IEEE MELECON 2002, May 7-9, 2002, Cairo , EGYPT.

MUST, a Multicast Synchronous Transfer Application for Fast Intra-Campus Replications

Josemaria Malgosa-Sanahuja, Joan Garcia-Haro, Fernando Cerdan, Francesc Burrull-Mestres

Department of Information Technologies and Communications, Polytechnic University of Cartagena
C/Dr. Fleming, s/n (Campus Muralla de Mar), Cartagena, E-30202, Spain
Ph. +34 968 325370, Fax +34 968 325338
email: {josem.malgosa, joang.haro, fernando.cerdan, francesc.burrull}@upct.es

Abstract

In this paper, we introduce a new multicast application called MUST (Multicast Synchronous Transfer) that allows the replication of hard disk partitions in a fast and efficient way. The main goal of MUST is to help system administrators to easily maintain and update all computers in a lab. MUST transmits its IP packets in multicast mode, avoiding the use of unnecessary bandwidth. Unfortunately, the well known TCP transport protocol cannot be used in this multicast environment since the number of TCP flow control windows (and consequently, the number of ACK) that server should process, grow exponentially. Therefore, we also outline an alternative multicast transport protocol specifically designed for this application. This way, the number of acknowledge frames are drastically reduced, allowing the use of MUST application in a LAN and intra-campus (directly interconnected LANs via few routers) scenarios. The application has been programmed in C language using standard *kernel* routines, therefore, MUST can mount almost all existing file systems. In addition, we can distribute our application world wide without incurring legal conflicts.

Keywords: IP-multicast, multicast transport protocols, multicast applications.

1. INTRODUCTION

The MUST application is a necessary tool to reduce the time expended in maintaining both research and teaching laboratories, but especially in the last one since students often need to login as *root* and consequently, the risk is high. Furthermore, teaching labs are used and shared by different subjects, therefore at the end, they normally need an in depth reinstallation.

To start working with MUST, the system administrator has to configure one client machine manually. This step includes setting-up the hard-disk partition table and the installation of the operating systems and programs needed by that client. Then, the system administrator burns the client's hard-disk partition in a CD or DVD (or directly copies it to a local server hard-disk). At this time, MUST is responsible for transferring each partition image to all clients connected to the network. Indirectly, we are supposing that all clients have the same hard-disk structure but in fact, this is not strictly necessary, since LBA (*Logical Block Addressing*) hard-disk controller allows linear access to storage devices using sectors as unique positioning parameters. The only real restriction is that all clients must have the same partition table. In addition, MUST is able to transfer not only a partition image, but also individual files.

Doubtless, multicast mode is the best way to transfer information between server and clients. Each information packet is transferred to all clients simultaneously, avoiding unnecessary network overload. Therefore, IP packets are transmitted using the private class D addressing scheme. Unfortunately, the well-known connection oriented TCP transport protocol cannot be used in this multicast environment due to the feedback implosion (the number of ACK frames increases exponentially when the number of clients grows). Moreover, if the network has some asymmetry (as in an intra-campus environment, where the links between routers may temporarily be overloaded) the flow and error control probably will shut down the server process capacity. Therefore, MUST also includes a transport protocol implementation that drastically reduces both the number of feedback frames and the number of re-transmissions due to error.

MUST has been written in C language using standard Linux *kernel* routines. This approach has several advantages, i.e., the possibility to construct a boot floppy disk (or a bootable CD-ROM) without generating legal conflicts. In addition, the actual *kernel* code can mount almost all types of file systems, enabling MUST to replicate not only Unix files, but also Windows, MacOS, Solaris, Irix and so on. An alternative to MUST is *ghost* (trademarked by Symantec), but *ghost* requires a windows operating system for execution. MUST does not. Furthermore, thanks to the developed transport protocol, MUST can work properly in an intra-

0-7803-7527-0/02/$17.00 ©2002 IEEE.

campus environment, with some sort of network asymmetries. *Ghost* cannot.

The basic multicast concepts and their repercussions on the physical [1][2][3][4], link and network [5][6][7][8] levels of the OSI layer model are out of the scope of this paper. Section 2, explains how our proposed transport protocol operates. In section 3, we describe how to adapt our proposed transport protocol to work properly in an intra-campus environment. Finally, in section 4 concludes the paper.

