An End-to-End QoS Framework with On-Demand Bandwidth Reconfiguration

Mei Yang, Yan Huang, Jaime Kim, Meejeong Lee, Tatsuya Suda and Mastubara Daisuke
{meiy, yanh, suda}@ics.uci.edu; jkim@ecs.csun.edu; lmj@mm.ewha.ac.kr; daisuke.matsubara@hal.hitachi.com

Abstract--- **This paper proposes a new QoS framework, called On-Demand QoS Path (ODP), which provides end-to-end QoS guarantees to individual flows with minimal overhead, while keeping the scalability characteristic of DiffServ. ODP exercises per-flow admission control and end-to-end bandwidth reservation at the edge of the network and only differentiates service types in the core of the network. In addition, to be adaptive to traffic load, ODP monitors the bandwidth utilization status of the network and performs dynamic bandwidth reconfiguration in the core based on the network status. Through extensive simulations, the performance of proposed architecture is investigated and compared with that of IntServ and DiffServ architecture. The simulation results clearly showed that ODP could provide end-to-end QoS guarantees to individual flows with much less overhead than IntServ.**

I. Introduction

Next generation high-speed IP networks are expected to support diverse multimedia applications in a scalable manner. In such networks, it is important to provide quality-of-service (QoS) guarantees to a wide range of applications in a scalable manner. Significant efforts have been made recently, and a few network architectures, such as the Integrated Services (IntServ) architecture [1] and Differentiated Services (DiffServ) architecture [2], have emerged as promising QoS-enabling architectures. IntServ provides end-to-end QoS guarantees to individual flows by having routers maintain the states of each passing flow and by reserving resources for each flow at routers along the path. As a consequence, IntServ does not scale well as the network size grows. On the contrary, DiffServ aggregates individual flows into different service classes at the edge of the network, and core routers within the network forward each packet to its next hop according to the per-hop behavior [13] associated with the service class of the packet. In DiffServ, core routers do not need to keep states for individual flows, and this greatly improves the scalability. However, due to flow aggregation and lack of admission control, DiffServ does not provide end-to-end QoS guarantees to individual flows.

This work was supported by the National Science Foundation through grants ANI-0083074 and ANI-9903427, by DARPA through Grant MDA972-99-1-0007, by AFOSR through Grant MURI F49620-00-1-0330, and by grants from the University of California MICRO Program, Hitachi, Hitachi America, Novell,Nippon Telegraph and Telephone Corporation (NTT), NTT Docomo, Fujitsu, and NS-Solutions.

In this paper, we propose and study a new QoS framework, called On-Demand QoS Path (ODP), which provides end-to-end QoS guarantees to individual flows with much less overhead than IntServ, while keeping the scalability characteristic of DiffServ. In order to provide QoS to individual flows with minimal overhead, ODP exercises per flow admission control and end-to-end bandwidth reservation at the edge of the network. In the core of the network, it only differentiates the service types as in DiffServ.

In the core of the network, ODP provisions bandwidth statically for each traffic class (provisioned link) at each physical link as DiffServ does, and applies class-based Weighted Fair Queue [3][4] to differentiate them. In ODP, each provisioned link assigns some amount of its bandwidth to each edge router. The bandwidth assigned to a specific edge router can only be used by flows originating from that edge router. By recording utilization status of the assigned bandwidth on each provisioned link locally at each edge router and by deploying source routing, an edge router can make admission decisions instantly without hop-by-hop signaling, unlike IntServ, when a user flow with QoS requirement arrives. For each admitted flow, an edge router first chooses an appropriate path for the flow and then reserves bandwidth required on each provisioned link along the path by updating the local bandwidth utilization table. Being adaptive to traffic load, the assigned bandwidth to each edge router can be reconfigured on demand through bandwidth releasing or requesting. The signaling procedure incurred by bandwidth reconfiguration is targeted for aggregated flows so that the signaling overhead of ODP is reduced significantly compared to IntServ.

In ODP, we propose three approaches to fulfill the reconfiguration task: (1) Central Control, (2) Router-Aided, and (3) Edge-to-Edge. They stem from which entity monitors the bandwidth utilization status at provisioned link level and makes reconfiguration decision. In Central Control, a central network entity maintains information of bandwidth utilization status at provisioned link level and makes reconfiguration decision accordingly; in Router-Aided Control, it is the core routers which are responsible for maintaining the utilization status and making reconfiguration decision; in Edge-to-Edge Control, it is the edge routers which are responsible for these tasks. These three approaches explore centralized, semi-centralized and distributed architecture, respectively. In this paper, through simulations, we investigate and compare three approaches. Additionally, we compare ODP with IntServ and DiffServ. Service quality metrics (such as the average packet delay and delay variance) and signaling overhead incurred by bandwidth management are measured to show that ODP provides QoS at the same level as IntServ with much less signaling overhead.

This paper is organized as following: Section II surveys related work and shows that the proposed framework is new and original. Section III describes the proposed framework, along with the three proposed bandwidth reconfiguration approaches. In Section IV, simulation models are described, and in Section V, simulation results are presented to investigate the performance of the proposed framework. Finally, concluding remarks are given in Section VI.

II. Related Work

In this section, we survey the existing QoS frameworks: IntServ, DiffServ, IntServ over DiffServ, DiffServ extensions with admission control, and DiffServ-aware Traffic Engineering (DS_TE).

