A High-Performance Data Transfer By Using An Application Level Transport Protocol

Khin Ngu Wah Lai Yee,Khaing Khaing Wai
nguwah89@gmail.com,khaingkhaing.73@gmail.com
Computer University, Maubin

Abstract

Delivery of data files as large as media type files between business organizations using file transfer is becoming an increasingly attractive alternative to the physical movement of CDs, flash memory drives or disks. However, widespread adoption of file-based delivery, especially over public networks such as the Internet, requires the adoption of secure, reliable and interoperable solutions. This thesis presents the work of sending media type files using the UDP based Data Transfer (UDT). Case studies of the implementation and use of UDT are included, together with an investigation of transfer performance.

Keywords : Fast Data Transfer, UDP Data transfer, UDT, Application level data transport, UDP based Data Transfer

1. Introduction

As network bandwidth and delays increase, TCP becomes inefficient. These problems are due to slow loss-recovery, a RTT (Round Trip Time) bias inherent in its AIMD (Additive Increase Multiplicative Decrease) congestion-control algorithm, and the bursting data flow caused by its window control. Data-intensive applications over high bandwidth delay product (BDP) networks, such as computational grids, need new transport protocols to support them. To meet this requirement, we have developed an application-level protocol built above UDP, called UDP- based Data Transfer protocol, or UDT. UDT has a congestion control mechanism that maintains efficiency, fairness and stability, and its application-level nature enables it to be deployed at the lowest cost, without any changes in the network infrastructure or operating systems.

This thesis studies different types of data transfer protocols and observe the efficiency of using UDT protocol. It is intended to provide a general purpose data transfer service that can utilize the bandwidth efficiently and fairly. This protocol transfers data from one single byte to hundreds of gigabits. It uses UDP with loss detection /retransmission and congestion control.

2. Related works

UDP-based File Transfer Protocol UFTP is implemented on the top of the connectionless oriented protocol UDP. Its objective is to optimize the efficiency and performance of the packet transfer on the Internet to greatly reduce the network latency. To better understand the UFTP, let's briefly explain what kinds of network connections are used for an FTP session and compare them to those used in our new protocol. FTP uses two TCP connections, one for exchanging command/control packets, and the other for the data itself. Basically, FTP protocol is not concerned about retrieving the missing/corrupted packets. The job of "providing a reliable network connection" is delegated to the transfer layer protocol TCP. [1]

Media Dispatch Protocol (MDP) : It specifies how the manifest documents should be exchanged, and has defined the semantic rules for a delivery. This has led to the development of the Media Dispatch Protocol (MDP) for orchestrating and controlling file transfer of media between organizations. Because MDP is an open protocol, anyone can implement systems using it. Just as with SMTP, the protocol that makes email possible, MDP is designed to let systems work together without prescribing how those systems are implemented. MDP provides a common way for different systems to plug together. [2]

3. Background theory

3.1. UDT - (UDP - based data transfer)

UDT is an application level protocol that is based on UDP (User Datagram Protocol). UDP has similar functionalities to TCP (Transmission Control Protocol) which is connection-oriented reliable duplex unicast data streaming. Compared to TCP, UDP is connectionless protocol that emphasis on fast data transfer. UDP is used in streaming media applications such as IPTV, Voice over IP (VoIP),

Trivial File Transfer Protocol (TFTP), online games and Instant Messaging.

Although based on a widely used protocol UDP, UDT is considered as a new protocol design and I mplementation accompanied with a new congestion control algorithm. It also has a capability of configurable congestion control framework.

3.1.1. Packet formats. There are two basic classes of packets in UDT: data packet and control packet. They are distinguished by the first bit of the packet header. The header of a data packet is a flag bit of "0" followed by a 31-bit sequence number. The value of sequence packet's numbers are ranged from 0 to (2^30_1). UDT does not allow packet size larger than MTU (Maximum Transfer Unit) ,so the largest application data size can be carried in one packet is(MTU - 32) bytes. UDT always tries to pack data in the maximum size and the unit of numbers of packets per second is used to measure transfer speed in data communication. If the first bit of a UDT header is 1, then the packet is marked as a control packet,

B it 0: Flag=1	**Bit 1-3: Type**	**Bit 4-15: Reserved**	**Bit 16-31: ACK ID or loss length**

Figure 1. Control packet format

3.1.2. Control information field. Content depends on type field:

- Type 000 (handshake) : maximum flow window size, MTU
- Type 001(keep-alive):None
- Type 010(ACK):acknowledged sequence number, RTT, packet arrival speed, estimated bandwidth
- Type 011(NAK): loss information
- Type 100(delay increase warning): None
- Type 110(ACK^2):None

