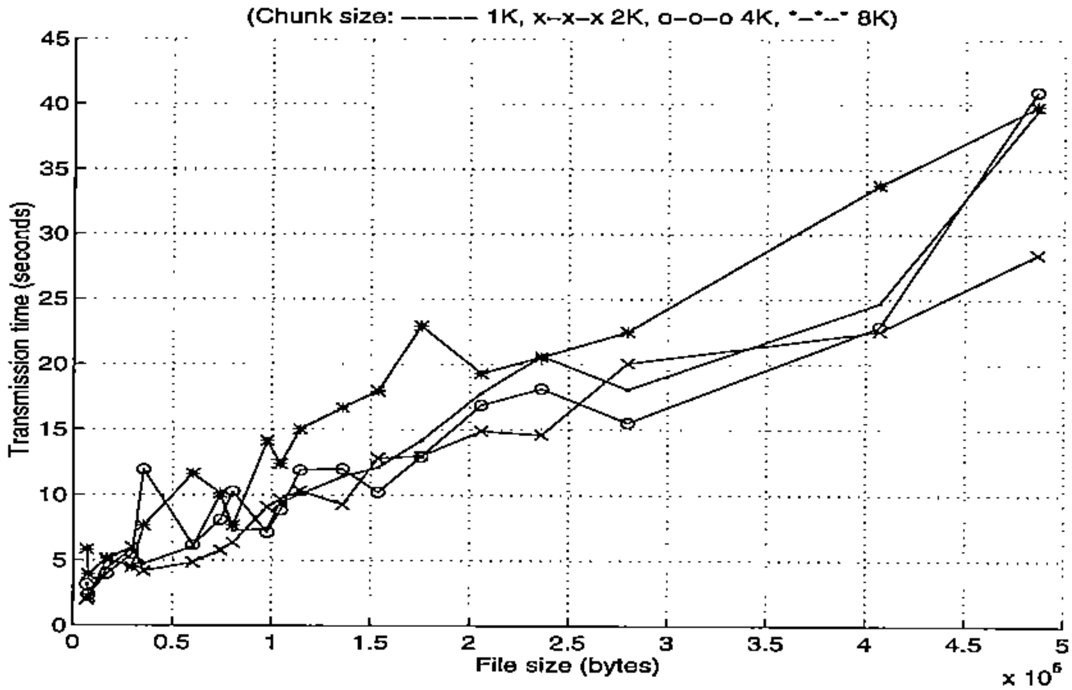

Figure 12: Transmission time of sending rate 24 KBytes/sec

significant. The transmission time of 4KB and 2KB are optimal among the four chunk sizes. The transmission time of 1KB chunk size is a little bit longer due to the overhead of sending small packets. The transmission time of 8KB chunk size is much longer due to the high packet loss rate. Therefore we suggest that 2KB or 4KB packets be used when transmitting data over the WAN environment.

5 Features of MpTP

- Experimental results indicate when transmitting large objects, MpTP can, on average, be as much as 4 times faster than TCP. When the network is congested, this advantage is even more stark and MpTP can be as much as 8.6 times faster than TCP. In section 7 we will discuss how MpTP achieves such a high performance.

- In contrast to the quick growth of TCP, the transmission times of MpTP grow slowly with file size. The file `eclipse1.gif` (486426 bytes) requires only 68.22 seconds to

