INTERNET DRAFT
Expiration: 25 August 2002

M. Pullen
George Mason University
Y. Huang
George Mason University
L. Huang
George Mason University
P. Singh
OPNET Technologies
25 February 2002

Expanded Simulation Models for IntServ and DiffServ with MPLS
<draft-pullen-qos-sim-models-02.txt>

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Abstract

This document describes detailed models for analysis of proposed protocols for Quality of Service (QoS) delivery with IPv4 (including multicast). The models have been developed using the OPNET simulation package. They are intended to allow investigation of performance using proposed protocols and should have wide applicability in the Internet QoS community. We are making these models publicly available with the intention that they can be used to provide expanded studies of Qos delivery for both unicast and multicast, using IntServ, DiffServ, and MPLS. This Internet-Draft is intended to form the basis for an Informational RFC that extends the previous RFC2490.

1. Background

2. The OPNET 8.0 Simulation Environment
   2.1 MOSPF
   2.2 IP and Multicast Models
   2.3 IGMP Process Models
   2.4 RSVP Process Model
   2.5 Multicast Application Models
   2.6 Creation of Multicast Routing Processor Node
   2.7 MOSPF Interaction with Other Protocols

3. OSPF and MOSPF Models
   3.1 Init
   3.2 Idle
   3.3 BCOspfLsa
   3.4 BCMospfLsa
   3.5 arr
   3.6 hello_pks
   3.7 cospfspfcalc
   3.8 ospfspfcal
   3.9 upstrNode
   3.10 DABRA
   3.11 QOSPF

4. Modeling DiffServ-Enabled Multicast over MPLS Tunnels
    4.1 Architectural Changes in OPNET 8.0
    4.2 OPNET 8.0 MPLS Model
    4.3 DiffServ Features
    4.3 Traffic Marker
    4.4 Simulation Models

5. Example of Use
    5.1 Network models
    5.2 IntServ simulation
    5.3 DiffServ simulation

6. Future Work

7. Security Considerations

8. References

Authors' Addresses

Full Copyright Statement

1. Background

The successful deployment of IP multicasting (IPmc)[1] in parts of the Internet and its availability in the Mbone has led to continuing increase in real-time multimedia internet applications. Because the Internet traditionally supported only a best-effort quality of service, there is considerable interest to create mechanisms that will allow adequate resources to be reserved in networks using the Internet protocol suite, such that the quality of real-time traffic such as video, voice, and distributed simulation can be sustained at specified quality of service (Qos) levels. The Integrated Services (IntServ)[2] and Differentiated Services [3] frameworks were developed for this purpose. The RSVP protocol [4,5] and MPLS protocol [5] that provide important facilities for IntServ and DiffServ are now progressing in the Internet standards process.

The ability to simulate the performance of a proposed new protocol or feature is very important to the work of the IETF. However, no open models of RSVP and IPmc were available in the late 1990s. Therefore a group at George Mason University (GMU) developed an open source family of models that work in the OPNET commercial simulation environment, and documented these models in RFC 2490 [6]. Since that time, much progress has been made in IntServ and DiffServ deployment and standardization. However, the associated protocols are still on the leading edge of Internet technology and thus the need for the ability to simulate them as part of the IETF's design and development work remains. Also, since publication of RFC 2490 the commercial OPNET models have been expanded to include more protocols of interest in IntServ and DiffServ. By making use of the OPNET versions it is possible to make the modeling process easier for potential users and also to improve the performance of the resulting simulations. We have therefore prepared an updated version of the models described in RFC 2490, which once again is open source and also works with the latest version of OPNET. Our revised models are presented in this document. They are available for download at http://netlab.gmu.edu/qosip.

2. The OPNET Simulation Environment

The OPNET Modeler (OPNET) is a commercial simulation product of OPNET Technologies, Inc. of Bethesda, Maryland. It employs, among other techniques, a Discrete Event Simulation approach that allows large numbers of closely-spaced events in a sizable network to be represented accurately and efficiently. OPNET uses a modeling approach where networks are built of components interconnected by perfect links that can be degraded at will. Each component's behavior is modeled as a state-transition diagram. The process that takes place in each state is described by a program in the C/C++ language. We believe this makes the OPNET-based models relatively easy to port to other modeling environments. Use of this environment also is attractive for unsponsored academic research in that the OPNET Modeler is available at no charge for educational purposes. Thus makes it graduate students can afford to pursue analysis of leading-edge Internet protocols.

The family of models described here is compatible with OPNET 8.0. The following sections describe the new models from OPNET that replace models presented in RFC 2490, and new state-transition models and process code we have created to analyze IntServ and DiffServ proposals using RSVP and MPLS. Please note that an OPNET layer is not necessarily equivalent to a layer in a network stack, but shares with a stack layer the property that it is a highly modular software element with well defined interfaces. Details of OPNET 8.0 modeling can be found in [8,9].

2.1 MOSPF

OPNET 8.0 supports the PIM-SM multicast protocol and uses it for multicast applications by default. It also provides a "hook" that facilitates user development of multicast protocols, which our MOSPF process model uses. In the remainder of this section, we discuss the OPNET 8.0 process models that collectively perform multicast. We give special attention to the interfaces by means of which a custom-built multicast routing protocol interacts with the OPNET processes. We also discuss the enhancements required for these process models to work with multicast routing protocols, such as DVMRP and MOSPF, that do not explicitly establish multicast routes a priori by means of control messages but rather trigger route computations by the arrival of multicast messages. All the enhancements discussed here are included in our simulation model for public download.

2.2 IP and Multicast Models

In OPNET 8.0, two process models implement IP layer functions: ip_dispatch and ip_encap_v4. The former performs typical network layer functions such as routing and packet forwarding. The latter provides mainly a multiplexing/demultiplexing interface between ip_dispatch and higher-level protocols. For sake of brevity, we describe a high-level protocol "connected to IP," meaning the configuration where the protocol process accesses ip_dispatch by connecting to ip_encap_v4 using packet streams. In particular, multicast routing processes must connect to IP to exchange Control messages with their counterparts on peer routers.

Many multicast-related features in OPNET 8.0 are implemented as child processes of ip_dispatch. Such a process will be referred to as an "IP child process" hereafter. For example, custom-built multicast protocols are supported via the ip_custom_mrp IP child process. For a simulation model configured to use a custom multicast protocol, ip_dispatch invokes the ip_custom_mrp child process to perform multicast routing for any packet destined for a class D address, that is, to search for the address in the multicast routing table and return the interface(s) on which the packet must be forwarded. However, the ip_custom_mrp process model does not perform multicast routing computations by itself. We have revised the ip_custom_mrp process so that when the destination group is not found, the process produces a remote interrupt to the MOSPF process for routing computations. The current multicast packet is discarded in this case. It is the responsibility of the MOSPF process (or any multicast routing protocol other than PIM-SM) to add the group to the routing table.

2.3 IGMP Process Models

IGMP is implemented as two IP child process models in OPNET 8.0: ip_igmp_rte_intf and ip_igmp_rte_grp. For each multicast-enabled network attached to a router, an ip_igmp_rte_intf child process is created to perform IGMP functions on the interface to that network. For a multicast address that has members on a multicast-enabled interface, an ip_igmp_rte_grp child process is created. IGMP processes communicate with PIM-SM multicast protocol by default. We have revised the ip_igmp_rte_grp implementation so that it also communicates using remote interrupts with a custom multicast process (namely, MOSPF) for the generation of new groups and destruction of existing groups.