In IntServ architecture, RSVP (Resource Reservation Protocol) is used to reserve resources at routers along the path of a traffic flow. When a user flow arrives with a bandwidth requirement, the ingress edge router initiates to set up the path in the network through PATH messages to the destined egress edge router; the destined egress edge router tries to reserve the required resource on the path through RESV messages reversely to the source ingress edge router. Core routers on the path configure their traffic control mechanisms in such a way that each admitted flow is guaranteed to receive the service requested. Since each flow reserves network resource for itself, coexisting flows will not interfere with each other. This per-flow based hop-by-hop signaling method enables IntServ to provide end-to-end QoS guarantees. However, per-flow reservation processing and state maintaining at each router bring up significant scalability problems as the network grows.

In DiffServ architecture, scalability is achieved by offering services on a per-class basis instead of a per-flow basis. In DiffServ, packets are classified into a small number of classes at the boundary of a network, and the routers within the network merely implement a suite of scheduling/buffering mechanisms based on the classes. This per-hop per-class handling method simplifies a router's functionalities and reduces the states a router has to keep. It also removes the QoS signaling overhead involved with each flow. Thus it is much more scalable than IntServ. However, with DiffServ, no end-to-end QoS guarantees are provided to individual flows. In addition, DiffServ architecture lacks a standardized admission control scheme. When overload condition occurs in a given service class, all flows in that class suffer potentially harsh service degradation regardless of their QoS requirements.

Both IntServ and DiffServ attempt to provide QoS on top of the current Internet, but with different approaches; IntServ promotes end-to-end QoS guarantees by setting up an end-to-end connection for each flow and maintaining the states of all connections, whereas DiffServ promotes scalability by pushing flow classification to network boundary and offering QoS to service classes. Targeting at both end-to-end QoS guarantees and scalability, RFC 2998[7] suggests a combination of the two architectures, an IntServ over DiffServ framework. In this framework, IntServ is applied at the edge of a network where scalability is not an issue, and DiffServ is applied in the core of the network where a large number of flows coexist. DiffServ core routers may or may not be RSVP-aware. Edge routers are responsible for admission control and service mapping from IntServ to DiffServ.

In IntServ over DiffServ framework [15][17][18], resources can be provisioned either statically or dynamically using RSVP. With static provision, the core network provides a predefined set of service levels to customers (e.g. ingress edge routers), and network resources are statically provisioned according to those service level specifications (SLSs). Each edge router contains a local table that describes the bandwidth provisioned for each service class through it. Based on this table and service mapping, an edge router can make admission decisions for IntServ flows. Compared with pure DiffServ architecture, this method protects QoS for admitted user flows by enforcing admission control at the network boundary. However, it does not readily support new SLS(s), and network resource utilization is usually low because over-provisioning is assumed to provide end-to-end QoS. With dynamic provision, RSVP signaling procedure is initiated when resources provisioned for certain path are not enough for new user flows. It adopts admission control and end-to-end bandwidth reservation from RSVP, and tries to keep scalability as DiffServ. If resources provisioned on a path are used by several flows, this method can simultaneously improve resource utilization and service quality. However, states describing all passing connections have to be kept at each core router. If resources provisioned on a path are used by single flow, the framework degrades to IntServ. Compared to this framework, admission control in ODP is as simple and fast as the static provision, and resource management in ODP is as effective as the dynamic provision using aggregated RSVP with even less states kept at routers.

In [8] [9] [10][19], an admission control function is provided over DiffServ network by means of the so-called Endpoint Admission Control (EAC). EAC builds upon the idea that admission control can be managed by pure end-to-end operation, involving only the source and destination hosts. At connection set-up, each sender-receiver pair starts a Probing phase whose goal is to determine whether the considered connection can be admitted to the network. The source node sends packets that reproduce the characteristics (or a subset of them) of the traffic that the source wants to emit through the network. Upon reception of the first probing packet, the destination host starts monitoring probing packets' statistics (e.g., loss ratio, inter-arrival times) for a given period of time. At the end of the measurement period and on the basis of suitable criteria, the receiver makes the decision whether to admit or reject the connection and notifies this decision to the source node. Although the described scheme looks elegant and promising (e.g., it is scalable since it does not involve inner routers), a number of issues come out when we look for QoS performance. A scheme purely based on end point measurements suffers performance drawbacks mostly related to measurement period at the destination. Measurements taken in a short time cannot capture stationary network states, and thus, the decision is made based on a snapshot of the network status, which can be quite an unrealistic picture of the network congestion level. On the other hand, if measurements are performed on a longer time scale, then the admission control process will be too slow.

In [11] [12], a new QoS framework, named DiffServ-aware MPLS Traffic Engineering (DS_TE), is proposed. One possible solution to DS_TE is to further aggregate service classes to class types (CT) and provision resources for each CT inside the network. In order to reduce flooding link state advertisements, Inter Gateway Routing Protocol (IGP) extensions of per-class-type (CT) link-state advertisements (LSA) is used to communicate available bandwidth for each class type (CT). When edge routers are making admission control, they choose a path for the incoming flow with Constraint Based Routing. Although using CT improves the scalability of IGP LSAs by propagating information on a per-CT basis instead of on a per-class basis, no bandwidth requirement and provisioning are enforced for each service class within a CT. Flows of different service classes within a CT may interfere with each other. There is still concern about scalability of the IGP overhead, and particularly the IGP response to overloads and failures.

Compared with above frameworks, ODP solves scalability problem of IntServ, service quality problem of DiffServ and DS_TE, and inaccuracy problem of DiffServ/EAC. It guarantees the QoS for admitted flows, while keeping the scalability characteristic of DiffServ at the same time.

III. Scheme Description

To provide end-to-end QoS guarantees for each flow, ODP performs admission control and end-to-end bandwidth reservation for each flow at the network boundary. To achieve scalability, class-based service differentiation is provided in the network core. In Section 3.1, the basic architecture of ODP is first described; in Sections 3.2 and 3.3, two major functionalities of ODP, namely, admission control and bandwidth reconfiguration are described respectively.