2. MUST TRANSPORT PROTOCOL (MTP)

Transport protocol is the main issue of the MUST application since *kernel* code and the *socket* interface are both prepared to accept multicast packets at the network layer. However, TCP is not well suited for this purpose and therefore, multicast programmers need to define for themselves an appropriate transport protocol. Normally, programmers prefer to use an UDP *socket* and afterwards insert the desired transport protocol inside the UDP packet. This scheme has the advantage that error control is automatically carried out by the *kernel*.

In the last years, LAN technology has experimented substantial changes. For example, traditional coax-wired LANs have been replaced by UTP, STP twisted pair wired LANs (simultaneously, the shared bus topology also has been substituted by a star topology). Transmission capacity has been increased from 10 Mbps to 100 Mbps (and 1 Gbps). There are also LAN switches in the market at competitive prices offering to LAN users better performance than traditional HUB devices. MTP is programmed considering all these improvements. That is, in both LAN and intra-campus environments, the transport protocol can be simplified because the underlying network is good enough.

2.1 Flow Control

The most common access medium control schemes in LAN (CSMA/CD and CSMA/CA) are not full-duplex. In both cases, just before sending a frame, the access control algorithm checks the channel occupancy and when occupied, the transmission is delayed until the channel is liberated. In CSMA/CA (traditional AppleTalk networks and IEEE 802.11 wireless networks) it is more evident since the physical interface cannot send and receive simultaneously.

The LAN technology improvements explained above and the half-duplex nature of the medium access control algorithms allow the implementation of an extremely simple flow control mechanism. It works as follows:

- The client can be in Receive state (R) or in ACK state (A). In R state, the client waits until a MTP packet is received. In A state, each client tries to confirm the received packet but taking into account the existing correlation between all clients (see next section, feedback implosion reduction. At the moment, let us suppose that there is only one client). The ACK frame header carries the sequence number associated with the received packet and it is sent to all group members.
- The server can be in Send state (S) or in Wait state (W). When it is in S state, it sends the *nth* information packet to the network and immediately jumps to W state. In W state, it waits until one and only one ACK frame acknowledges the previous packet sent or the time-out expires. When W state has finished, the server returns to the S state and transmits the next information packet. Notice that the received ACK must acknowledge the *nth* transmitted packet. If sequence number of received ACK differs, server discards it.
- In both cases, the server returns to S state and transmits the next packet, but if the time-out expires before ACK is received, its value is multiplied by a factor of α. In the same way, if an ACK frame is received before the timer expires, its value is redefined as follows,

$$T_{out} = max\{T_{out}/2, default\_T_{out}\} \qquad (2.1)$$

The presented algorithm really controls the flow of information, since if congestion arises in the network, Tout is incremented and therefore, the rate at witch information packets are sent goes down. But if congestion disappears, the Tout redefinition allows to increase the information transfer rate. Another special feature is that the server never retransmits an erroneous received packet (there are not NACK frames). In section 2.3 we explain in more detail how to solve this problem. Finally, in an intra campus environment, the algorithm has some characteristics that slightly modify the rules explained above.

2.2 Feedback Implosion Reduction

To reduce feedback implosion, MTP uses the correlation between ACK frames generated by different clients. To do that, each feedback frame is sent by a client to all members of the group and therefore, each client can foresee the behavior of the other clients. Therefore, to reduce the amount of

feedback, just before sending an ACK, a client must both wait a random time period (called ARTP, *ACK Random Time Period*) and simultaneously listen if another client is transmitting the same ACK. If so, the client disables its ACK transmission. As a conclusion, only one ACK is sent to the server, independently of the number of active clients. As we will see later, in an intra-campus environment, the server receives only one ACK for each interconnected LAN.

2.3 Error Correction

The main goal of MUST is to replicate hard-disk partitions. Therefore, multicast MTP packets are filled with the physical hard-disk sector information, and from now on, we can talk about packet or hard-disk sector without distinction. For a clear understanding of the error correction procedure, we have to notice that all the sector information is permanently stored in the server hard-disk, so the server can retransmit any of the sectors at any time. At same time, any sector can be saved in a client side without a predefined order.