2.4 RSVP Process Model

The RSVP protocol is implemented by the rsvp process model in OPNET 8.0 (please notice the use of upper case letters in the protocol name and the use of lower case letters in the process name). The rsvp process communicates with IP to exchange RSVP control messages with peer routers. To reserve bandwidth on a local link, the rsvp process communicates with the ip_output_iface process of the link. An ip_output_iface process is an IP child process created for each output interface of the router to manage the buffers/queues and bandwidth of that interface. The rsvp process model provided in OPNET 8.0 communicates only with the default multicast protocol, PIM-SM, for information about bandwidth availability. We have revised the process model so that it will also inform the MOSPF process of dynamics in resource availability, enabling QoS-aware multicast routing in the future.

2.5 Multicast Application Models

OPNET 8.0 includes a general-purpose client-server application process model, called gna_clsrv_mgr, that supports many application layer protocols, such as FTP, HTTP, Telnet, etc. The model supports multicast in its voice and video applications. The two multicast-enabled applications communicate with IGMP and RSVP processes on the local workstation and does not deal directly with multicast routing processes on routers.

The video application of gna_clsrv_mgr supports the following configuration parameters: the distribution of frame interarrival times, the distribution of frame sizes, type of service, and RSVP parameters. The voice application supports the following configuration parameters: the distribution of silence length, the distribution of talk spurt length, voice encoding scheme (GSM, g.723.1, etc.), type of service, and RSVP parameters. In either case, RSVP parameters include bandwidth (in bytes/second) and buffer size.

Both voice and video applications of the gna_clsrv_mgr process model employ a simple handshaking process. In this process, a voice/video source sends a SETUP message to the destination address, which could be of class D, and expects at least one CONNECT message from a receiver. If the destination address is of class D, each receiver must reply with a CONNECT message. However, one CONNECT message is sufficient to enable the source to start the transmission of voice/video packets; subsequent CONNECT messages are discarded by the source.

This handshaking process, however, is incompatible with the MOSPF protocol, where multicast routing computations are triggered by the arrival of multicast packets. With MOSPF, when a packet destined for a multicast group arrives at a router that does not have routing entries corresponding to the group, the router performs the OSPF Dijkstra's algorithm to compute a multicast tree rooted at the source of the packet and spanning all member interfaces of the group. The packet itself is discarded in this case. Because the very first multicast packet, the SETUP message, is bound to encounter routers without routing entries corresponding to its newly created multicast group, the message will be discarded, and the handshaking process cannot be completed. To deal with this problem, we revised the gna_clsrv_mgr model so that a voice/video source periodically retransmits its SETUP message until a CONNECT message is received.

2.6 Creation of Multicast Routing Processor Node

Interfacing a multicast routing protocol using the OPNET 8.0 Simulation package requires the creation of a new routing process in the node editor and linking it via packet streams. Packet streams are unidirectional or bidirectional links used to interconnect processors, queue nodes, transmitters and receivers.

A multicast routing processor is created in the node editor and links are created to and from the processor (duplex connection) that interact with this module ip_encap_v4. Communication between the multicast routing processor and IGMP/RSVP is achieved through remote interruptions; no packet streams are used for this purpose. Within the node editor, a new processor can be created by selecting the button for processor creation (plain gray node on the node editor control panel) and by clicking on the desired location in the node editor to place the node. Upon creation of the processor in OPNET, the name of the processor can be specified by right clicking on the mouse button and entering the name value in the attribute box presented. Links to and from this node are generated by selecting the packet stream button (represented by two gray nodes connected with a solid arrow on the node editor control panel), left clicking on the mouse button first to specify the source of the link and second to mark the destination of the link.

2.7 MOSPF Interaction with Other Protocols

The interrupt mechanism of the OPNET simulation environment forms the foundation of the interface between the MOSPF process and its surrounding processes. Using OPNET's paradigm, the arrival of a packet at a process via a packet stream also constitutes an interrupt to the process; such interrupts are called stream interrupts. Other important types of interrupts include self-interrupts, whereby a process schedules an event for itself, and remote interrupts, whereby a process schedules an event for another process. Stream interrupts are distinguished by the the index of the stream through which the interrupt/packet arrives. Self-interrupts and remote interrupts are distinguished by an integer type code that is delivered with the interrupt. Each interrupt is further associated with a set of attributes, collectively called the Interface Control Information (ICI) of the interrupt. The particular set of attributes associated with a given type of interrupt is defined by the creator of the simulation model using the ICI editor.

In this section, we discuss the interrupts that the MOSPF process model is programmed to understand. However, we do not include stream interrupts in the discussion because the information delivered by such interrupts are contained in the associated packets, which are MOSPF control messages in our simulation model. The formats and uses of MOSPF control messages are defined in [11].

2.7.1 IGMP to MOSPF Interrupts

Upon the creation of an ip_igmp_rte_grp process pertaining to a particular multicast group on a multicast-enabled interface, the process sends to MOSPF a remote interrupt with type code QospfC_Intrpt_Group_Activated to inform MOSPF of the activation of the group on that interface. This happens when the first host member on that interface joins the group. Upon its destruction, the ip_igmp_rte_grp process sends a remote interrupt with type code QospfC_Intrpt_Group_Deactivated to inform MOSPF of the deactivation of the group on that interface. This happens after all host members on that interface have left the group. The ICI of the above two types of interrupts comprises the "group_addr" and "interface_index" attributes.

2.7.2 RSVP to MOSPF Interrupts

When the rsvp process reserves/releases network resources on an (output) interface, it informs MOSPF of the changes in resource availability by sending a remote interrupt with type code QospfC_intrpt_Resource_Reserved/QospfC_intrpt_Resource_Released. The ICI of the two types of interrupts comprises the "src addr", "dest addr", "src port", "req bandwidth", and "req link delay" attributes.

2.7.3 IP to MOSPF Interrupts

As described earlier, in a simulation model configured to use custom multicast protocols, the ip_dispatch process invokes the ip_custom_mrp child process to perform routing table lookup operations for packets destined for class D addresses. If the given multicast address is not found in the multicast routing table, the ip_custom_mrp process sends a remote interrupt to MOSPF with type code QospfC_Intrpt_Ip_Notif. The ICI associated with such an interrupt comprises the "source addr", "dest group addr", "in major port", and "in minor port" attributes.

3. OSPF and MOSPF Models

OSPF and MOSPF [9] protocols are implemented in the OSPF model, containing fourteen states. They only exist on routers. Figure 1 shows the process model. The following processing takes place in the indicated modules.