3.1 ODP Architecture

In ODP, it is assumed that two types of routers, edge routers and core routers exist in a network. Edge routers make admission decisions, map individual flows to different service classes and transmit packets to the network for the network objects (i.e., clients). Core routers are DiffServ routers that provide class-based service differentiation.

In ODP, to provide class-based service differentiation and local admission control at network edge without hop-by-hop signaling, link bandwidth is organized in hierarchy. Each physical link is statically divided into multiple Provisioned Links (PLs); one PL is dedicated to one traffic class. Each PL is further divided into multiple trunks; one trunk is dedicated to one edge router. A trunk (of a given provisioned link) supports flows (of the given traffic class) originating from the same source edge router irrespective to their destination. Flows going through a same PL in the network but originating from different source edge routers use different trunks. An (source) edge router keeps track of available bandwidth of its assigned trunks and performs admission control locally without hop-by-hop signaling (admission control procedure is described in detail in Section 3.2.). Figure 1 illustrates this hierarchical bandwidth organization.

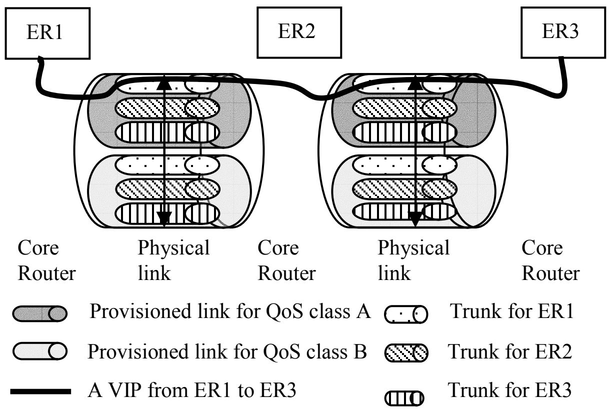

Figure 1. Hierarchical bandwidth organization

In ODP, it is assumed that the bandwidth is statically assigned to provisioned links based on a long-term traffic prediction made by a central network management server (NMS) or by the network administrator during network initialization. These assignments remain unchanged for a reasonably long period of time compared to our interests of time scale. In ODP, short-term dynamic traffic changes are managed by adjusting bandwidth assignment at trunk level dynamically; a portion of the bandwidth of an underutilized trunk (or trunks) is reassigned to a trunk (or trunks) whose bandwidth utilization is high.

Based on the network entity used to keep track of bandwidth utilization status, we propose three approaches to dynamically reconfigure trunk bandwidth: Central Control, Router-Aided Control, and Edge-to-Edge Control. Detailed description of each scheme is described in Section 3.3.

3.2 Admission Control Procedure

As described earlier, in ODP framework, edge routers perform admission control without hop-by-hop signaling. To achieve this, an edge router keeps two tables: a virtual IP path (VIP) table and a trunk table. A VIP is a path from a source edge router to a destination edge router for a specific traffic class. It is a concatenation of multiple trunks over a source-destination path belonging to the source edge router. A VIP table records all VIPs originating from a given edge router to all the other edge routers. A VIP table can be pre-configured or constructed by routing mechanisms [6], which is outside the scope of this paper. A VIP table consists of ID of destination edge router, traffic class, and list of IDs of trunks constituting the VIP path. (Refer to Table 1.) A trunk table records all the trunks belonging to a given edge router and their bandwidth utilization information. A trunk table consists of trunk ID, ID of the provisioned link to which the trunk belongs, the amount of reserved bandwidth on the trunk, and the amount of used bandwidth on the trunk. (Refer to Table 2.)

Destination Edge Router	Traffic Class	List of Trunks

Table 1. VIP table

Trunk ID	Provisioned Link ID	Reserved Bandwidth	Used Bandwidth

Table 2. Trunk table

When a new flow arrives at a source edge router, the edge router first identifies VIP to its destination edge router by examining the VIP table. (It is assumed that traffic classification is already done before the edge router examines the VIP table, as it is one of the basic functions of a DiffServ edge router.) The source edge router, then, checks bandwidth availability of each trunk on the VIP by examining its trunk table. If there is a VIP that has enough bandwidth to support the incoming call, the source edge router accepts the incoming call and makes end-to-end bandwidth reservation for this flow by updating the Used Bandwidth field of each trunk on the VIP in its local trunk table. If no VIP has sufficient bandwidth to support the incoming call, the new flow is rejected.

When a user flow finishes, the source edge router returns the bandwidth used by that user flow back to the trunks constituting the corresponding VIP by updating local trunk table.

3.3 Trunk Reconfiguration Procedure

In order to enhance the utilization of the network in the presence of short-term, dynamic traffic changes, ODP dynamically adjusts the amount of bandwidth assigned on the trunks. A source edge router releases or requests trunk bandwidth based on the trunk bandwidth utilization status. The basic procedure of trunk reconfiguration process is as follows:

1. A provisioned link table maintains bandwidth utilization of each provisioned link. It can be maintained centrally, semi-centrally or in a distributed way.
2. A source edge router periodically checks the bandwidth utilization status of its trunks by examining its trunk table shown in Table 2. If the utilization of the bandwidth on a given trunk is under the predetermined lower threshold, the edge router sends a control message (or messages) to the network entity (or entities), which maintain(s) the provisioned link table, in order to release some amount of the bandwidth.
3. When a new flow arrives, a source edge router checks the trunks on VIP for the flow. If the used bandwidth of some trunks reaches the predetermined upper threshold, the source edge router sends a control message (or messages) to the network entity (or entities), which maintain(s) the provisioned link table to request more bandwidth for congested trunks. This is the threshold-driven trigger of trunk reconfiguration procedure. On receiving a confirmation message, the initiating edge router increases the bandwidth of those trunks according to the confirmation message.