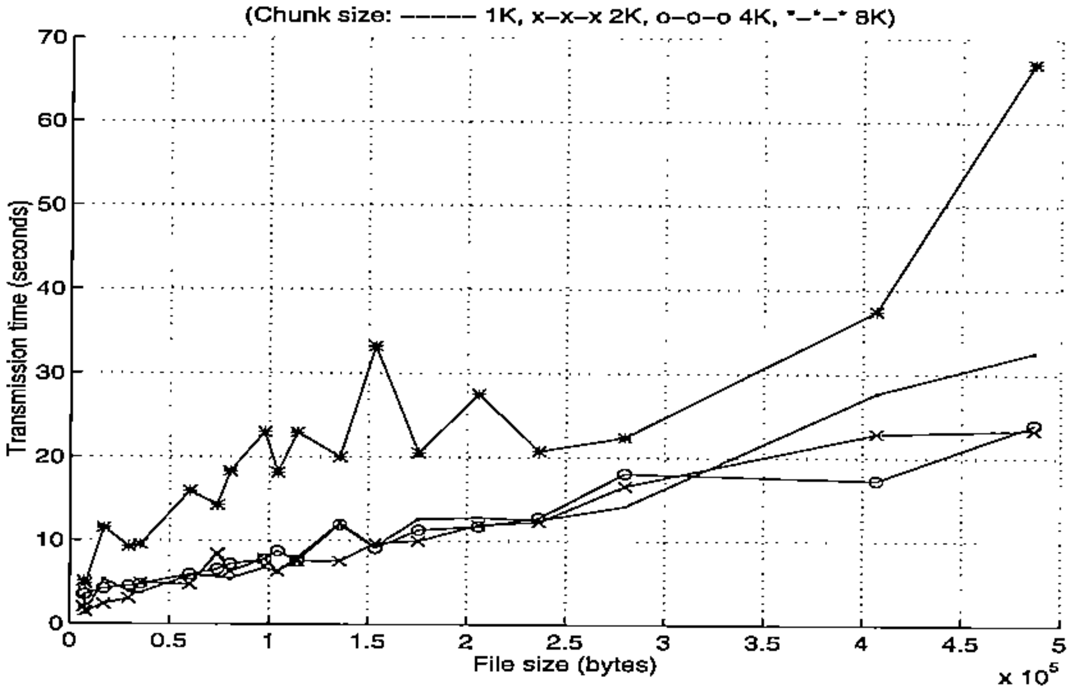

Figure 13: Transmission time of sending rate 32 KBytes/sec

transmit using MpTP but requires 590.42 seconds when the network is congested using TCP.

- MpTP supports multi-resolution data transmission. The sender can send course-grained data first, followed by fine-grained data. The receiver can choose to stop the session when the granularity of data received is sufficient.

- MpTP supports both reliable and unreliable transmission of data. Because the retransmissions are triggered by the receiver, the receiver decide whether retransmission is necessary or not.

- MpTP supports real-time transmission. The sender and receiver can negotiate in advance an upper bound of transmission rounds to meet the real-time requirement.

- MpTP can tolerate the mis-order delivery of packets. Any packet received is valid because the application knows exactly where the packets should be inserted into the buffer and how large the packet is. This will greatly reduce the need of retransmission.