3.1 init

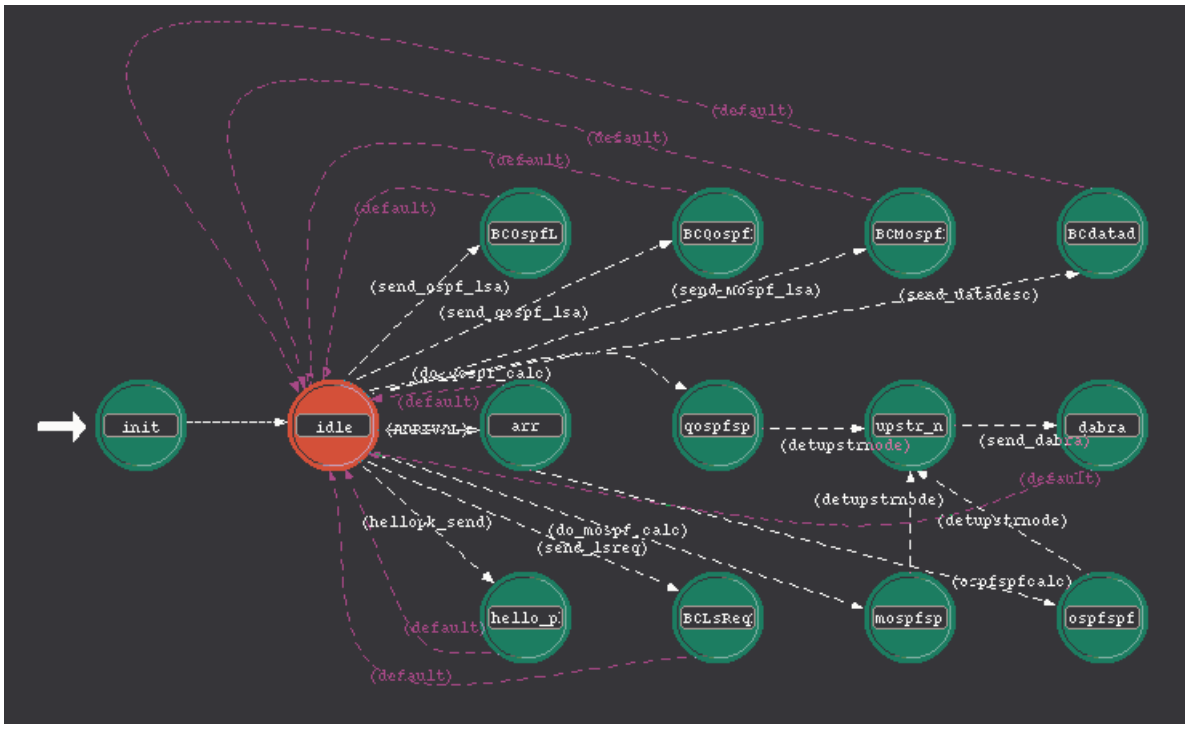
This state initializes all the router variables. Default transition to idle state.

3.2 idle

This state has several transitions. If a packet arrives or a remote interrupt is received from IGMP or RSVP, it transits to arr state. For remote interrupts from ip_custom_mrp and self interrupts, depending on the type and code of the interrupt, it will transit to BCOspfLsa, BCMospfLsa, or hello_pks state. In future versions, links coming up or down will also cause a transition.

3.3 BCOspfLsa

Transition to this state from idle state is executed whenever the condition send_ospf_lsa is true, which happens when the network is being initialized, and when ospf_lsa_refresh_timout occurs (causing a self interrupt). This state will create Router, Network, and Summary Link State Advertisements and pack all of them into an Link State Update packet. The Link State Update Packet is sent to the IP layer with a destination address of AllSPFRouters.



[Figure 1: OSPF and MOSPF process model on routers]

3.4 BCMospfLsa

Transition to this state from idle state is executed whenever the condition send_mospf_lsa is true. This state will create Group Membership Link State Advertisement and pack them into MOSPF Link State Update Packet. This MOSPF Link State Update Packet is sent to IP layer with a destination address of AllSPFRouters.

3.5 arr

The arr (arrival) state is entered upon the arrival of a packet or interrupt. Thus we have two cases:

Case A. The function OspfPkPro is called if the state is entered due to the arrival of a packet. The packet must be OSPF/MOSPF control message. Depending on the type of the packet, OspfPkPro further calls one of the following functions.

1. HelloPk_pro: This function is called whenever a hello packet is received. This function updates the router's neighbor information, which is later used while sending the different LSAs.
2. OspfLsUpdatePk_pro: This function is called when an OSPF LSA update packet is received (router LSA, network LSA, or summary LSA). If the Router is an Area Border Router or if the LSA belongs to the Area whose Area Id is the Routers Area Id, then it is searched to determine whether this LSA already exists in the Link

State database. If it exists and if the existing LSA's LS Sequence Number is less than the received LSA's LS Sequence Number the existing LSA was replaced with the received one. The function processes the Network LSA only if it is a designated router or Area Border Router. It processes the Summary LSA only if the router is a Area Border Router. The function also turns on the trigger ospfspfcalc which is the condition for the transition from arr state to ospfspfcalc.

3. MospfLsUpdatePk_pro: This function is called when a MOSPF LSA update packet is received. It updates the group membership link state database of the router.

Case B. An interrupt is received. One of the following procedure is invoked according to the code of the interrupt.

1. IgmpICI_pro:
   This code segment processes the interrupts from IGMP (specifically, from an ip_igmp_grp process with code QospfC_Intrpt_Group_Activated/Deactivated). It first marks the corresponding entry in the multicast routing table, causing a recomputation of the multicast tree for the group. Then it schedules a self-interrupt for the broadcasting of MOSPF LSAs.
2. RsvpPk_pro:
   This function is called when a remote interrupt from the rsvp process is received (with code QospfC_intrpt_Resource_Reserved/Released). Currently, it only updates the local network image of the router for changes in resource availability. In the future, the function can trigger multicast routing recomputations for QoS-aware multicast applications.

3.6 hello_pks

Hello packets are created and sent with destination address of AllSPFRouters. Default transition to idle state.

3.7 mospfspfcalc

The following functions are used to calculate the shortest path tree and routing table. This state transit to upstr_node upon detupstrnode condition.

a. CandListInit: Depending upon the SourceNet of the datagram, the candidate lists are initialized.

b. MospfCandAddPro: The vertex link is examined and if the other end of the link is not a stub network and is not already in the candidate list it is added to the candidate list after calculating the cost to that vertex. If this other end of the link is already on the shortest path tree and the calculated cost is less than the one that shows in the shortest path tree entry update the shortest path tree to show

the calculated cost.

c. MospfSPFTreeCalc: The vertex that is closest to the root that is in the candidate list is added to the shortest path tree and its link is considered for possible inclusions in the candidate list.

d. MCRoutetableCalc: Multicast routing table is calculated using the information of the MOSPF shortest Path tree.

3.8 ospfspfcalc

In this state, the following functions are used to calculate the shortest path tree. This information is used in the routing table. Transition to ospfspfcalc state on ospfcalc condition, which is set to one after processing all functions in the state.

a. OspfCandidateAddPro: This function initializes the candidate list by examining the link state advertisement of the Router. For each link in this advertisement, if the other end of the link is a router or transit network and if it is not already in the shortest-path tree then calculate the distance between these vertices. If the other end of this link is not already on the candidate list or if the distance calculated is less than the value that appears for this other end add the other end of the link to candidate list.

b. OspfSPTreeBuild: This function pulls each vertex from the candidate list that is closest to the root and adds it to the shortest path tree. In doing so it deletes the vertex from the candidate list. This function continues to do this until the candidate list is empty.

c. OspfStubLinkPro: In this procedure the stub networks are added to shortest path tree.

d. OspfSummaryLinkPro: If the router is an Area Border Router the summary links that it has received is examined. The route to the Area border router advertising this summary LSA is examined in the routing table. If one is found a routing table update is done by adding the route to the network specified in the summary LSA and the cost to this route is sum of the cost to area border router advertising this and the cost to reach this network from that area border router.

e. RoutingTableCalc: This function updates the routing table by examining the shortest path tree data structure.

3.9 upstr_node

This state does not do anything in the present model. It transitions to DABRA state.

3.10 DABRA

If the router is an Area Border Router and the area is the source area then a DABRA message is constructed and send to all the downstream areas. Default transition is to idle state.

3.11 QOSPF

The models described in RFC 2490 were developed originally in the context of studying performance of a proposed QoS version of OSPF (QOSPF). We have retained only those functions of QoSPF necessary to compare performance of our new models with those described in RFC 2490.

4. Modeling DiffServ-Enabled Multicast over MPLS Tunnels

We built a set of simulation models for the study of DiffServ + MPLS + Multicast networking environments. These models integrate PIM-SM multicasting with expedited services over MPLS tunnels. They are based on OPNET 8.0, which is the first OPNET version that supports MPLS.