As described above, the trunk reconfiguration mechanism in ODP is always initiated by a source edge router. A threshold driven mechanism is used to release and request bandwidth. The checking period, upper and lower thresholds, the amount of bandwidth to release and request are predetermined by a network administrator when the network is initialized.

The provisioned link table can be maintained centrally, semi-centrally or in a distributed way. Based on the network entity that maintains the provisioned link table, the following three approaches are proposed: Central Control approach, Router-Aided control approach and Edge-to-Edge control approach. In Central Control approach, the provisioned link table is centrally maintained by a Network Management Server (NMS). In Router-Aided approach, each core router keeps a record of the bandwidth utilization of the provisioned links on all physical links directly connected to it. In Edge-to-Edge approach, each edge router keeps a copy of the provisioned link table. Table 3 shows the format of provisioned link table. A provisioned link table consists of provisioned link ID, ID of the physical link to which the provisioned link belongs, the amount of reserved bandwidth on the provisioned link, the amount of used bandwidth on the provisioned link, and the traffic class of the provisioned link.

Provisioned Link ID	Physical Link ID	Reserved Bandwidth	Used Bandwidth	Traffic Class

Table 3. Provisioned link table

	Central Control	Non Central Control	
		Router-Aided	Edge-to-Edge
Provisioned Link Table	NMS maintains the bandwidth status (reserved bandwidth, used bandwidth) for all of the provisioned links in the network	Each core router maintains the bandwidth status for the provisioned links directly connected to it.	Each edge router maintains a copy of the bandwidth status for all of the provisioned links in the network
Trunk Table	Each edge router maintains the bandwidth status for all of its own trunks.		

Table 4. Summary of provisioned link table and trunk table

Table 4 summarizes how the provisioned link table and the trunk table are maintained in each of the three approaches. The detailed procedure of each of the three approaches is discussed in later subsections (Section 3.3.1 through 3.3.3).

In the trunk reconfiguration process, three types of messages are exchanged between the network entities: Trunk Reconfiguration Request message (TRRequest), Trunk Reconfiguration Release message (TRRelease), Trunk Reconfiguration Confirmation message (TRConfirm). All three messages have the same format, and it is shown in Figure 2. Each message has message type field that indicates the type of the message (i.e., TRRequest, TRRelease or TRConfirm), ID of source edge router, ID of destination edge router, ID of VIP route used by this message, the number of trunks carried in this message, and a series of (trunk ID, amount of bandwidth request/release) pair. Once a source edge router decides to reconfigure bandwidth for its bottleneck trunks, it sends out a TRRequest indicating which trunk to increase what amount of bandwidth to the network entity (or entities) which maintain(s) the provisioned link table; TRConfirm is the decision from the network entity (or entities) for the trunk reconfiguration request. TRRelease is used by edge routers to indicate which trunk to release what amount of bandwidth.

Msg ype	Src ID	Dest ID	Route ID	# of Trunks	Trunk 1	Bdw for Trunk 1	…	Trunk n	Bdw for Trunk n

Figure 2. TRRequest / TRRelease/ TRConfirm

On receiving a TRRelease message, the network entity with the provisioned link table, says X, carries out the following trunk bandwidth release process:

```
For trunk i specified in the TRRelease message (Figure 2),
1. check if provisioned link $PL_i$ of trunk $T_i$ is in X's provisioned link table.
2. If yes, update the provisioned link table by $PL_i$.UsdBdw = $PL_i$.UsdBdw - bdw for $T_i$.
```

On receiving a TRRequest message, the network entity with the provisioned link table carries out the following trunk bandwidth request process:

```
1. Generate a TRConfirm message. (Figure 2)
2. Copy each (trunk i, Bdw_Req) pair specified in the TRRequest (Figure 2) to TRConfirm.
3. For trunk i specified in the TRRequest,
   3.1 Check if provisioned link $PL_i$ of trunk $T_i$ is in X's provisioned link table;
     if yes, then do 3.2
     else set trunk i bdw_req field in TRConfirm message to 0
     ( i.e. can not make decision for this request)
   3.2 Check if provisioned link i has enough available bandwidth;
       if PL i.RsvBdw-PL i.UsdBdw $\geq$ trunk i bdw_req
       then update the provisioned link table by
       PL i.UsdBdw = PL i.UsdBdw + trunk i bdw_req;
       else set trunk i bdw_req field in TRConfirm message to 0
       ( i.e. denial to this request for the trunk)
4. Send the TRConfirm message to the edge router which initiates the TRRequest message.
```

Note that in ODP, bandwidth is dynamically reassigned not for all the trunks of a VIP, but only for trunks that do not have sufficient bandwidth. Detailed explanation on the above two procedures for different approaches in ODP are described in the following three subsections.

3.3.1 Central Control Approach

In Central Control (CC) approach, an edge router updates local trunk table and sends out a TRRelease to NMS when some bandwidth can be released to the provisioned link. When NMS receives TRRelease, trunk bandwidth release process is executed. According to the threshold-driven trunk reconfiguration trigger mechanism, an edge router requests a set of bandwidth for some trunks. The edge router sends out a TRRequest to NMS. When NMS receives TRRequest, it executes trunk bandwidth request process. On receiving of TRConfirm from NMS, the initiating edge router updates its local trunk table according to the message.

