ENSC 833-3: NETWORK PROTOCOLS AND PERFORMANCE
CMPT 885-3: SPECIAL TOPICS: HIGH-PERFORMANCE NETWORKS

End System Multicast: Implementation and Simulation

Spring 2001

FINAL PROJECT

Wen Jin
wjin@cs.sfu.ca

1. Abstract

The conventional wisdom has been that IP is the natural protocol layer for implementing multicast related functionality. However, ten years after its initial proposal, IP Multicast is plagued with concerns pertaining to scalability, network management, deployment and support for higher layer functionality such as error, flow and congestion control. IP multicast service is still not applied widely because it requires all the routers in the path supporting multicast function. In this project, I try an alternative architecture, where end systems implement all multicast related functionalities including membership management and packet replication. Such a scheme is called End System Multicast. This shifting of multicast support from routers to end systems has the potential to address most problems associated with IP Multicast. I will design a multicast membership management protocol, develop an end system multicast prototype, and use ns2 to simulate and analyze the system performance and scalability etc.

2. Introduction

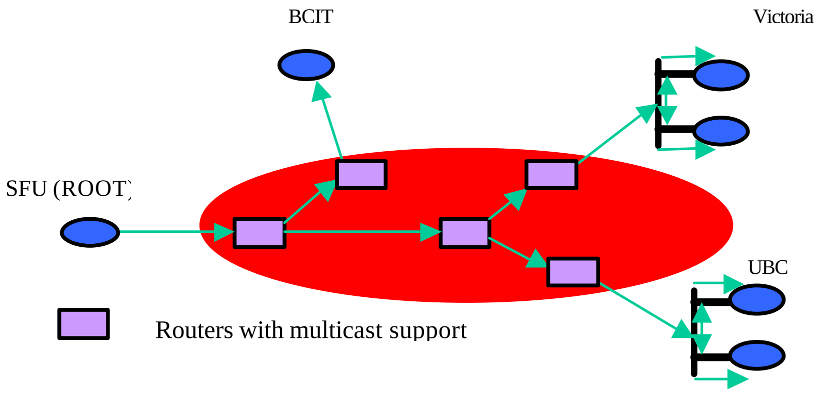


Figure 1 Traditional Structure of Multicast Systems

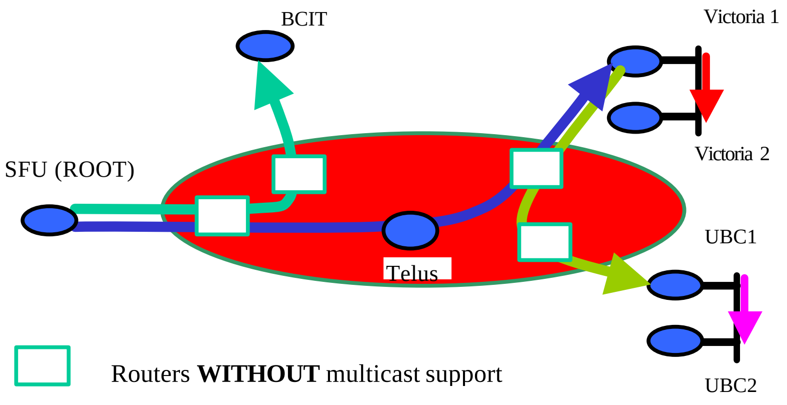


Figure 2 Structure of End System Multicast

Figure 1 shows the structure of a traditional multicast system. Here the data source as root is SFU, and several other machines in other places join this multicast group such as UBC, Telus, Victoria and BCIT together with root. Each of these machines sends requests