4.1 Architectural Changes in OPNET 8.0

In OPNET 8.0, IP layer functions have been reorganized extensively. Unlike OPNET 7.0 where a central process (namely ip_dispatch) implements most of the IP functions, in the new version a new central process, called ip_dispatch, serves only to spawn processes that implement the features (MPLS, traffic markers, etc.) configured by the users; all IP functions are now implemented in the child processes of ip_dispatch. This new architecture allows for greater efficiency, modularization, flexibility, and easy accommodation of new Internet technologies such as MPLS. In this section, the term "IP child process" refers to those processes spawned by ip_dispatch.

4.2 OPNET 8.0 MPLS Model

Multi-Protocol Label Switching (MPLS) uses labels to determine how packets are forwarded through a network. The MPLS model is part of OPNET 8.0 model library and is implemented collectively by the following process models: mpls_mgr, mpls_ldp_mgr, mpls_discovery_mgr, mpls_session_mgr and mpls_lsp_mgr.

An instance of mpls_mgr is spawned by ip_dispatch on each MPLS enabled router. It represents the forwarding component of MPLS and the forwarding control plane. When the IP module of an LSR receives labeled packets or packets with matching FEC descriptions, it performs no IP processing on the packet. Instead, the packet is redirected to the mpls_mgr process for MPLS forwarding. The mpls_mgr process is the only MPLS process directly related to the simulation models described in this section. The other models are introduced briefly below.

The mpls_ldp_mgr IP child process implements the LDP control plane in the LDP module of all routers. It also spawns the mpls_discovery_mgr, mpls_session_mgr, and mpls_lsp_mgr processes, if the features provided by these processes are needed. The mpls_discovery_mgr process sends periodic broadcast hello messages over UDP to discover MPLS enabled neighboring routers. The mpls_session_mgr process, based on RFC 3036, negotiates, opens and maintains TCP sessions to neighboring LDP routers. The mpls_lsp_mgr process controls the exchange to label mapping between LDP peers. Communication with LDP peers occurs through the session established by the mpls_session_mgr process.

4.2.2 MPLS Configuration

MPLS configuration can be made through global MPLS attributes, router-specific MPLS attributes, and LSP attributes. Global MPLS attributes, which are used to configure network-wide MPLS parameters, are grouped in the MPLS configuration object. Important attributes include:

a. FEC Specifications: Each FEC is specified source/destination addresses/ports and ToS.

b. Traffic Trunk Profiles: A traffic trunk profile is specified by maximum bit rate, average bit rate, maximum burst size, traffic class, and out-of-profile action (unchanged, discarded, and degraded (change of traffic classes).

c. Exp bits to drop precedence mappings

d. Exp bits to PHB mappings: Router-specific attributes include LDP Parameters Discovery Configuration, Session Configuration, Recovery Configuration, and Traffic Engineering Configuration. Traffic Engineering Configuration specifies the bindings between FECs, traffic trunks, and LSPs. LSP attributes include LSP Type, Path Details, Directionality, Recovery Parameters, Setup Parameters and Traffic Engineering Parameters.

4.3 Diffserv Features

OPNET 8.0 supports various queueing schemes including FIFO, Priority Queueing, WFQ, and Custom Queueing. Moreover, multiple profiles can be defined for each queueing scheme through IP QoS Configuration object in order to meet certain DiffServ requirements. Profiles can be ToS-based, Protocol-based, Port-based, DSCP-based, and so forth. Each profile defines a packet-to-queue mapping to classify outgoing packets. For MPLS forwarded packets, DSCP-based queueing profile is employed at the output interface. Each interface of a router can be configured one of profiles described above.

4.4 Traffic Marker

Traffic markers are used in routers to control and shape incoming traffic. Two kinds of markers, Single Rate Three Color Marker (srTCM) and Two Rate Three Color Marker (trTCM), based on RFC 2697 [12] and RFC 2698 [13] respectively, are implemented as two IP child process models: ip_rte_srtcm and ip_rte_trtcm. The two process models are extensions to 8.0 Model library (implemented by GMU) and are available for public download.

Parameters of traffic markers are configured for each router independently. All of the parameters fall in two categories: Flow Specification and Traffic Specification. When a packet arrives at a srTCM or trTCM enabled router and matches a flow specification, the marker configured for the flow is invoked to process the packet.

4.4.1 ip_rte_srtcm

This process meters an incoming IP packet stream and marks its packets green, yellow, or red. Marking is based on a Committed Information Rate (CIR) and two associated burst sizes, a Committed Burst Size (CBS) and an Excess Burst Size (EBS). A packet is marked green if it doesn't exceed the CBS, yellow if it exceeds the CBS, but not the EBS, and red otherwise. Red packets are discarded immediately, while green and yellow packets are further processed, queued and forwarded. Yellow packets will have higher drop precedence than green ones when congestion occurs in the downstream path. The ip_rte_srtcm process model has three states:

a. init: The process enters this state when it is created and invoked for the first time. Process parameters such as CIR, CBS, EBS and the flow index are passed from the parent process. Various statistics handles are registered. Token buckets are set to the initial values. The process then enters the "idle" state

b. idle: The process awaits packets in the idle state. Upon the arrival of a packet, the process enters the "mark" state.

c. mark To mark a packet, the token buckets are updated first according to the srTCM parameters (CIR, CBS and EBS) and the simulation time elapsed since last update. Second, the packet is marked according to RFC 2697, and statistics are updated. The process then returns to the Idle state.

4.4.2 ip_rte_trtcm

This process meters an IP packet stream and marks its packets based on two rates, Peak Information Rate (PIR) and Committed Information Rate (CIR), and their associated burst sizes to be either green, yellow, or red. A packet is marked red if it exceeds the PIR. Otherwise it is marked either yellow or green depending on whether it exceeds or

doesn't exceed the CIR. The marked packets are treated exactly same as those in ip_rte_srtcm process. The ip_rte_srtcm process model also has three states:

a. init: The process enters this state when it is created and invoked for the first time. Process parameters such as CIR, CBS, PIR, PBS, and the flow index are passed from the parent process. Various statistics handles are registered. Token buckets are set to the initial values. The process then enters the "idle" state

b. idle: The process awaits packets in the idle state. Upon the arrival of a packet, the process enters the mark state.

c. mark: To mark a packet, the token buckets are updated first according to the srTCM parameters (CIR, CBS PIR, and PBS) and the simulation time elapsed since last update. Second, the packet is marked according to RFC 2698, and statistics are updated. The process then returns to the idle state.

4.5 Simulation Models

We have built wo simulation models (called Projects in OPNET terminology) using the above features for the study of DiffServ-enabled PIM-SM multicast over MPLS tunnel. They are DebugModel and Routers42. Shown in the above figure is the DebugModel. The node models at the top layer are either LERs (Label Edge Router) or LSRs (Label Switched Router). Two static LSPs are configured: LER 3 --> LSR 1 --> LER 1, and LER 3 --> LSR 2 --> LER 2. LER 3 are configured with traffic marker and MPLS traffic engineering so that video conferencing streams in the model are bound to a flow specification that uses LSPs with PHB = EF (EXP = 6 or 7) for transmission. The model we have provided supports two scenarios to study the effects of DiffServ/MPLS features. (The "scenario" feature of OPNET enables different predefined configurations and event sequences within one simulation model).

