Distributed QoS Measurement and Edge-to-Edge QoS Aggregation in Differentiated Services Networks

Jae-Young Kim and James Won-Ki Hong

Department of Computer Science and Engineering
Pohang University of Science and Technology
{jay,jwkhong}@postech.ac.kr

Abstract. Differentiated Services (DiffServ) framework, which is currently being standardized in the IETF DiffServ working group, is a solution that can provide different service qualities to different classes of Internet users. However, QoS management of DiffServ networks is not yet fully discussed and still has many areas to be solved. In this paper, we suggest a distributed QoS measurement and edge-to-edge QoS aggregation of DiffServ flows in a DiffServ domain. An edge-to-edge DiffServ flow is modeled by combining a routing path and aggregated QoS measurement results observed locally at transit routers located on the routing path. We show construction methods of the edge-to-edge DiffServ flows with SNMP MIBs and simple aggregation rules. We believe that managing the edge-to-edge DiffServ flows can serve as a useful building block for managing QoS of DiffServ networks.

1 Introduction

The explosive growth of the Internet has resulted in an exponential increase in the number of users and the amount of network traffic. However, the amount of network bandwidth is frequently insufficient to satisfy the exponential increase in bandwidth requirements from various network applications. The quality of network service has been degraded at such points where network bandwidth becomes scarce. Unexpected packet loss, delay, and jitter occur when many network packets compete for insufficient network bandwidths at bottleneck points. Since both people and applications are becoming increasingly dependent on network services, to guarantee a sufficient amount of service quality has become the natural requirement of every user.

In order to provide service differentiation in the Internet, the Internet Engineering Task Force (IETF) has recently introduced Differentiated Services (DiffServ) framework [1]. DiffServ is an alternative approach to Integrated Services (IntServ) [2] because IntServ relies on per-flow states and per-flow processing in every network node and thus is very difficult to deploy in large backbone networks. Instead, DiffServ controls the aggregation of traffic flows; that is, DiffServ flows, at each routing decision point without using a signaling protocol. IPv4 Type-of-Service (ToS) octet or IPv6 traffic class octet is used for distinguishing the DiffServ flows [3]. Since DiffServ is a simpler and more scalable solution for Internet backbone networks, DiffServ is widely accepted as a feasible solution for providing Internet QoS.

DiffServ applies administrative domain concepts. At the boundary of the DiffServ

domain, edge routers perform classification of traffic flows based on 5-tuple information, composed of source and destination IP addresses, a type of transport protocol, and source and destination application port numbers. And the edge routers perform marking the most-significant 6 bits of ToS fields of incoming packets. This 6-bit mark is called Differentiated Services Code Point (DSCP). Within one domain, core routers forward traffic according to the DSCP value of DiffServ flows. Since the edge routers have already marked the DSCP value of the incoming traffic, core routers need not handle complex information in such traffic. Core routers perform various differentiating actions, such as dropping, metering, shaping, and remarking according to different DSCP values. This different packet treatment performed at each router is called Per-Hop Behavior (PHB). Best-Effort (BE), Expedited Forwarding (EF) [4] and Assured Forwarding (AF) [5] are examples of proposed PHBs from the IETF.

However, current DiffServ specifications have limitations in providing complete QoS management framework. As described in Fig. 1, QoS management tasks are composed of repeated cycles of configuration, measurement, and analysis. Without any one component of the cycle, the QoS management cannot be provided completely. According to this point of QoS management view, current DiffServ RFCs and drafts are mainly for QoS provisioning and configuration only. QoS measurement and analysis based on the measurement results are not yet addressed in detail.