3.3.2 Router-Aided Approach

In router-aided (RA) approach, edge router updates its local trunk table and broadcasts TRRelease to all core routers. Core routers receiving TRRelease will execute trunk bandwidth release process. According to the threshold-driven trunk reconfiguration trigger mechanism, an edge router requests more bandwidth for bottleneck trunks on a VIP by sending TRRequest(s) to core router(s) along the VIP. On receiving a TRRequest, each of them executes trunk bandwidth request process. The initiating edge router receives TRConfirm(s) from core routers on the path and updates its local trunk table according to the messages.

3.3.3 Edge-to-Edge Approach

In edge-to-edge (E2E) approach, an edge router updates its local trunk table and broadcasts TRRelease to all the other edge routers. Edge routers receiving TRRelease will execute trunk bandwidth release process. According to the threshold-driven trunk reconfiguration trigger mechanism, an edge router initiates a trunk reconfiguration procedure for its bottleneck trunks. It is possible for several edge routers to compete for the bandwidth on the same provisioned links. Therefore, each copy of provisioned link table has to be synchronized, and the initiating edge router needs to get permission from all other edge routers by broadcasting TRRequest to them. On receiving TRRequest, an edge router executes trunk bandwidth request process and sends back its decision. Due to the time variation between edge routers, decisions from different edge routers may be different. Only after receiving TRConfirm(s) from all the other edge routers, the originating edge router can find out whether this request is admitted or not.

To synchronize all provisioned tables kept at other edge routers, the originating edge router needs to send either TRComplete or TRCancel to inform other edge routers of the final decision of the trunk reconfiguration procedure. TRComplete and TRCancel are two additional types of control messages used in edge-to-edge approach, and they have the same format shown in Figure 2. If TRRequest is accepted by all the other edge routers, the originating edge router updates its trunk table and provisioned link table according to TRConfirm(s) and broadcasts TRComplete to all other edge routers; otherwise, the request is denied and the originating edge router will cancel this request by broadcasting TRCancel to all other edge routers. On receiving a TRCancel, all other edge routers roll back their modifications to those provisioned links.

IV. Simulation Model

To verify our expectations, IntServ/RSVP, DiffServ and ODP, are simulated using OPNET. In our simulation, packets of three traffic classes, video, audio and UDP, are generated. Table 5 shows traffic characteristics of these three types of traffic. Video parameter values are obtained from MPEG-4;

audio parameter values are obtained from VoIP [14][20] ITU-T G.723.1 standard with the assumption that an audio packet consists of 4-byte compressed IP/RTP header and 26-byte payload; UDP parameter values are randomly chosen to add some noise. It is assumed that the ratio of audio sessions to video sessions generated in a LAN is 99:1.

Traffic Class	Aver Bdw Req	Session Duration	Packet Size (bits)	Packet Inter-arrival time (s)	Major Source
Video	3 Mbps	EXP (900s)	Constant (8000)	Peak: EXP (0.0016)	VIDEO MPEG-4
				Non-peak: EXP(0.0033)	
				Peak: Non-peak =1:15	
Audio	8 Kbps	EXP (300s)	Constant (240)	Constant(0.03)	VoIP ITUT G.723.1
UDP			Constant (8192)	EXP (0.03)	

Table 5. Traffic parameter values

To simulate a realistic network environment, the backbone of VBNS+ [5] is used as the network topology in our simulation, and this topology is shown in Figure 3. As in VBNS+ network, OC-3 fiber links are used as physical links between core routers, and the distance between two core routers is also obtained from VBNS+. Altogether, there are 15 core routers, and 3 edge routers are connected to each core router to transmit packets to and from LANs. Therefore, there are 45 edge routers and 45 LANs in our simulation model. It is assumed that source-destination pair is evenly distributed in this topology. In ODP Central Control approach, a network management server is connected to core router 6 with an OC-3 fiber link.

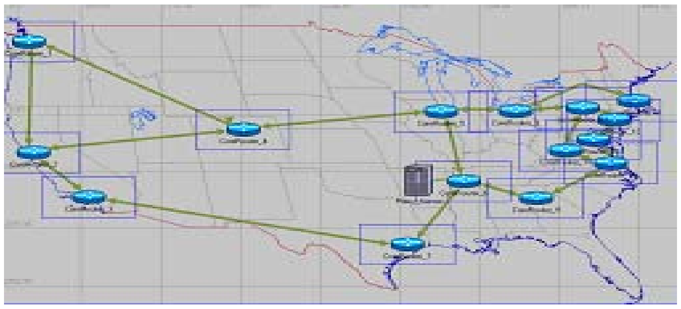

Figure 3. Topology from VBNS+

In our simulation, different levels of traffic load are considered, ranging from low to high. Under low traffic load, it is assumed that the inter-arrival times of two successive user flows follow an exponential (50 seconds) distribution, which corresponds to traffic load of Erlang(275.4); under medium traffic load, the average inter-arrival time is reduced to 25 seconds, which corresponds to traffic load of Erlang(518.4); under high traffic load, the average inter-arrival time is further reduced to 10 seconds, which corresponds to traffic load of Erlang(1377). Each simulation is run for 5000 seconds.

Using the simulation model described above, the performance of ODP is evaluated and compared with that of IntServ and DiffServ. The following five performance metrics are used in comparison:

- Blocking rate: This is the ratio of the number of blocked user flows to the number of all user flows.
- Signaling overhead: This is measured in the total number of control messages transmitted during the simulation period. In IntServ, only RSVP signaling messages, such as PATH, RESV and PATH-TEAR messages, are counted. In DiffServ, no control messages are transmitted. In ODP, TRRequest, TRConfirm and TRRelease are counted as control messages. In ODP edge-to-edge approach, TRComplete and TRCancel are also counted as control messages.
- Average connection setup time: This is the average time interval between the arrival time of a user flow at an edge router and its admission decision time.
- Average packet delay: This is the average time interval between the arrival time of a packet at an ingress edge router and the delivery time of the same packet to the egress edge router. In our simulation, average packet delays of audio, video and UDP sessions between an edge router connected to CoreRouter_1 and an edge router connected to CoreRouter_13 are computed since this path traverses the longest path in the simulation network.
- Average packet delay jitter: This is the variance in packet delay. It is also measured between an edge router connected to CoreRouter_1 and an edger router connected to CoreRouter_13.