The first scenario supports both DiffServ and MPLS with PIM-SM multicast. All outgoing interfaces of LERs and LSRs along the LSPs use DSCP-based priority queueing. Packets with PHB = EF are queued in queue 4 (queue 0 is for best effort traffic, queues 1 to 3 not used) which is configured in the QoS Configuration object. Queues with higher index have higher priority.

Two applications are used in the simulation. One is video conferencing; the other is file transfer which provides background traffic. File transfer packets do not enter LSPs and are of lower priority; they are queued in queue 0 along LSRs. The total traffic is set such that the utilization of backbone links is almost 100%.

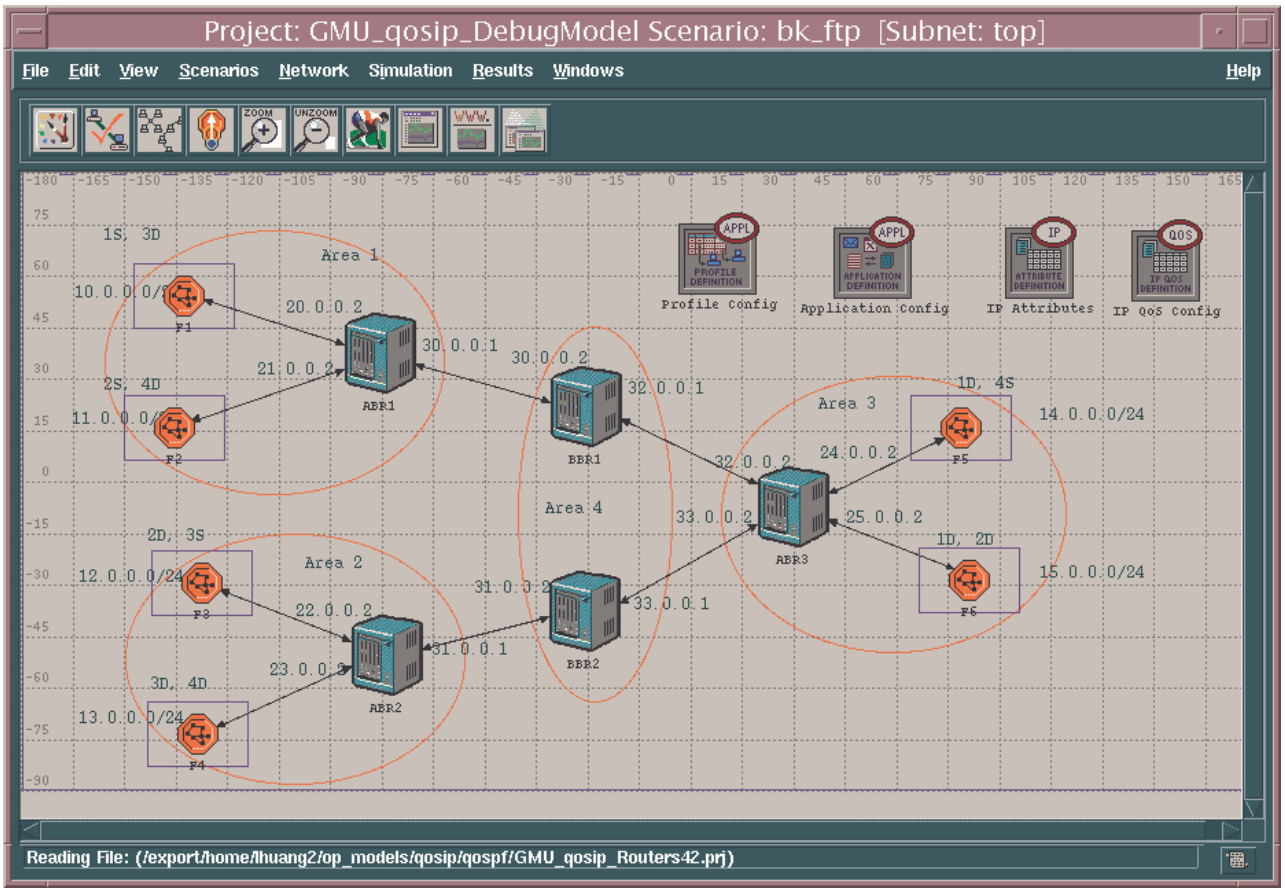
For purposes of comparison, a second scenario is included. This version operates without DiffServ, MPLS and priority queueing, so that videoconferencing packets and file transfer packets compete for

capacity equally. This model shows the expected impact on video QoS, caused by competition for network capacity.