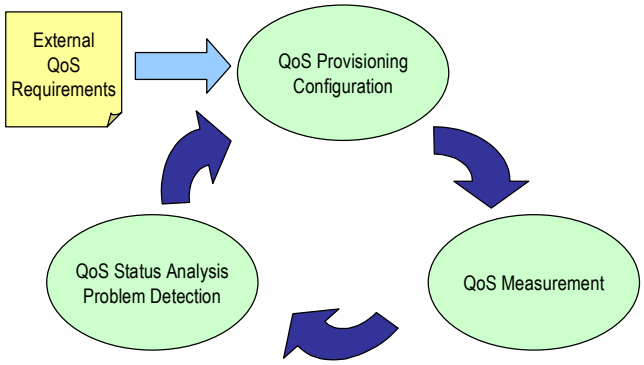

Fig. 1. QoS Management Cycle

In this paper, we suggest a distributed QoS measurement and edge-to-edge QoS aggregation method in DiffServ networks. Edge-to-edge QoS information is very useful when providing sophisticated QoS management operations. To build edge-to-edge QoS information, locally-observed QoS information is aggregated by following routing paths. Aggregated QoS information of DiffServ flows can provide network-wide traffic status in detail. Locating overloaded links and suggesting possible solutions can be drawn from the QoS status of DiffServ flows.

In our methods, each DiffServ router in a DiffServ domain measures QoS information of multiple QoS classes for every network interface. The distributed measurement has benefits in using less network resources and providing more efficient solution. When measuring network QoS information, we can have two kinds of measurement methods, active and passive measurements. Our measurement

method is passive measurement that does not alter or affect network traffic.

The rest of this paper is organized as follows. Section 2 investigates related work and Section 3 presents the modeling of DiffServ networks and edge-to-edge DiffServ flows. Section 4 describes the suggested distributed edge-to-edge QoS measurement and aggregation methods. Section 5 summarizes our work and discusses possible future work.

2 Related Work

Network QoS measurement has an important role when building complete network QoS management systems. We note several related research work concerning the monitoring QoS of IP networks.

IETF's real-time flow measurement (RTFM) architecture [6, 7] and Cisco's NetFlow [8] suggest and implement real-time measurement systems for IP flows. The systems retrieve information from IP packet header they monitor and distinguish different types of flow information. Various real-time flows are measured and statistical information can be reported periodically. IETF's RMON [9] and its extension RMON2 [10] are also able to monitor real-time traffic statistics on local area networks within the SNMP framework. However, these systems only collect information from a single network point. The systems are not aware of IP topology or routing protocols. If there is a need to construct a network-wide view of packet transmission, additional effort should be made to the current architecture. Besides, the systems do not provide edge-to-edge performance information.

Jiang et al. [11] proposed a distributed QoS monitoring mechanism in IP networks. An agent called a 'relevant monitor,' resides at several network points for monitoring real-time flows and reports collected information to monitoring applications. A central database, called a 'real-time application name server,' is used for monitoring applications to locate proper relevant monitors to retrieve specific flow information. The system proposes a distributed mechanism to construct end-to-end flow information; however, it does not understand routing topology and fails in showing how to construct network-wide flow information in detail.

Feldmann et al. [12] have developed the 'NetScope' toolkit that integrates accurate models of topology, traffic, and routing with a flexible visualization environment to support traffic engineering in large ISP networks. The approach is similar to our work in providing a network wide view of routing topology with performance statistics on network links. The purpose of the system is to locate heavily-loaded links so that traffic engineering tasks can distribute the load to other possible links. However, the purpose of our method is to show edge-to-edge QoS information for each DiffServ flow. Both methods are able determine which network link is overloaded, but; only our method is able to determine the loaded traffic comes from and where it sinks to because our system has QoS information of every edge-to-edge DiffServ flow. Further, the NetScope system has been applied to general IP networks while our method has been applied to QoS-enabled DiffServ networks. Our method is able to differentiate various traffic classes and monitor different QoS parameters.

Our approach is unique in providing edge-to-edge QoS information in QoS-enabled DiffServ networks. The proposed method combines topology and performance information to construct topology-aware QoS information that is very helpful in understanding and analyzing QoS provisioning status of the network and managing the DiffServ domain as a whole.

3 Modeling Edge-to-Edge DiffServ Flows

For modeling edge-to-edge DiffServ flows of our concern, we investigate and conceptualize DiffServ components building a DiffServ domain. First, conceptual model of a DiffServ router is presented and mathematical definitions of topology and flow are suggested.