to a group address and the routers will process these multicast requests. Later all these joined machines construct a multicast tree. The lines with arrow depict the physical path between source and destinations. Ellipse nodes depict client machines where the data are consumed and the square nodes depict multicast IP routers. Data has to be replicated in the internal nodes which have multiple recipients. Many algorithms can be applied to decide the structure of the multicast tree, such as flooding, spanning trees and reverse path forwarding. However, most of the internal nodes are commercial routers that don't support multicast function in the current Internet. That is the main reason why the multicast service can't be deployed widely although ten years has passed from the concept's emergence. Since multicast service is so important to today's multimedia service, we need to think of some other ways to explore the good nature of multicast in which we will use end systems to implement all multicast functions.
In contrast to traditional multicast systems, end system multicast is built on top of the unicast services provided by network or transport layer. See figure 2, the edges are links in the network layer or connections in the transport layer and the root is the data source. The internal square nodes are routers without multicast support, and they are used only to do unicast transfer. The ellipse leaf nodes are end systems. Data is replicated in the application level of end systems which act as sub-server(the end system who is the father of other clients for passing data) such as Telus in Fig 2. In this project, I designed a multicast membership management protocol, developed an end system multicast prototype, and used ns2 to simulate and analyze the system performance and scalability etc. The system prototype is illustrated with video streaming as the data source.

3 End System Design

3.1. Architecture

As shown in figure 3, there is one root central member management server (MM-server) and one data server which is the data source of video stream. Clients acted as sub-servers are also video servers of their children. Each client has two connections: one for control connection (TCP) and the other for video transfer connection (UDP). All of the control information will be transmitted to the root MM-server. And the MM-server maintains all the information about its multicast tree, controls each client's corresponding actions.

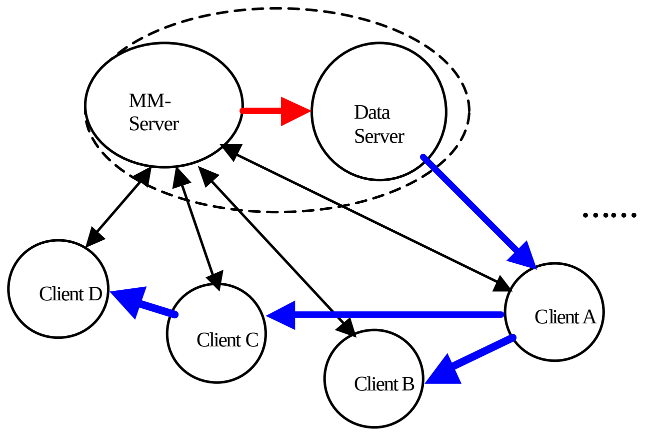


Figure 3. System architecture: Thinner black lines are member management messages, thicker blue lines are video streams, thicker red line indicates the direction from MM-server to data server.

3.2. Joining and leaving operations

The main function of a multicast control system is that it must process the clients' randomly joining and leaving requests. We use one finite state machine to illustrate these operations as in figure 4.