3.1.3. Protocol architecture

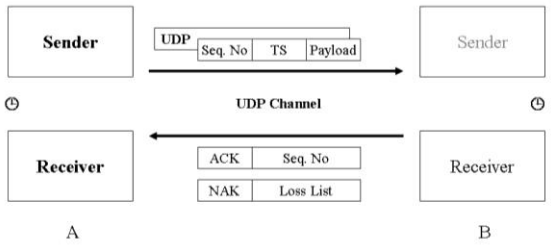

Figure 2. Diagram of protocol architecture

4. Architecture of UDT based application system

The UDT layer has five function components:

1. the API module
2. the sender
3. the receiver
4. the listener and
5. the UDP channel

Four data components:

1. sender's protocol buffer
2. receiver's protocol buffer
3. sender's loss list and
4. receiver's loss list.

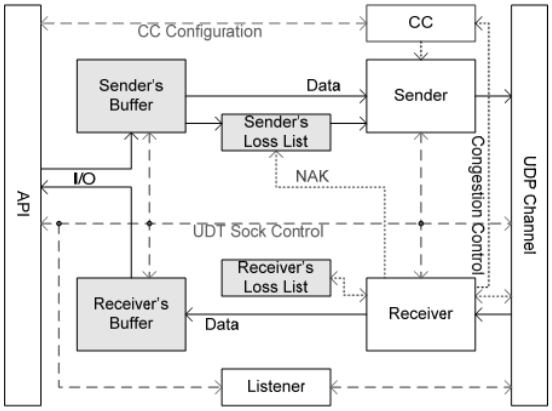

Figure 3. UDT components

UDT is bi-directional, all UDT entities have the same structure. The API module is responsible for interacting with applications. The data to be sent is passed to the sender's buffer and sent out by the sender into the UDP channel.

At the other side of the connection (not shown in this figure but it has the same architecture), the receiver reads data from the UDP channel into the receiver's buffer, reorders the data, and checks packet losses. Applications can read the received data from the receiver's buffer. The receiver also processes received control information. It will update the sender's loss list (when NAK is received) and the receiver's loss list (when loss is detected).Certain control events will trigger the receiver to update the congestion control module, which is in charge of the sender's packet sending.

The UDT socket options are passed to the sender/receiver (synchronization mode), the buffer management modules (buffer size), the UDP channel (UDP socket option), the listener (backlog), and CC (the congestion control algorithm, which is only used in Composable UDT). Options can also be read from

these modules and provided to applications by the API module. The API model of UDP provides data transfer services to software developers to develop network applications.

4.1. Sender's algorithm

1. If there is no application data to send, sleep until it is waken by the application.
2. Packet sending:
   a) If the sender's lost is not empty, remove the first lost sequence number from the list and send the proper packet out.
   b) Otherwise, if the number of unacknowledged packets does not exceeded the flow window size, pack a new packet and sent out.
   c) Update the number of sent packets since last SYN time.
   d) If the current packet and the next packet are sampled probing packet pair. Go to 1).
   e) Wait to the next packet sending time; Wait for additional SYN time if the rate control has decided that data sending should be frozen. Go to 1).

4.2. Receiver's Algorithm

1) Query the timers
   a) If ACK timer is expired and there are new packets to acknowledge, send back an ACK report;
   b) If NAK timer is expired and the receiver's loss list is not empty, send back an NAK report;
   c) If SYN timer is expired:
      - If the number of received packets since last SYN time is greater than 0, update the NAK interval.
      - If the number of sent packets since last SYN time is greater than 0, update sending rate.
      - If EXP timer is expired, put all the sequence numbers of sent packets since the last acknowledged number into the sender's loss list.
      - Reset the expired timers.

2) Start time bounded UDP receiving. If nothing received before the UDP timer expires, go to 1).
3) If the received packet is a control packet, process it, and reset EXP timer; if it is ACK or NAK; go to 1);
4) Compare the sequence number of current data packet (A) and the largest sequence number ever received (B):
   a) If A > B + 1, generate a loss report , put the sequence number between A and B into the receiver's loss list;
   b) If A < B, remove A from the receiver's loss list;
   c) Update B;
5) If the size of current packet is not equal to the fixed UDT packet size, record the current sequence number and the size difference (for buffer management use). Go to 1).