V. Numerical Results

In ODP framework, there are some tunable parameters that affect the performance of ODP. Therefore, the optimal values of these tunable parameters are first determined in Section 5.1, and these optimal values are used for ODP in comparison with IntServ/RSVP and DiffServ in Section 5.2.

5.1. ODP Tunable Parameters

In ODP framework, there are 5 tunable parameters:

- Bandwidth utilization checking period (*m*): It is expressed in seconds, and it shows how often a source edge router checks the bandwidth utilization status of its trunks.
- Lower threshold (*l*): If the bandwidth utilization on a given trunk is lower than this threshold, a source edge router releases some bandwidth. To avoid the situation where bandwidth request needs to be made immediately after the bandwidth release, the bandwidth reconfiguration procedure always tries to maintain enough bandwidth to accommodate one future flow. Therefore, the ratio of $(B_{used} + B_{average})$ to $B_{reserved}$ is used as the bandwidth utilization measure here, and bandwidth release process is triggered if $\frac{B_{used} + B_{average}}{B_{reserved}} < l$. B_{used} represents amount of used bandwidth on a trunk; $B_{average}$ represents average amount of bandwidth requirement of a flow, which depends on the

flow's traffic type; $B_{reserved}$ represents amount of reserved bandwidth on a trunk.

- Bandwidth releasing granularity (r_1): When bandwidth release process is triggered, some percentage of spare bandwidth is released, and r_1 shows this percentage. Again, to maintain enough bandwidth to accommodate one future flow, ($B_{reserved} - B_{used} - B_{average}$) is used as the amount of spare bandwidth.
- Upper threshold (h): If the bandwidth utilization on a given trunk is higher than this threshold, a source edge router requests more bandwidth. To maintain enough bandwidth to accommodate one future flow, the ratio of ($B_{used} + B_{average}$) to $B_{reserved}$ is again used as the bandwidth utilization measure. Therefore, bandwidth request process is triggered if $\frac{B_{used} + B_{average}}{B_{reserved}} > h$.
- Bandwidth requesting granularity (r_2): When bandwidth request process is triggered, the amount of requested bandwidth can be multiples of the average amount of bandwidth requirement of a flow, and r_2 shows the multiplying factor. That is, the amount of requested bandwidth is $r_2 \; x \; B_{average}$.

In this section, the optimal values of these 5 parameters are determined through simulation. In each set of simulation experiments, only one of these parameters is varied while others are fixed. Once the optimal value is determined for a parameter, that value is used for the parameter in the subsequent discussions. In determining optimal values for tunable parameters, the blocking rate is used as the most important criteria since the major objective of ODP is to provide QoS guarantees to as many user flows as possible. In case of a tie, the signaling overhead is used as a secondary metric.

In all the figures presented in this section, the x-axis shows different values for the parameter in investigation at three different traffic loads (i.e., low, medium and high traffic loads); the left y-axis represents the blocking rate; the right y-axis represents the signaling overhead.

Figures 4 through 6 show the performance of ODP-CC (Central Control), ODP-RA (Router-Aided control) and ODP-E2E (Edge-to-Edge control), respectively, for various values of bandwidth utilization checking period m. In these figures, only m is varied, while others are set as follows: l = 1, r_1 = 100%, h = 1 and r_2 = 1. These values are chosen as default values based on the preliminary flow-level simulations, which are omitted due to the page limit. In general, as a source edge router checks the bandwidth utilization status of its trunks more frequently, it can reconfigure its trunk bandwidth more dynamically, thereby decreasing the blocking rate. However, since traffic pattern doesn't change during the simulation period, after the checking period is reduced to some threshold value, the blocking rate does not decrease further. Instead, the blocking rate starts to increase as spare bandwidth is released too early to accommodate near-future flows. Therefore, the blocking rate of ODP in all three approaches decreases at first as the bandwidth utilization checking period decreases, but after the checking period reaches some thresholds, the blocking rate starts to increase. This trend is more apparent at higher traffic load. The signaling overhead, on the other hand, steadily increases as the checking period decreases. This is because as the more frequently bandwidth utilization is checked, the more control messages, such as TRRelease and TRRequest, are generated.

From Figures 4 through 6, it can be determined that the checking period of 10 seconds gives the best result for ODP-CC and ODP-RA, whereas the checking period of 25 seconds gives the best result for ODP-E2E. In ODP-E2E, when several edge routers compete for the bandwidth on the same provisioned link, it is possible that none of these requests are satisfied due to the asynchronous arrivals of TRRequests at other edge routers. Therefore, in ODP-E2E, if the checking period is too small, there is higher probability that multiple TRRequests compete for the same provisioned link, thereby increasing the blocking rate. As a result, in ODP-E2E, the lowest blocking rate is achieved at larger value of m.