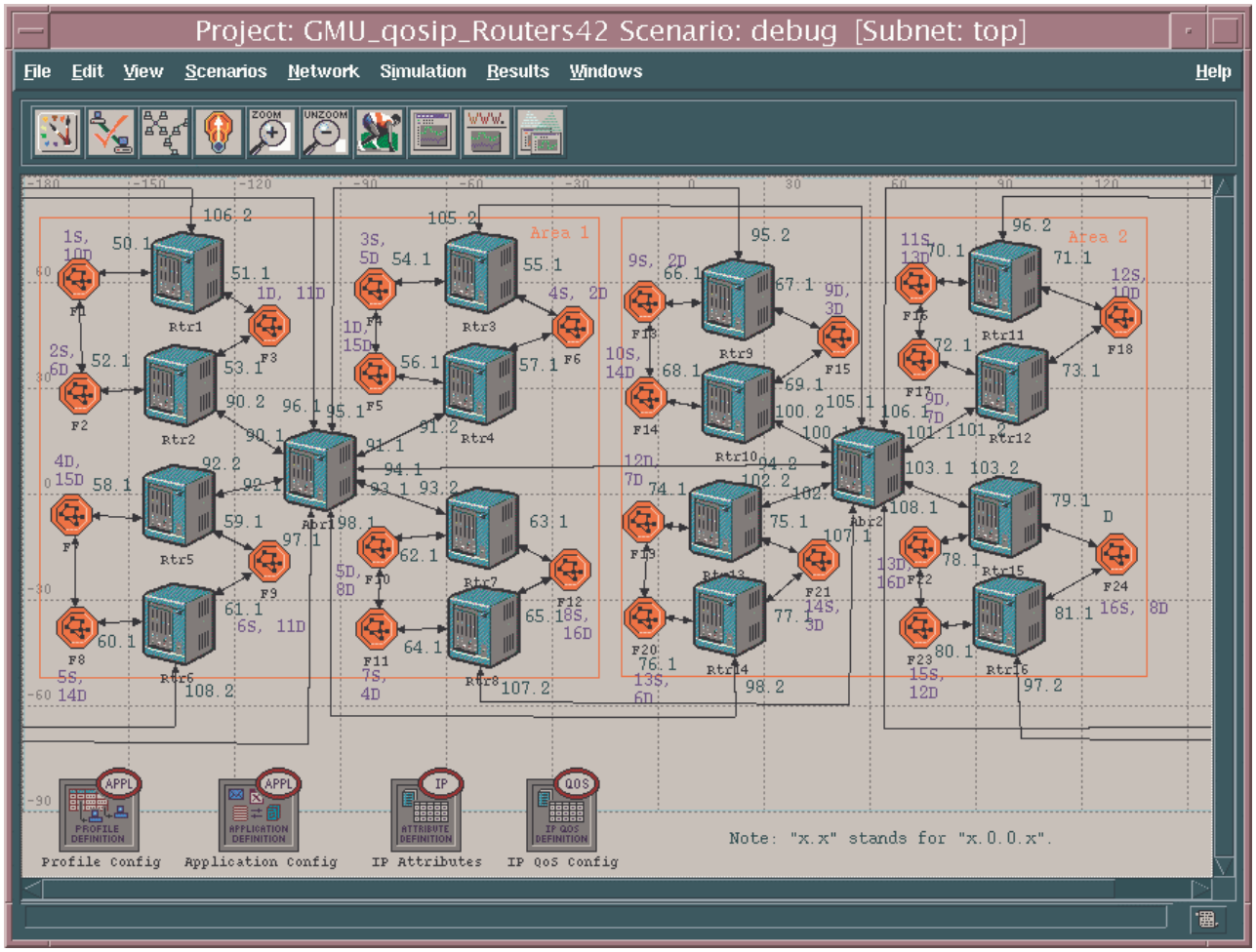
5. Example of Use

5.1 Network Models

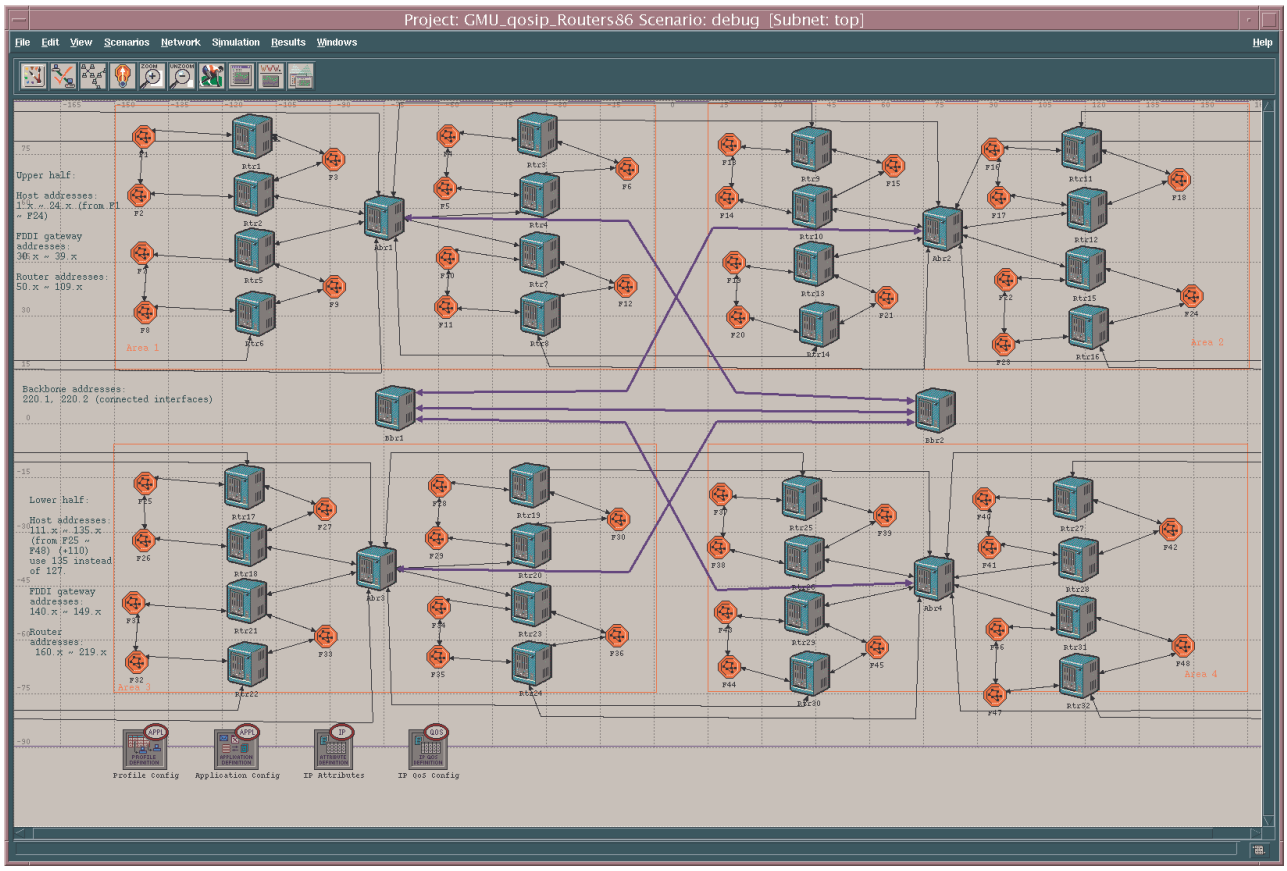
Here we give an example of using the QoS protocol models. We have developed three network configuration that exercise the models. Figure 2 shows the Debug model, which is too small for practical performance predictions but can be used to test whether protocols are working correctly, with minimum run time. Figure 3 shows the Intermediate model, which is large enough to draw some conclusions about performance, while Figure 4 shows the Large model, which is intended to be typical of a sizable corporate Intranet or regional ISP operation. While the Large model takes significantly longer to run, we believe its results are the most credible in terms of depicting QoS network performance.



[Figure 2: Debug Network Model]



[Figure 3: Intermediate Network Model]

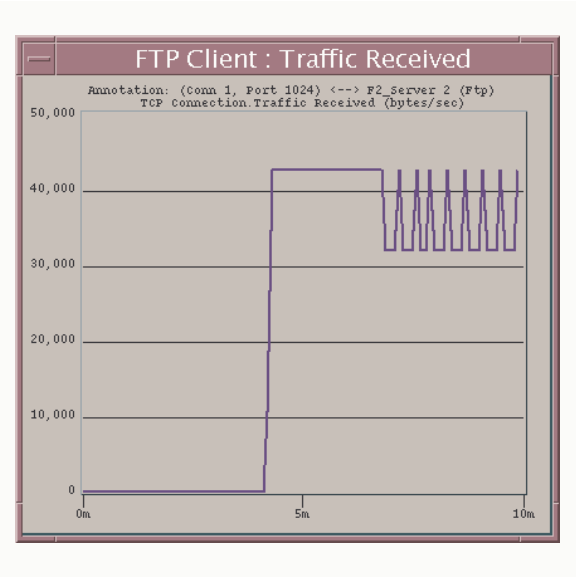


[Figure 4: Large Network Model]

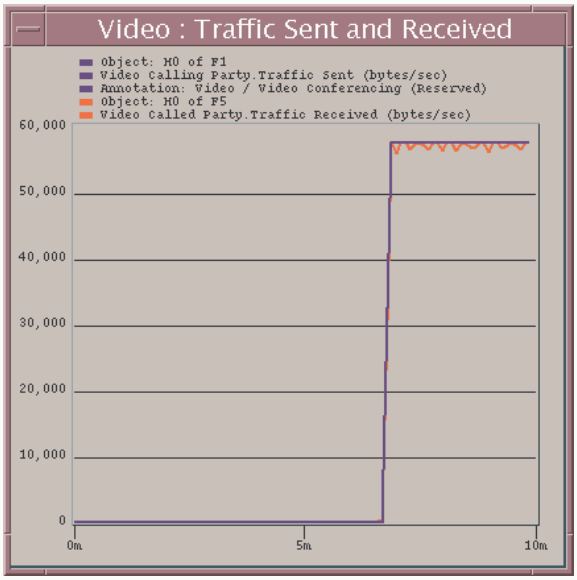
5.2 IntServ Simulation

We applied the IntServ protocols to a situation where videoconferencing is taking place over a network heavily loaded with background traffic. Hosts join and leave the conferences at random. Each connected host produces a video packet every 10 milliseconds in addition to the background FTP traffic. The resulting network performance is shown in Figures 5 through 8. Figure 5 shows the traffic volume of an FTP transfer from a server in LAN F5 to a client in LAN F1. The transfer starts at time 340 second and continues throughout the simulation. The video multicast traffic starts at 400 second. With the TCP congestion control mechanisms, FTP traffic was scaled back after appearance of video traffic at time 400. As shown in Figure 6, video traffic, protected by RSVP-based IntServ, suffers few packet losses in the presence of FTP traffic. The utilization of the ABR1—BBR1 link throughout the simulation is plotted in Figure 7. As shown, the FTP traffic is not able to use all the residual bandwidth both before and after the emergence of video traffic. This is owing to the standard, but insufficiently large in this case, receiver window size of 8760 bytes. In Figure 8, it can be seen that QoS traffic delay is controlled under