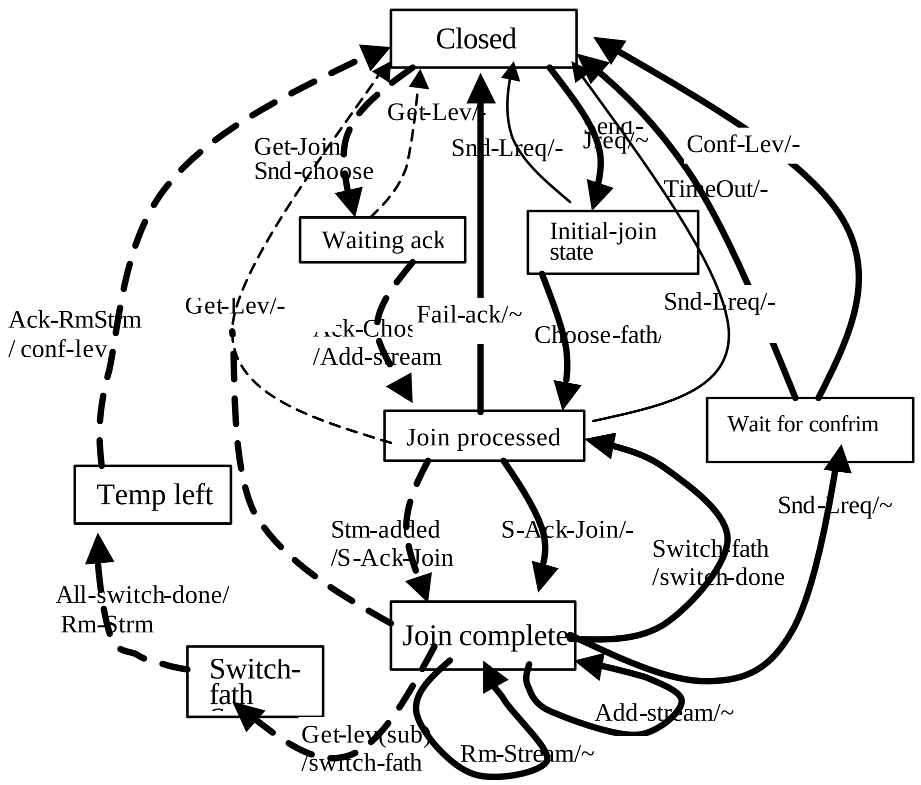


Figure 4 End system membership management state transition
Solid line represents client side transition, dashed line server side, and the thin line indicates some unusual actions.

In the end system multicast scheme, when a client want to join the group, it will send a join request, the MM-server will judge what are the optional sub-servers which are already in the group this client can join, it is judged by looking at the network address to try to find those in the same network as that client. If none of them exist, server can choose some machines arbitrarily and send to the client. Client will use its judging strategy to find one of these optional parent nodes as the father and send acknowledge to server. The selection strategy can be the shortest hops or the shortest round trip time. If the server successfully informs the sub-server this client selected to add a node as child, the join completes successfully. Otherwise, the client has to go to the initial state and do rejoin again. After the client joined the group, it can receive the data from his parent continuously.

In the case of leaving, client will send leave request first, the server will check whether this client has any children, if not, the server will inform father node of this client to remove the node from his children list and then send the confirm leave to the client. If this client has some children, the server has to send switch father command to each of those children and ask them to transform its father node. The server can decide what the new father for each child is mandatory, or it can ask that child to do a rejoin process. After all the children have switched to a new father node successfully, the server can inform that client to leave safely.

4 End System Implementation and Simulation

- **Prototype Implementation**

This is a standalone prototype system. To demonstrate the end system multicasting function, video streaming is used as the data source to show that each relay and leaf nodes in the tree structure receiving and play the same video image at the same time. The implementation process just follows the state transition chart as Fig 4.

The prototype system consists of two parts which are Server and Client part respectively. Both server and client parts include membership management and data distribution. The central membership management server maintains all the clients' current membership state and process their joining and leaving request including the abrupt leave. It is also responsible for the video transmission to its direct children nodes. While the client part is on behalf of the user, it will send join or leave request and receive the video to do a local play. Also if it is a sub-server, it will be responsible to accept command from the central server and transfer the data to its children.

The prototype is developed using standard JDK and JMF which is a java media packet for multimedia application. In both server and client sides, two threads are used to deal with membership management and data distribution respectively.

In the strategy of optional sub-server selection, the central server uses a hash table indexed by the IP address to find whether there are any nodes that are in the network as that client. While the client side uses the round trip time to judge which optional sub-server he will like to join. For the abrupt leave, the java connection exception is caught and the server will do the remaining job for switching those children of the node which left abruptly. The total code for the prototype system is over 3000 lines.

- **System Simulation**

I use NS-2 to do the system simulation. In order to transmit application-level data in *n*s, a uniform structure is needed to pass data among applications, and to pass data from applications to transport agents. It has three major components: a representation of a uniform application-level data unit (ADU), a common interface to pass data between applications, and a mechanism to pass data between applications and transport agents. The functionality of an ADU is similar to that of a Packet. It needs to pack user data into an array, which is then included in the user data area of an *n*s packet by an Agent. I use the agent wrapper TcpApp to transfer user data from application, in our case, EmulApp. The ADU in end system multicast is the data packet called EmulData. The following figure 5 shows the application simulation structure in NS for the system.