MTP is based on UDP transport protocol. This means that error correction is made automatically by the *kernel*. When the *kernel* detects an erroneous packet, it simply discards it. Consequently, in MUST application, receiving an erroneous packet is equivalent to losing that packet. When a client losses a packet, it will continue in Receive-state until an error free packet is received. Therefore, losses are equivalent to out of order packet receptions. Normally, when a client receives a packet with its sequence number out of order, it discards it and issues a NACK to force a retransmission. But now, since the server never retransmits and the client never issues a NACK, the client saves the out of order packet in its local hard-disk (in the position marked by the sector), and then acknowledges the packet if necessary. Since all lost packets are forgotten at the client side, MTP flow control algorithm never retransmits an erroneous frame. This fact increases the server throughput, but sooner or later, erroneous (that is, lost) packets will have to be retransmitted.

We selected this high throughput flow control mechanism because the present LANs show almost no errors (or at least, not too many errors).

The error correction algorithm operates as follows: In a first stage, neither server nor clients are focusing in error control tasks. When the entire hard-disk partition image has been transferred, both the clients and the server switch to the error-correction mode. At this mode, all clients re-scan their hard-disk, detecting all non properly formatted sectors. If one of them is found, the client delivers a Repair-Request frame towards the server. Finally, the server answers the client sending the appropriate sector into a MTP packet.

Figure 2.1 shows an example of the proposed flow control and feedback reduction algorithms. At the beginning, all the clients are in R state. When packet 1 is received, all the clients start its ARTP clocks. One of them wins the competition and automatically the other clients inhibit their ACK frame and jump to R state. When packet 2 is sent, one of the clients does not receive it (it will be recovered at the end of the hard-disk image transmission). The other clients start their ACK competition and finally one of them acknowledges the packet. Packet 3 is completely lost, so all clients continue in the R state until packet 4 is received. The server transmits packet 4 when its Tout expires and updates it, increasing its value by a factor of α. Packet 4 is fully received and correctly acknowledged, therefore server decreases its Tout value accordingly.

2.4 Losses and Out of Order Packets

Lost packets are recovered after the entire hard-disk partition image has been transferred. The no reception of lost packets has the consequence that packets are received out of order, but again, due to the fact that a packet is associated to a physical sector, the lack of order has no importance.

In an intra-campus environment, in the absence of losses, packets may still arrive out or order, because it is possible than there is more than one path for the packet to arrive at the same destination. Again, this

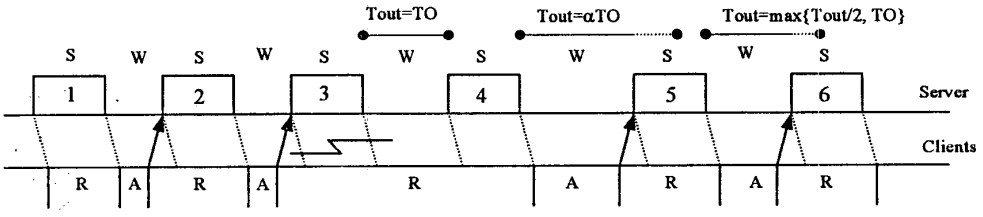

Figure 2.1, MTP-flow control example. The letters indicate the client state (R for Receive and A for ACK). The server state is S (send) when it is sending a packet and W (Wait) otherwise.

has no effect in our proposed transport protocol since the server silently discard all ACK frames with sequence number out of order.

2.5 Proposed Header

The proposed header has up to 4 fields: SN, TOP BP and PL (figure 2.2). The first field is the Sequence Number, also called the Sector Number (SN). This field is followed by the Type Of Packet (TOP) field, witch indicates if the MTP packet is an information packet, an ACK frame, a Repair-Request or a Repair-Response. The Broken-Packet field (BP) is necessary because it is possible that one sector cannot fit in only one MTP packet (usually, sector size is 512 bytes, but it could be 1024 bytes as well, and UDP datagrams are up to 512 bytes). The Payload-Length (PL) field indicates the total length (in bytes) of the MTP packet. Finally, the header is followed by the payload, that transports the information bytes.

It is possible that SN field could be no large enough to accommodate all sectors in a hard-disk partition. One solution consists of, when SN arrives to its maximum value, MUST starts the error-correction procedure and, when all the clients repair their lost packets, then SN is reset and the transfer restarted. Other quite simple and efficient solution is to transmit in one MTP packet more than one consecutive sector. Obviously, in this case, an ACK acknowledges all these sectors.