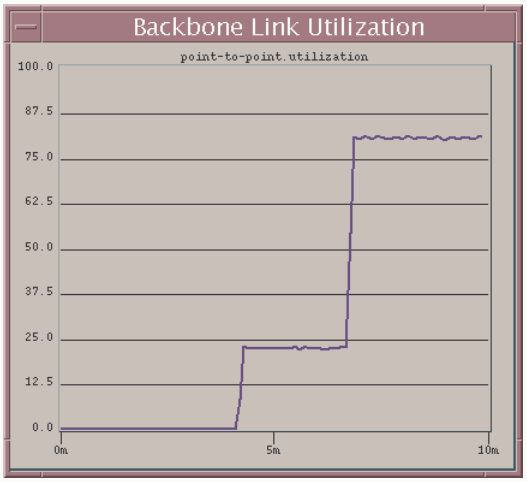
IntServ in the presence of background traffic, when the link in question is heavily but not fully loaded.



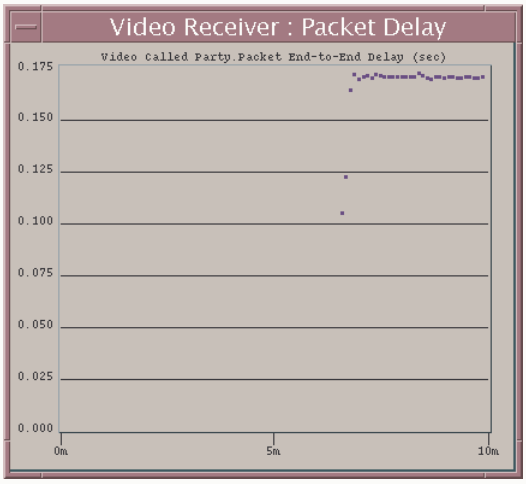
[Figure 5: Background Traffic Loading]



[Figure 6: QoS Traffic Loading]



[Figure 7: Network Utilization]



[Figure 8: QoS Traffic Delay]

Our simulations of three network models with MOSPF routing showed the Simulation to have good scalability. The simulation platform we used is a Sun Ultra 60 Workstation with 512 MB main memory an UltraSparc 440 MHz processor. The performance is greatly improved from that reported in RFC 2490. Here we list the real running time of each model along with its major elements and the packet inter-arrival times for the streams generated in the hosts. Please notice that the simulation times of 100

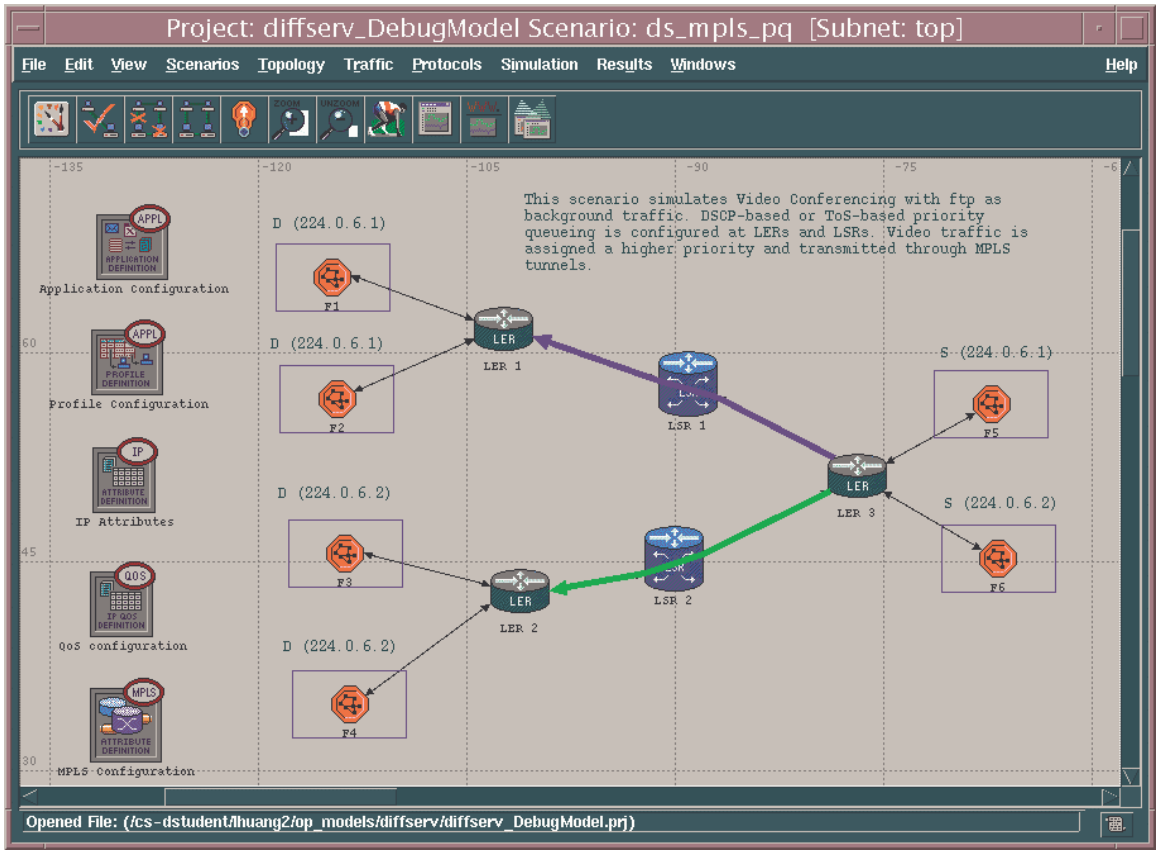
or 200 seconds are the elapsed times since the beginning of video traffic, rather than from time 0. Our experiences show that the events before video traffic, mainly composed of routing control traffic and a short period of FTP traffic, require little computation. Thus the majority of the simulation time is spent in video activities. Table 1 below should give a better estimate of scalability, compared to measuring times from time 0.