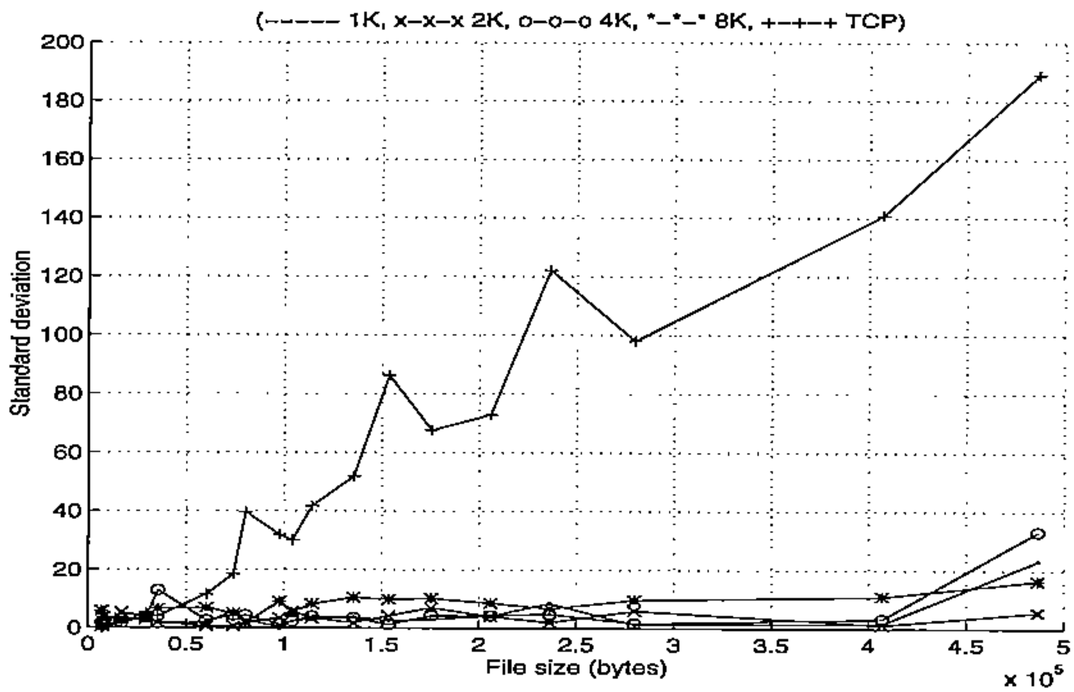

Figure 14: Standard deviation of MpTP and TCP at 24 KB/sec

- For all cases, it is clear that the standard deviations of MpTP are much smaller than of TCP. Figures 14 and 15 show the standard deviation of MpTP and TCP. Therefore, it is easier to predict the behavior of MpTP sessions. Although with MpTP there are some lost data packets, the receiver gets most of the data in the first few rounds of transmission and is able to proceed. The lost packets can be worried about later.

- The object's size must be known before transmission. MpTP, therefore cannot be used in live video or audio streams. A technique called *segmentation* can be used to partially resolve this problem.

- Because the sender has to maintain a status for each receiver, the sender's resource requirement increases linearly with the number of receivers. This problem also exists in real-time transport protocol (RTP) [SCFJ96] since for each participating cite in a RTP session, the information about every other participating sites must be kept.

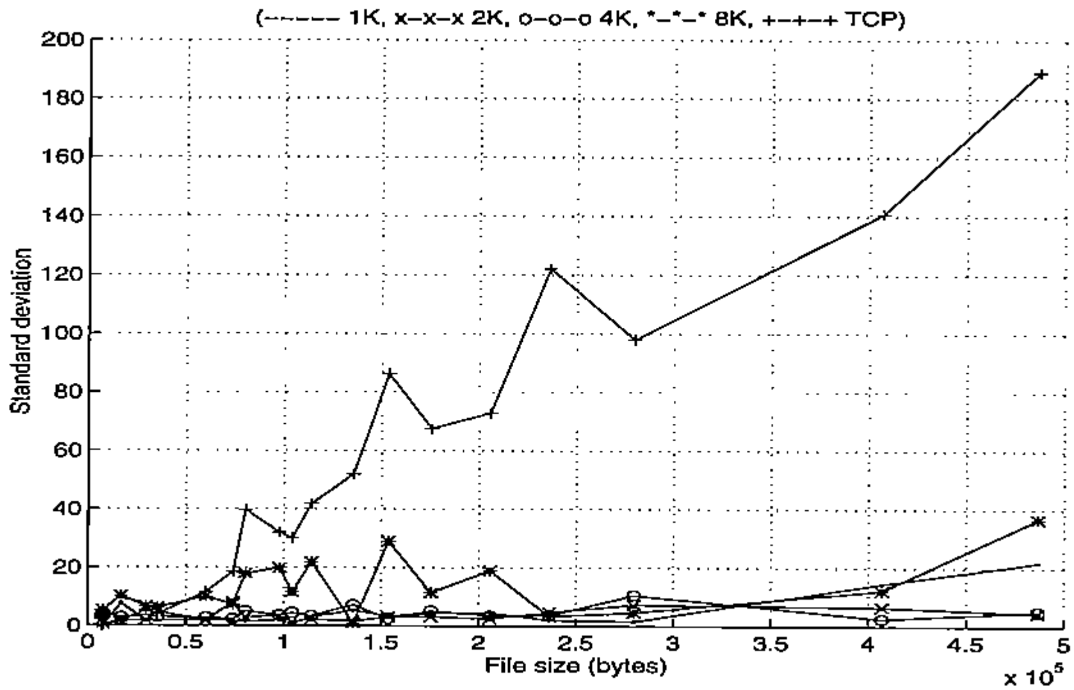

Figure 15: Standard deviation of MpTP and TCP at 32 KB/sec

- The delay in MpTP sessions is not guaranteed. Since the standard deviations of MpTP sessions are small, however, MpTP sessions are easier to handle than TCP sessions. It would be even better if MpTP can be used in conjunction with some resource reservation protocol such as RSVP [BZE+95] to guarantee the Quality of Service (QoS).