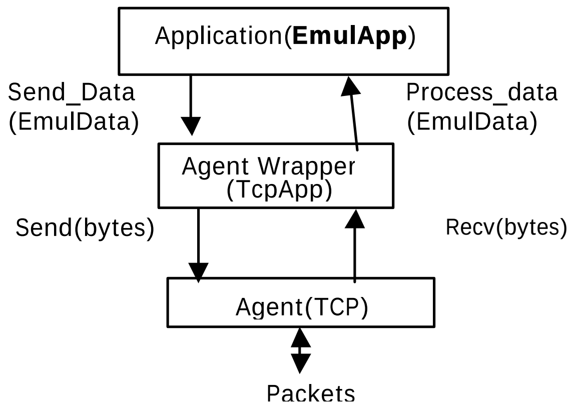


Figure 5 General Application Simulation Structure for End System Multicast

In NS, each object to be simulated is a subclass of TclObject, the same as our end system multicast application EmulApp. Under the general end system multicast application class, I also defined two subclass as server and client. The base class of Application, Process, allows applications to pass data or request data between each other. EmulApp is implemented as the subclass of Application. The following two figures (6 and 7) show the class hierarchy and the data handling structure of the end system multicast.

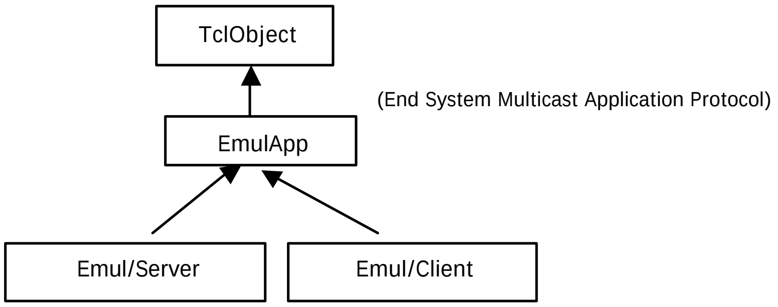


Figure 6 Class Hierarchy of End System Multicast

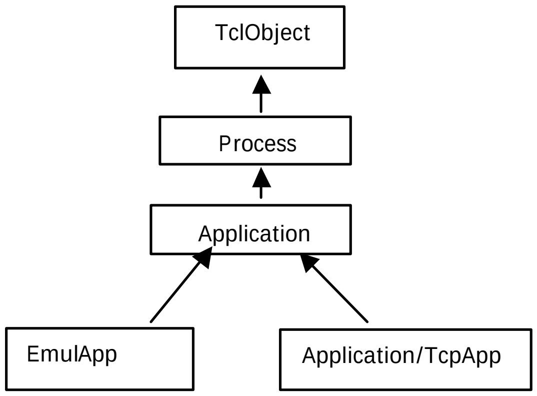


Figure 7 Hierarchy of Class related to EmulApp data Handling

In NS-simulation, each of the protocol primitive such as send-join, send-leave, choose-father, ack-join etc. is implemented as a command in the c++ classes EmulApp, EmulServer and EmulClient, or implemented as a method in the TclClass Emul, Emul/Server and Emul/Client. Figure 8 and figure 9 show the join and leaving process simulation in NS-2 using these primitives.

The whole simulation code is written by C++ and TCL. Total source code is above 2000 lines.

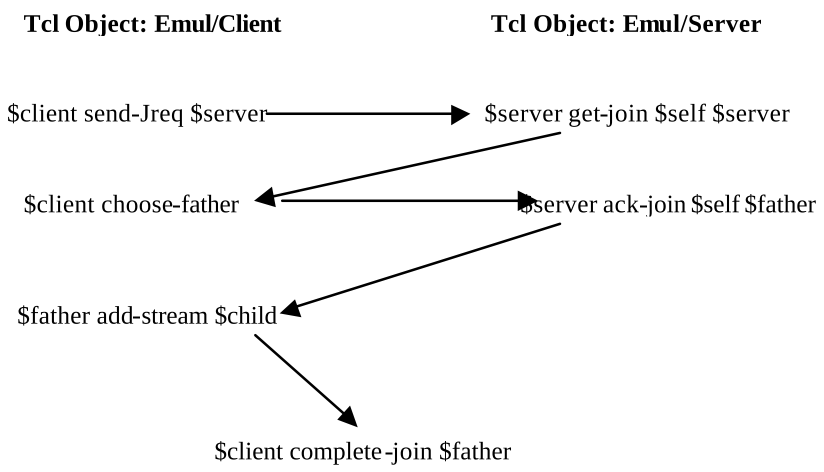

Tcl Object: Emul/Client
Tcl Object: Emul/Server
$client send-Jreq $server
$server get-join $self $server
$client choose-father
$server ack-join $self $father
$father add-stream $child
$client complete-join $father