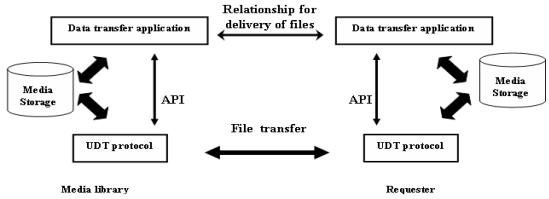

Figure 4. Media transfer diagram

The data transfer programs of the requester and the Media library use the API of the UDT protocol. The sender and the receiver programs are connected through the protocol. When sending a requested file from the media library, the program selects the required file and sends it to the requestor PC. When the transmission is initiated, the file transfer handling is performed by the UDT protocol.

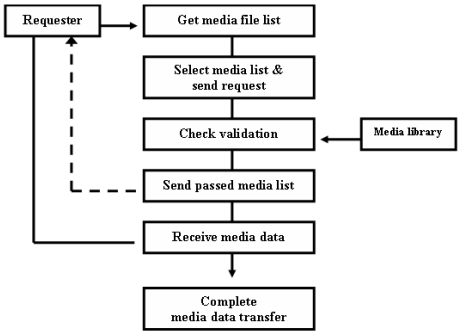

Figure 5. System flow diagram

Figure 5 shows the process flow of the proposed system. Select the file from the Media file list. The request will be sent to the "Media library" to validate the media file existence. The selected media file will be sent to the requester on UDT protocol based data transfer. The algorithm of UDT protocol is mentioned above by sender's and receiver's.

algorithms. The process will terminate after the data transfer is completed.

Figure 6. Select media file

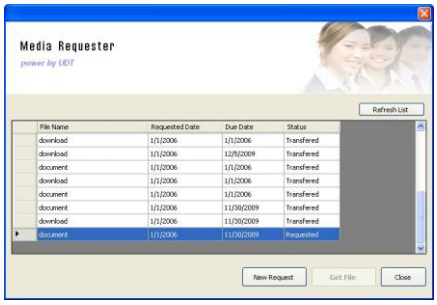

Figure 7. View of requested files list from requester

Figure 8. Requested file lists from media center

All requested files list is seen in this media center. Selects requested files and then click the "Allow" button to transfer requested files When media file has been successfully received, the status of media center becomes "Transferred".

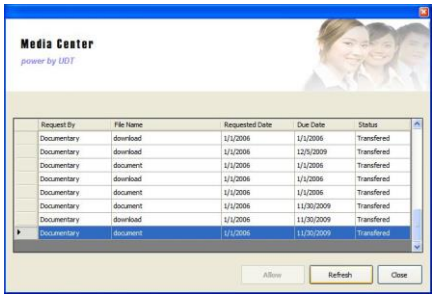

Figure 9. Transfer media files list

4.3. Experimental results

According to the transfer test using the UDT protocol, the following result

File type	File size	Transfer Duration in Minutes : seconds
Pdf	(3.1 mb)	1:35
Mp3	(5.2 mb)	1:45
Mp4	(9.8 mb)	3:40

Table 1 : Comparison of Transfer rate on media file

5. Conclusion

High-speed data transfer creates many challenges for the design and implementation of different transport protocols. Any additional action on per packet processing can lead to a significant increase in CPU usage, whereas a bursting of CPU usage can further lead to packet loss. Moreover, on long distance links, the number of on flight packets is also huge and requires large data storage to temporally record their information. Access to such data storages is also critical. The UDP based protocol combines both rate based and window based congestion to reach efficiency and fairness objectives. The protocol of UDT can utilize the abundant optical bandwidth independent of the link capacity and network delay. Using constant synchronization time enables the protocol to reach fairness independent of RTT.

6. References

[1] Jingsong Zhang and Robert D. McLeod, "A UDP-Based File Transfer Protocol (UFTP) with Flow Control using a Rough Set Approach" , Dept. of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, R3T 2N2,Canada.

[2] Peter Brightwell , "Standardising media delivery in a file-based world" , BBC Research White Paper WHP 158.

[3] YUNHONG GU, B.E., "UDT: A HIGH PERFORMANCE DATA TRANSPORT PROTOCOL", Hangzhou Institute of Electronic Engineering, China, 1998, M.E., Beijing University of Aeronautics and Astronautics, China, 2001.

[4] Yunhong Gu and Robert L. Grossman, "UDT: An Application Level Transport Protocol for Grid Computing", Laboratory for Advanced Computing / National Center for Data Mining, University of Illinois at Chicago, 700 SEO, M/C 249, 851 S Morgan St, Chicago, IL 60607, USA.