6 Related Works

Selective retransmission is not a new concept. In fact, several retransmission mechanisms are proposed in the past, but was abandoned because it is considered not suitable for real-time data transmission. However, there are a lot of renewed interest on application of retransmission mechanism to multimedia data transmission (see [Dem94, PSA96, MAC96]). In some protocol such as XTP [Str95], there are built-in error-control mechanisms which support selective retransmission. Even the traditional TCP protocol is revamped to add selective retransmission [MMFR96] functionality. However, it is still not clear how to choose the parameters for better performance. Since MpTP is not a full-fledged protocol, it is impossible to do an apple-to-apple comparing of MpTP with other protocols. In fact, MpTP can be implemented on top of several transport protocols which are specifically designed for multimedia and real-time transmission. In the following paragraph we will discuss the differences between MpTP and XTP and the difference between MpTP and RTP.

XTP is a transport layer protocol designed to provide a wide range of communication services in a single protocol. The differences between MpTP and XTP includes:

- XTP maintains the stream semantic. It use sliding window to achieve flow-control. In MpTP we consider an object as a transmission unit with reliability features.

- XTP separates the transmission paradigm and policy out. Therefore it didn't specify how and when the retransmission should be triggered. The condition to trigger retransmission is defined by the applications. In this respect MpTP can be thought as a complement to XTP which specifically define the policy of retransmission.

- XTP didn't negotiate packet and object sizes before transmission. This will make the receiver side difficult to maintain the status of current transmission.

RTP is a transport protocol which is designed for real-time transmission. In RTP there is no notion of retransmission and rate control. The designer of RTP think that retransmission is not allowed in a real-time application, so they completely abandoned this idea. However, in [Dem94] the author shows that retransmission is sometimes useful and feasible if used properly. For some near-real-time application such as audio/video-on-demand, it is possible to use MpTP to transmit data in a chunk fashion. With proper buffering, MpTP will provide low-variance data chunks with very high reliability for continuous media applications. Incorporating this characteristic of MpTP into RTP will be very useful to this kind of applications since it is very difficult to employ open-loop error control mechanisms (which tolerate errors in the data and try to recover the data based on the correct data received) to the data with too many errors. It may even be not useful at all if the data contains too many errors. MpTP can balance between the transmission time and reliability. This can be think of as amortizing the cost of multiple-pass retransmissions into several consecutive frame of video or segments of voice.

7 Conclusion and Discussion

There are several reasons why in contrast to TCP, MpTP can achieve very high throughput. First, TCP incorporates congestion control algorithms when packet loss is detected. The congestion control algorithms slow down the transmission process. In fact, if we transmit the data too aggressively using MpTP, other connections which share the same link with us and use TCP may back off exponentially and give the bandwidth to MpTP sessions. This is a common problem for all the real-time transport protocols. However, it is not fair to the other connections. Therefore, the program must be very careful not to send data too fast. Even if we do not send the data too aggressively, we can still get very good throughput. The reason is that for a connection with large bandwidth-delay product (like the one in our experiments, which have packet RTT of about 1000 ms and bandwidth 1.54 Mb/sec), the acknowledgment mechanism of TCP cannot effectively utilize the available bandwidth. This is because the capacity of the link will be large for a long fat connection (connection with large bandwidth-delay product) and TCP needs to have a very large window of unacknowledged data to keep the network busy. Another disadvantage of TCP is that TCP tends to retransmit unnecessary packets when a single packet is lost. MpTP will not retransmit any packet which is acknowledged by the receiver, and so it is more efficient. One other reason for the efficiency of MpTP is that the receiver only acknowledges the reception of data after a complete data sending round. In contrast, TCP acknowledges very frequently which wastes some bandwidth.

Second, MpTP is inherently very fast. This is because that even in the high loss rate network, the first several rounds of data transmission will send most of the data to the destination. In fact, only the first or the second round of data transmission contains a large volume of data, the subsequent rounds simply transmit small quantities of data. Therefore, it is inherently very fast.