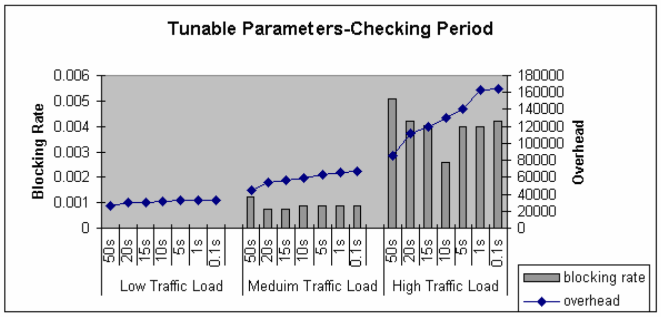

Figure 4. ODP-CC with varying checking period

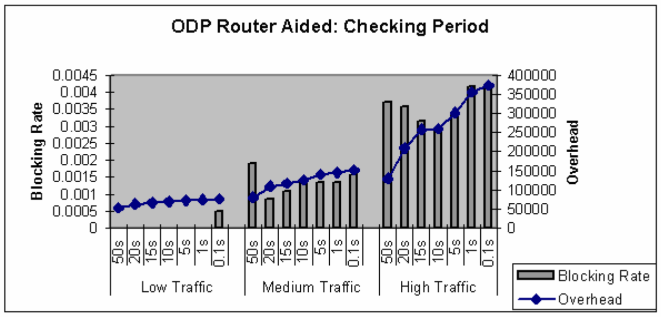

Figure 5. ODP-RA with varying checking period

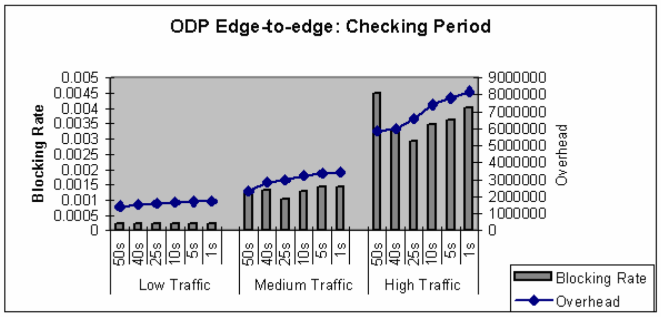

Figure 6. ODP-E2E with varying checking period

Because of page limitation, here we only give out one tunable parameter discussion, for more tunable parameter discussions, please refer to the extended version at http://www.ics.uci.edu/ ~meiy/ODP-CCW03extended.pdf

5.2. Performance Comparison

In this section, the performance of ODP is compared with that of IntServ and DiffServ, using the five performance metrics discussed in Section IV. Since neither admission control nor signaling messages are used in DiffServ architecture, for blocking rate, signaling overhead and average connection setup time, only IntServ architecture is compared with three approaches of ODP framework. For average packet delay and delay jitter, three approaches of ODP framework are compared with IntServ and DiffServ.

Figure 7 shows the blocking rate and signaling overhead of IntServ and three approaches of ODP. Among the tree approaches of ODP, ODP-E2E has the highest blocking rate and signaling overhead. In ODP-E2E, when several edge routers compete for the bandwidth on the same provisioned link, it is possible that none of these requests are satisfied due to the asynchronous arrivals of TRRequests at other edge routers. Therefore, ODP-E2E has the highest blocking rate. In addition, ODP-E2E requires extra signaling messages, such as TRComplete/TRCancel, to synchronize PL tables kept at each edge router, and it broadcasts TRRequest messages to all edge routers. Therefore, it also has the highest signaling overhead among the three approaches of ODP. Between IntServ and ODP, IntServ has lower blocking rate than all three approaches of ODP. In ODP, due to the static bandwidth provisioning at PL level, it is possible that a new flow of a given traffic type is rejected even if there is enough bandwidth reserved for other traffic types. However, the signaling overhead of IntServ is significantly higher than that of ODP-CC and ODP-RA as expected. This is because trunk-level bandwidth management in ODP requires significantly less signaling messages than per-flow bandwidth management in IntServ.

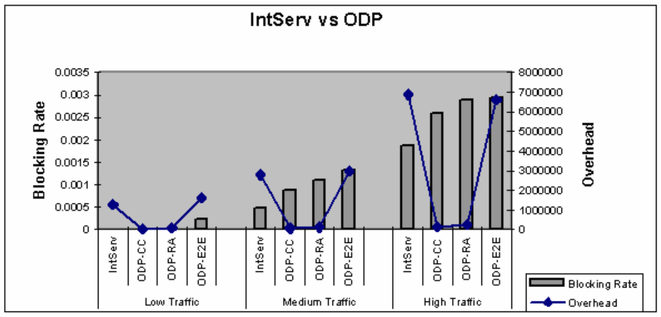

Figure 7. Blocking rate and signaling overhead comparison

Figure 8 shows the average connection setup time of IntServ and three approaches of ODP. As expected, all three approaches of ODP give significantly lower connection setup time than IntServ. This is because in ODP framework, edge routers perform local admission control without hop-by-hop signaling.

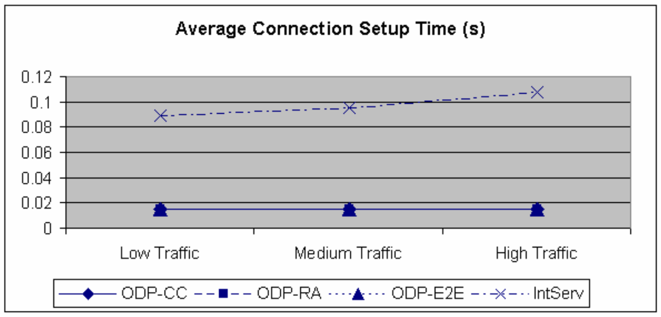

Figure 8. Average connection setup time comparison

Figure 9 shows the average packet delay and delay jitter of audio traffic for IntServ, DiffServ and ODP. It is seen in this figure that ODP-CC and ODP-RA provide the same level of QoS guarantees as IntServ and that DiffServ has greater packet delay and delay jitter than these three. For admitted user flows with QoS requirement, IntServ and ODP provide end-to-end QoS guarantees by reserving bandwidth on the paths that each flow will follow. DiffServ, on the other hand, does not use admission control, and it does not perform per-flow end-to-end resource reservation. Therefore, greater packet delay and delay jitter are experienced by end users in DiffServ. The average packet delay and delay jitter of ODP-E2E is almost same as that of DiffServ since in ODP-E2E, more bandwidth is consumed to transmit added signaling messages than the other two approaches of ODP. Note that the average delay and delay jitter of video traffic and UDP traffic were also obtained, and they showed similar behavior seen in this figure. These results are omitted due to the page limit.