Figure 8 Join Sequence Simulation in Ns-2

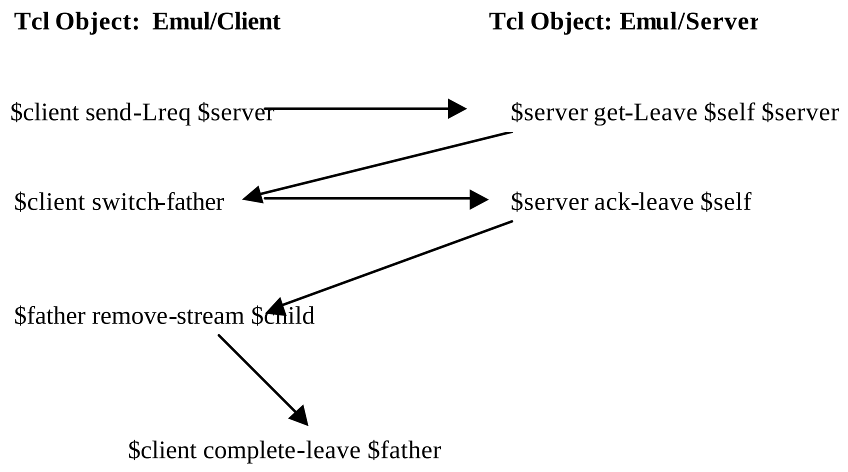

Tcl Object: Emul/Client
Tcl Object: Emul/Server
$client send-Lreq $server
$server get-Leave $self $server
$client switch-father
$server ack-leave $self
$father remove-stream $child
$client complete-leave $father

Figure 10 Leaving Sequence Simulation in Ns-2

In the simulation, there is one central server, and it will accept quite several clients' request simultaneously. We use the following scheme to allocate connections to each client. Figure 11 shows the connection allocation procedure in the level of tcp. And after the tcp agent is created, the actual communication agent TcpApp is built for the application on the clients to use.

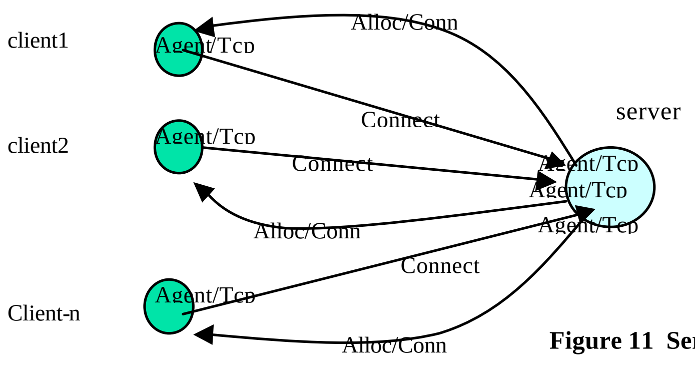


Figure 11 Server connection allocation

5 Experimental Evaluation

5.1 Standalone End System Multicast Demo

To demonstrate the end system multicasting function, we will use the video streaming as the data source to show that each relay and leaf nodes in the tree structure receiving and play the same video image at the same time. In the demo, we first run server side program in one machine, it then starts to listen to the requests from clients side. At the same time, it will play the source of video. Then we run the client side program at different machines, when it starts, it will send join requests to server. After some time, the message shows it has been joined the group, then the video is played locally. After all the clients started, we can see all the clients display the video simultaneously.

After some time, type “quit” on one of the client, it will quit normally. If it is a sub-server, it will take a short time, and the child node of this client can be seen switched to another node as the father, because the video appears on the child node takes a short while to transfer and then displayed normally. If type ctrl-c on one of the clients, the similar phenomenon will happen. The demo shows that the end system can finish multicast functionalities well without support of any commercial multicast routers.