One very important observation of our experiments is that when sending data over WAN environments, smaller packet tends to produce better results. Some researchers suggest that in LAN environment, using the maximum possible UDP packets improves performance [VM90]. However, as we can see from the experiment results, this rule is not applicable to the WAN environment.

Although MpTP is aimed at transmitting data over a WAN environment, it is also useful in the LAN environment. Of course, the gain in the LAN environment is not as large as in the WAN environment because the loss rate is low and TCP can adjust its throughput to near optimal.

Currently the optimal size of a chunk is determined only by experiments. Although we have conducted similar experiments on other remote sites and have obtained similar results, the optimal size of a chunk under a different loss rate and different remote site is not yet decided analytically.

In order to avoid packet loss in the sender side and control the sending rate (UDP cooling method), in MpTP sessions there is a parameter (called the cooling parameter) which controls the delay between two successive packet transmission. From the experimental results we can see that sending data in a too aggressive way will not always result in an improved throughput. In some cases it even aggravates the throughput. The optimal number is also not known analytically and needs more investigations.

The semantic of MpTP is neither packet nor stream. It is a completely new semantic. It is interesting to know the applicability of this semantic to a wider range of problems in order to best utilize this method. One promising application of MpTP is the World-Wide Web (WWW) because currently the HTTP protocol form a new TCP connection between the server and the browser for each data object. This separation of connections correspond naturally to separate MpTP sessions. We can replace each new TCP connection by a MpTP session and achieve the same functionality but with faster respond time.

When the size of the object is too large (for example, several MBytes), feedback delay using MpTP may be too long for some applications. A possible improvement of MpTP is segmentation. That is, divide the object into some smaller pieces (say 512K a piece) and establish multiple sessions one after the other.

8 Acknowledgments

We acknowledge Chinese Knowledge Information Processing Group (CKIP) at Institute of Information Science, Academia Sinica, Taiwan, ROC for their support on providing a remote machine and network for the WAN experiments.

References

[BZE+95] R. Braden, L. Zhang, D. Estrin, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) - version 1 functional specification. *Internet Draft*, March 1995.

[Dem94] Bert J. Dempsey. *Retransmission-based Error Control For Continuous Media Traffic In Packet-Switched Networks*. PhD thesis, University of Virginia, May 1994.

[GV93] M.W. Garrett and M. Vetterli. Joint source/channel coding of statistically multiplexed real-time services on packet networks. *IEEE/ACM Transactions on Networking*, 1(1):71–81, February 1993.

[KV89] G. Karlsson and M. Vetterli. Packet video and its integration into the network architecture. *IEEE Journal on Selected Areas in Communications*, 7(3):739–751, June 1989.

[LCB95] Xianging Sean Liu, Lebin Cheng, and Bharat Bhargava. Effective strategies for multimedia message transmission: UDP Cooling and TCP. Technical Report 95-067, Purdue University, Department of Computer Sciences, October 1995.

[MAC96] R. Marasli, P. D. Amer, and P. T. Conrad. Retransmission-based partially reliable transport service: An analytic model. In *Proceedings of IEEE INFOCOM '96*, March 1996.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A Romanow. TCP selective acknowledgment options. *RFC 2018*, April 1996.

[Pos80] J.B. Postel. User datagram protocol. *RFC 768*, August 1980.

[Pos81] J.B. Postel. Transmission control protocol. *RFC 793*, September 1981.

[PSA96] Sassan Pejhan, Mischa Schwartz, and Dimitris Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. *IEEE/ACM Transactions on Networking*, 4(3):413–427, June 1996.

[SCFJ96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. *RFC 1889*, January 1996.

[Spr79] M. D. Springer. *The Algebra of Random Variables*. John Wiley & Sons, 1979.

[Str95] Tim Strayer(Ed.). Xpress Transport Protocol Specification. Technical Report XTP 95-20, XTP Forum, March 1995.

[VM90] Paramesh Vaidyanathan and Scott F Midkiff. Performance evaluation of communication protocols for distributed processing. *Computer communications*, 13(5), June 1990.