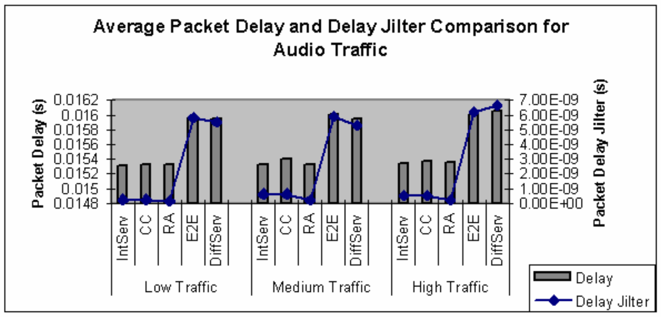

Figure 9. Average packet delay and delay jitter comparison

Simulation results presented in this section clearly show that the two approaches of ODP, namely, ODP-CC and OCP-RA, can provide end-to-end QoS guarantees to individual flows with much less signaling overhead than IntServ, thereby keeping the scalability characteristic of DiffServ.

VI. Conclusion

In this paper, a new QoS framework, called On-demand QoS Path (ODP), was proposed. In order to provide QoS to individual flows with minimal overhead, ODP exercises per-flow admission control and end-to-end bandwidth reservation at the edge of the network and only differentiates service types in the core of the network. In addition, to be adaptive to traffic load, ODP monitors the bandwidth utilization status of the network and performs dynamic bandwidth reconfiguration in the core based on the network status. Through extensive simulations, the performance of proposed QoS framework was investigated and compared with that of IntServ and Diff-Serv architectures. The simulation results clearly showed that ODP Central Control and Router-Aided approaches could provide end-to-end guarantees to individual flows with much less overhead than IntServ.

Even through the numerical results did not show apparent advantage of ODP End-to-End approach, it is the most robust solution since it avoids single-point-failure problem. For example, in Central Control, if NMS fails, trunk reconfiguration process will come to a complete halt. In Edge-to-Edge Control, on the other hand, even when an edge router fails, trunk reconfiguration process can still be carried on by excluding the failed edge router from the process.

References

1. J. Wroclawsky, “The use of RSVP with IETF Integrated Services", IETF RFC2210, September 1997.
2. K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and Ipv6 Headers", RFC2474, December 1998.
3. S. Blade, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services", IETF RFC2475, December 1998.
4. L. Zhang, VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks., ACM Transactions on Computer Systems, v9.2, 1991, p. 101-121.
5. http://www.vBNS.net
6. G. Apostolopoulos, S. Kamat, R. Guerin, SK. Tripathi, "Quality of Service Based Routing: A Performance Perspective", SIGCOMM, 1998.
7. Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer etc., “A Framework for Integrated Services Operation over Diffserv Networks”, RFC 2998, November 2000.
8. Viktória Elek, Gunnar Karlsson, Robert Rönngren, “Admission Control Based on End-to-End Measurements”, INFOCOM 2000
9. G. Bianchi, N. Blefari-Melazzi, “A Migration Path to provide End-to-End QoS over Stateless Networks by Means of a Probing-driven Admission Control”, Internet Draft, November 2001
10. Matthias Grossglauser , David N. C. Tse, A framework for robust measurement-based admission control, IEEE/ACM Transactions on Networking (TON), v.7 n.3, p.293-309, June 1999
11. Francois Le Faucheur, “Protocol extensions for support of Diff-Serv-aware MPLS Traffic Engineering”, IETF Internet Draft, October 2002
12. Francois Le Faucheur, Waisum Lai, “Requirements for support of Diff-Serv-aware MPLS Traffic Engineering”, IETF Internet Draft, February 2003
13. Brim, S.; B. Carpenter, F. Le Faucheur. “Per Hop Behavior Identification Codes”. IETF RFC 2836, May 2000
14. Lin, H.; T. Seth; A. Broscius; C. Huitema. “VoIP Signalling Performance Requirements and Expectations”. IETF Internet Draft, Jun 1999
15. Reichmeyer, F.; L. Ong; A. Terzis; L. Zhang; R. Yavatkar. “A Two-tier Resource management model for Differentiated Services Networks”. IETF Internet Draft, Nov 1998
16. Kelly, F. P.; P.B. Key; S. Zachary. “Distributed Admission Control”, IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, Dec: 2617-28, 2000
17. F. Borgonovo, A. Capone, L. Fratta, M. Marchese, C. Petrioli "End-to-end QoS provisioning mechanism for Differentiated Services", Internet Draft, July 1998
18. F. Borgonovo, A. Capone, L. Fratta, C. Petrioli, "VBR bandwidth guaranteed services over DiffServ Networks", IEEE RTAS Workshop, June 1999
19. G. Bianchi, A. Capone, C. Petrioli, "Packet management techniques for measurement based end-to-end admission control in IP networks", KICS/IEEE Journal of Communications and Networks (special issue on QoS in Internet), July 2000
20. Brady, P. T., "A Model for Generating On-Off Speach Patterns in Two-Way Conversation", The Bell System Technical Journal, September 1969, pp.2445-2471