5.2 Ns-2 Simulation Results

To further understand the performance features of end system multicast such as time delay in join, leaving, switch-father, data replication etc., we use ns2 to simulate these functionalities in end system, then analyze the simulation results with different flow rate, and different size of packet etc.

As for different topology of simulation, we can write TCL scripts to generate different size of groups for simulation, and also the bandwidth and delay for different links between these nodes can be specified. In the experiment, several different sizes of topology are simulated. The following two figures (12 and 13) show two cases with different nodes number in the network as 7 and 50.

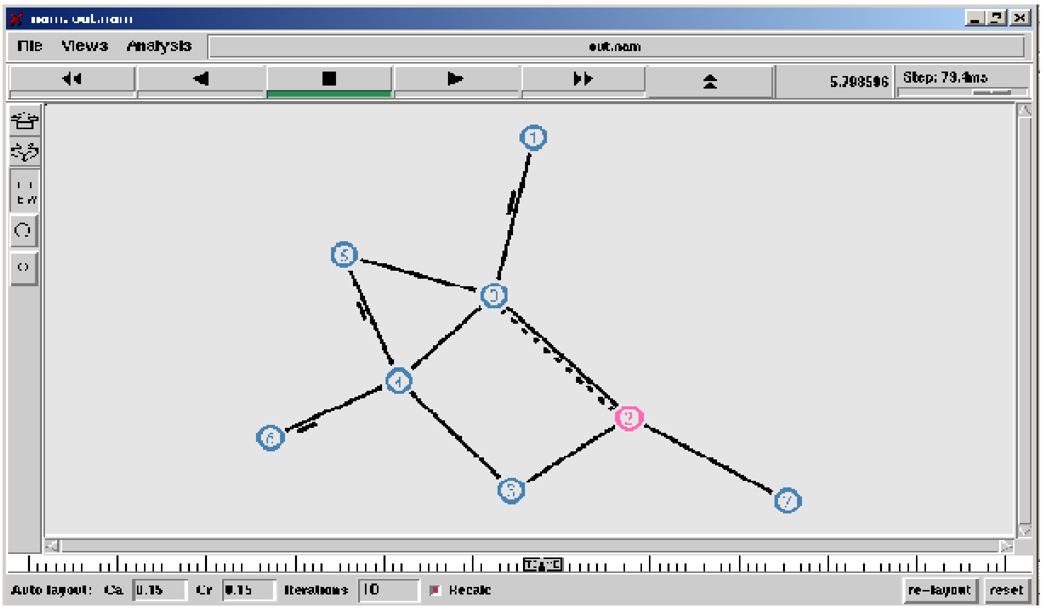
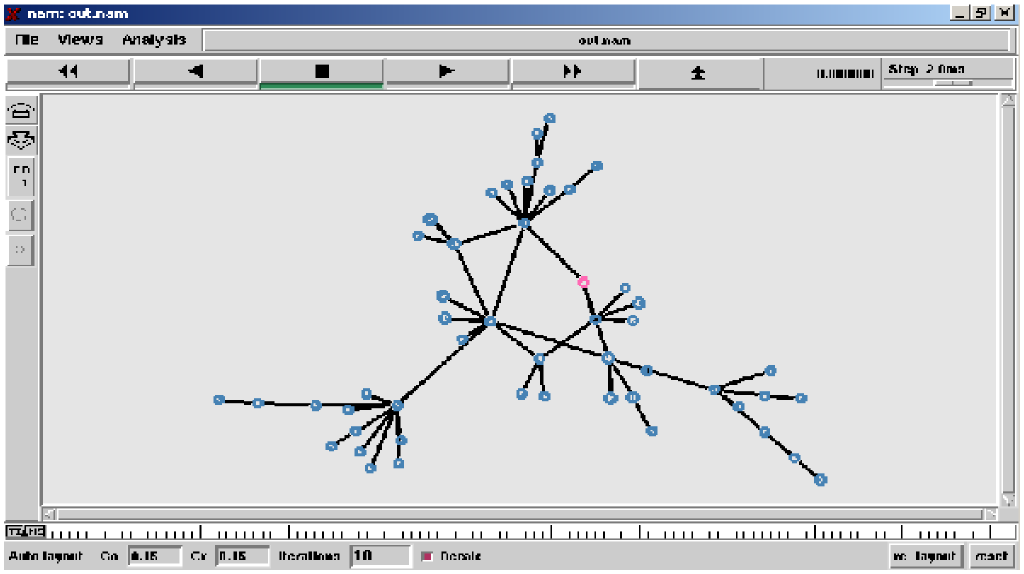