Simulated time	Debug Model 11 Routers, 12 Hosts	Intermediate Model 42 routers, 48 hosts	Large Model 86 routers, 96 hosts
100 sec	11 min	35 min	119 min
200 sec	22 min	87 min	229 min

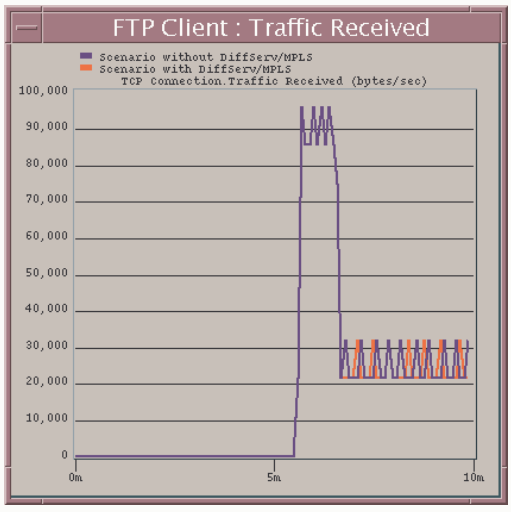
[Table 1: IntServ Simulation Run Times on Ultra 60]

5.3 DiffServe Simulation

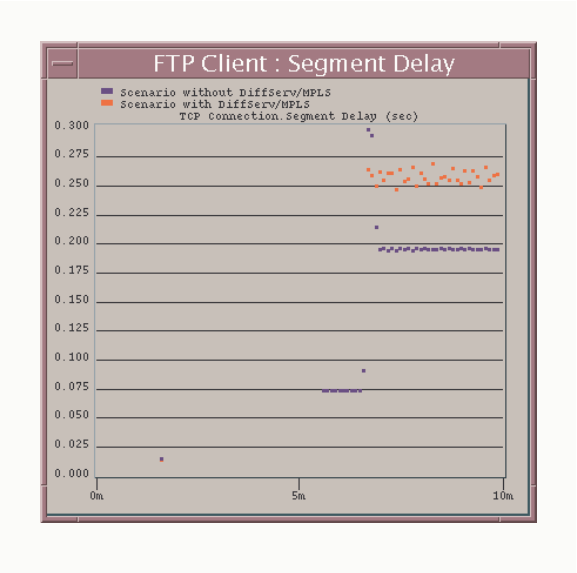
To demonstrate modeling DiffServ, we adopted a similar scenario adapted to the differences in the emerging QoS environment for DiffServ. Specifically, we made use of MPLS for backbone trunks, and employed traffic marking as described in section 4 above. Figure 9 shows the Debug model for this case (the Intermediate and Large models were scaled up using the same MPLS substitution). The resulting network performance is shown in Figures 10 through 13. Figure 10 shows the traffic volume of an FTP transfer from a server in LAN F5 to a client in LAN F1. The transfer starts at time 340 seconds and continues throughout the simulation. The video multicast traffic starts at 400 seconds. With the TCP congestion control mechanisms, FTP traffic was scaled back after appearance of video traffic at time 400 seconds. Compared to the scenario where video traffic is not supported by DiffServ/MPLS, the FTP transfer suffers long delays (see Figure 11). In terms of bandwidth usage, video traffic shows little differences with or without DiffServ/MPLS, as seen in Figure 12. However, Figure 13 shows that the use of DiffServ/MPLS does improve video end-to-end delays significantly and thus provide a consistent, low delay for the QoS video traffic.



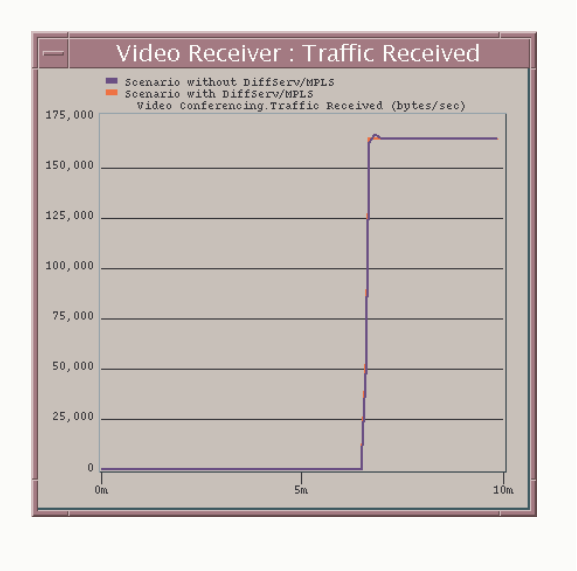
[Figure 9: DiffServ Debug Model with MPLS Trunks]



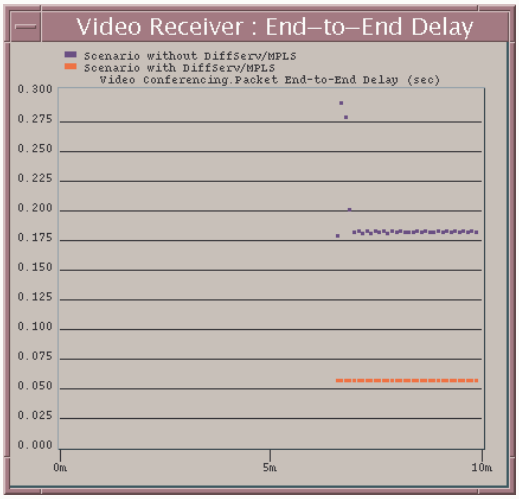
[Figure 10: Background (ftp) Traffic Load]



[Figure 11: Background (ftp) Traffic Delay]



[Figure 12: QoS (video) Traffic Load]



[Figure 13: QoS (video) Traffic Delay]

We used the same computer configuration (Ultra 60) for the DiffServ Simulation, however the simulation times must be considered somewhat differently. In this simulation there is considerable network setup activity in the beginning, which does not impose much of a load on the simulation computer. The background traffic (file transfer starts at 340 seconds and the video application starts at 400 seconds, with a transmit rate of 80 kB/sec. Therefore, in table 2 the Simulation Time is measured from 400 seconds onward.

Simulated time	Debug Model 11 Routers, 12 Hosts	Intermediate Model 42 routers, 48 hosts	Large Model 86 routers, 96 hosts
400 sec	.67 min	.70 min	1.0 min
600 sec	18 min	89 min	212 min
800 sec	35 min	179 min	425 min

[Table 2: DiffServ Simulation Run Times on Ultra 60]

6. Future work

We hope to receive feedback and assistance from the Internet QoS Development community within the IETF in validating and refining these models. We believe they will be useful tools for predicting the behavior of IntServ and DiffServ QoS systems, using new protocol and traffic engineering proposals.

7. Security Considerations

This RFC raises no security considerations.

8. References

[1] Deering, S., "Host Requirements for IP Multicasting", STD 5, RFC 1112, August 1989.

[2] Braden, R., Clark, D. and Shenker, S., "Integrated Services in the Internet Architecture: an Overview." June 1994.

[3] Blake, S. et. al., "An Architecture for Differentiated Services", RFC 2475, December 1998.

[4] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource Reservation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[5] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

[6] Rosen, E., Viswanathan, A., and Callon, R., "Multiprotocol Label Switching Architecture, " RFC 3031, January 2001.

[7] Pullen, M., et. al., "A Simulation Model for IP Multicast with RSVP," RFC 2490, January 1999.

[8] OPNET Technologies Inc., "OPNET Modeler Modeling Manual", Bethesda, MD, release 8.0.c, June 2001

[9] OPNET Technologies Inc., "OPNET Protocol Model Documentation", Bethesda, MD, release 8.0.c, June 2001

[10] Moy, J., "OSPF Version 2", RFC 2328, April 1998.

[11] Davie, B., et. al., "MPLS Support of Differentiated Services, " work in progress

[12] Heinanen, J. and Guerin, R., "A Single Rate Three Color Marker, " RFC2697, September 1999

[13] Heinanen, J. and Guerin, R., "A Two Rate Three Color Marker, " RFC2698, September 1999

Authors' Addresses

J. Mark Pullen
C3I Center/Computer Science Dept
Mail Stop 4A5
George Mason University
Fairfax, VA 22032

EMail: mpullen@gmu.edu

Yih Huang
Computer Science Dept
Mail Stop 4A5
George Mason University
Fairfax, VA 22032

EMail: huangyih@gmu.edu

Pradeep Singh
OPNET Technologies Inc
7255 Woodmont Avenue
Bethesda, MD 20814

EMail: psingh@opnet.com

Leijun Huang
Computer Science Dept
Mail Stop 4A5
George Mason University
Fairfax, VA 22032

EMail: lhuang2@gmu.edu

Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of

developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.