3. INTRA-CAMPUS EXTENSIONS

In section 2.2 we have seen a methodology to avoid feedback implosion. It works only if ACK frames are transmitted to the group, because all the clients must be able to receive such of these frames in order to inhibit its own ACK frames. The immediate consequence is that, in an intra-campus environment, the server receives only one ACK frame from each LAN.

To adapt our flow control scheme to this situation, we slightly modify one of the rules: a packet is completely confirmed only when the server receives one confirmation from each LAN. The server can do this task easily since each multicast packet has in its IP source address the unicast address associated with the sender. Therefore, if one of the intra-campus routers enters congestion, the transmission rate is reduced by the flow control. Other consequence is that the server must know the number of participating LANs. This can be manually introduced at startup or automatically detected in runtime. Another quite simple implication: since ACKs are sent in multicast mode, all clients in the intra-campus

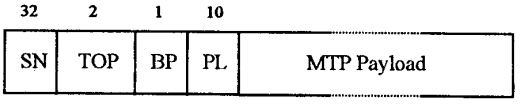

Figure 2.2, Proposed packet format. The numbers indicates the length (in bits) of each header field.

network can receive them. This can confuse the competition algorithm to reduce feedback implosion. A very easy solution is, again, extracting the unicast source address of the ACK packet and, if it is sent from another LAN network, discard it.

4. CONCLUSIONS AND FUTURE WORK

With MUST application, user can replicate in an efficient way hard-disk partitions images, since it uses multicast to avoid overloading the network capacity. The main issue in multicast programming environment is the development of the transport protocol. In this article, we have proposed a transport protocol for MUST called MTP. Its main goals are that it guaranties error-free and high throughput replication. In addition, it exhibits a low complexity implementation.

MUST (and its associated MTP transport protocol) operates good enough in a LAN scenario. For example, we have replicated a 5 Gbytes hard-disk image in 10 clients simultaneously in approximately 2 hours. Using a unicast version of MUST (with TCP) the replication time, in the same conditions, grows up to 10 hours.

However, there are still some minor drawbacks that must be solved in future work. We must find out, by means of analysis as well as by simulation, the optimum values of some parameters like Tout, α or ARTP. These values must be optimized in a broad range of scenarios. Also, the functionality of MTP must be improved in order to obtain efficient results when it tries to replicate images in a WAN environment.

Another point of interest is to distribute the error-correction task among all clients and the server instead of be centralized by the server. This is possible since all clients share the same information, and therefore, they can collaborate along with the server in the error-correction mode. Then, the traditional client-server architecture probably does not completely apply in future versions of MUST.

Acknowledgements

This work has been supported by the Spanish Research Council under projects FARIP (TIC2000-1734-C03-03) and MTCES (TIC2001-3339-C02-02).

References

[1] J. Garcia-Haro, J. Malgosa-Sanahuja, J. L. Melus, "Multicasting Facilities in ATM Switching Architectures. A Study of Several Approaches", *IEEE PACRIM'95*, pp. 90-95, ISBN 0-7803-2553-2.

[2] J. Malgosa-Sanahuja, J. Garcia-Haro, C. Marin, "Providing Multicast Services in ATM Switching. A Solution for Local Area Networks", *IEEE MELECON'98*, pp. 687-693, ISBN 0-7803-3879-0.

[3] T. T. Lee, "Nonblocking Copy Networks for Multicast Packet Switching", *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, pp. 327-339, December 1988.

[4] J. Garcia-Haro, A. Jajszczyk, "ATM Shared Memory Switching Architectures", *IEEE Network Magazine*, Vol. 8, No. 4, pp. 18-26, July-August 1994.

[5] W. Fenner, "Internet Group Management Protocol, Version 2", *RFC-2236*, November 1997, IETF Standard.

[6] D. E. Comer, "*Internetworking with TCP/IP. Principles, Protocols and Architectures, Vol. I*", Ed. Prentice-Hall, fourth edition, 2000, ISBN 0-13-018380-6.

[7] D. Kosiur, "*IP Multicasting: the Complete Guide to Interactive Corporate Networks*", Ed. John Wiley & Sons, 1998. ISBN 0-471-24359-0.

[8] M. D. Cano-Baños, J. Malgosa-Sanahuja, F. Cerdan, J. Garcia-Haro, "Internet Measurements and Data Study Over the Regional Network", *IEEE PACRIM'01*, pp. 393-396, ISBN 0-7803-7080-5.