Figure 12 Topology of nodes (7)



Ca 0.15 Cr 0.15 Iterations 10 Recalc

Figure 13 Topology of nodes (50)

In the simulation, we can see some node begin join, and after some time, it joined and the color of the node changed from blue to green, also there is some data flow send to it from its parent. And when some node begin to leave, if it is a sub-server, we can see its children being switched to some other node as the father, and then this node changes color from green to red which means it left the group.
In the NS simulation, I use the dynamic routing policy for the whole network, that means the client node can compute the hops to any of other nodes, so that it can choose the node with the shortest metric as its father when the server send several optional sub-servers for it to choose.
In the switching father process of leaving, for the simplicity of implementation, I just let the server ask the child to switch to the parent of the leaving node (the child's original grandfather).
For testing the data send time delay, I let the leaf or any sub-server node to send data to the root. Each parent of a node receives the data, it will send to its parent again. When this iteration completes, the data reached the server, and the server calculate the transfer time by subtracting the current scheduling time with the data start time.
Table 1 and figure 14 give the time delay for different process such as join, leave, data transfer etc. for different size of network. It is reasonable to see that when the number of the relay (sub-server) nodes between the server and the clients grows, the data delay will increase in some extent, but it is also acceptable.

Type	Top(5)	Top(10)	Top(20)	Top(30)	Top(50)
Join	0.195	0.214	0.181	0.193	0.217
Leave	0.065	0.107	0.112	0.082	0.103
Switch-Father	0.087	0.094	0.120	0.065	0.096
Data Transfer	0.148	0.195	0.245	0.356	0.430

Table 1 time delay for different operations in end system
Top(n): topology with n nodes. Time: seconds
Flow character: Packet size: 500bytes, burst time: 1 sec., idle time: 0.5sec. Traffic rate: 2Mb/sec

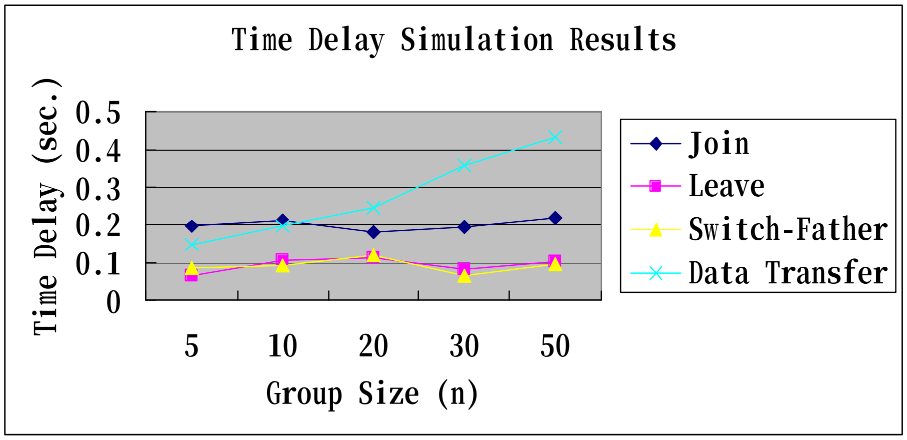


Figure 14 Time delay chart for the operations in end system

To measure the actual flow in some links, we choose several links to observe the flow variations during the testing period. I use an exponential distributed flow to test with the peak rate of 2Mb/s, Packet size: 500bytes, burst time: 1 sec., idle time: 0.5sec. The following figure shows the case of network size of 50 nodes with 3 link samples between nodes 4-5, 12-26, 4-41. It shows in the network configuration of the test topology, all the sampled links can satisfy the flow requirements without any packet loss. For example, the flow of the green one reaches the peak data rate of 2Mb/s which meets the flow requirement.

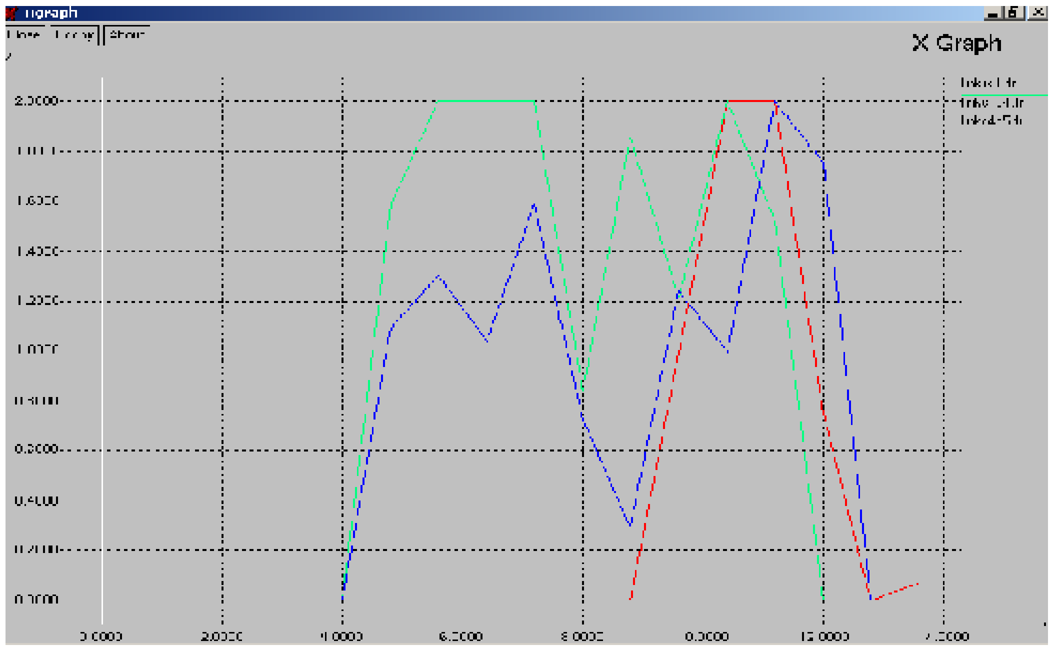


Figure 15 Flow rate of 3 links under network size of 50

6 Discussion and Conclusion

In this project, I finish end system prototype with multicast functionalities. I simulate the end system protocol in NS-2 in order to analyze its performance and scalability. The experiments show this approach is feasible, especially for medium-size groups. The simulation can be contributed to NS-2. The future work can be extended to distributed control and dynamical topology strategy.

Acknowledge
Thanks Prof. Ljiljana Trajkovic and Milan for their very useful suggestions in this project.

Reference

[1] S. Deering. "Multicast routing in internetworks and extended lans." In proceedings of the ACM SIGCOMM 88, pages 55-64, Stanford, CA, Aug. 1988.
[2] E. Bommaiah, A. McAuley, R.Talpade, and M. Liu. "Yallcast: Extending the internet multicast architecture." Technique report, Sept.1999, http://www.yallcast.com
[3] J. Liebeherr and B.S. Sethi. "A scalabe control topology for multicast communications." In proceedings of IEEE Infocom, April 1998
[4] Y.H.Chu, S.G.Rao, and H.Zhang, "A Case for End System Multicast.", In proceedings of ACM Sigmetrics, Santa Clara, CA, June 2000
[5] H. Schulzrinne, V. Kumar. "Frequently Asked Questions (FAQ) on the Multicast
[6] Backbone (MBONE)." http://www.cs.columbia.edu/~hgs/internet/mbone-faq.html.

Appendix

Code List (NS-2 simulation code is attached)