A DiffServ domain consists of a set of DiffServ routers under a set of consistent provisioning rules. There are two kinds of DiffServ routers, one is edge router located at the boundary of the DiffServ domain and the other is core router interconnecting edge routers. Each DiffServ router is configured to handle DSCP-marked packets with pre-defined PHBs.

A DiffServ router is a fundamental DiffServ-enabled network node. The conceptual model and requirements of the DiffServ routers are discussed in [13, 14]. A DiffServ router is considered to have a routing module, a set of Traffic Control Blocks (TCBs), a queuing module, and a configuration and monitoring module that are organized as in Fig. 2.

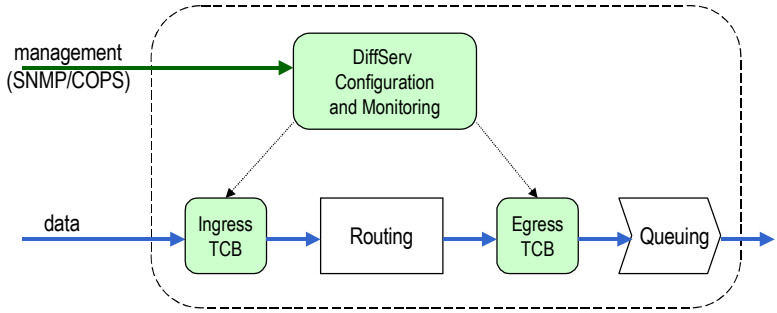

Fig. 2. Conceptual Model of a DiffServ Router

DiffServ-related modules are separated from the routing module to simplify the addition of the DiffServ capability to the existing router. Traffic conditioning can be performed either at the ingress point or at the egress point, or both. At each ingress or egress point, a set of TCBs is cascaded to form complicated traffic shaping. The queuing module is a set of underlying packet queues that store packets before a router sends them out. The management module for the DiffServ router can be operated in several ways, such as SNMP or Common Open Policy Service (COPS) protocol [15, 16]. The management module configures TCB parameters and monitors the performance of each TCB.

A conceptual TCB can be modeled as in Fig 3. When a packet comes in, the packet is classified by the classifier with the predefined classification rules. Classified packets are sent to one of multiple priority queues and affected by the packet drop algorithms within each queue when congestion occurs. The scheduler picks a packet from one of the priority queues by following scheduling algorithms. Traffic flows following the procedures are shaped and tailored to meet the requirements of the given traffic class.

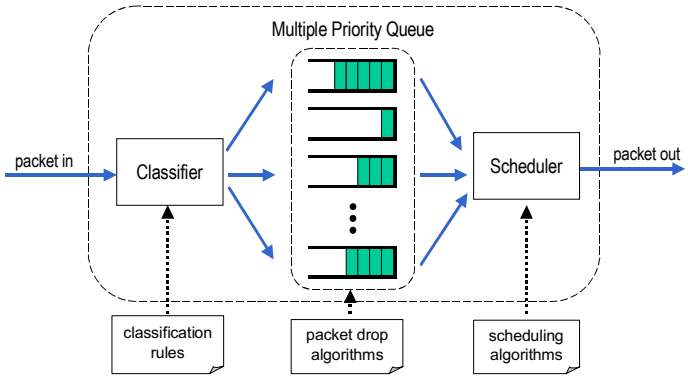

Fig 3. Conceptual Model of a TCB

The important thing to consider from this DiffServ router model is that a DiffServ router can handle QoS information of multiple DiffServ classes. If a router r_j handles packets of DiffServ class of c_i, we can denote the QoS of the packets of DiffServ class of c_i experienced in the router r_j, is $Q(c_i,r_j)$. QoS information can include various performance parameters, such as throughput, jitter, delay, number of packet drops, etc..

In order to construct an edge-to-edge model of DiffServ flows, it is necessary to have a topological model of DiffServ flows as well as a QoS information model because edge-to-edge DiffServ flow is built on a routing path from one edge router to the other edge router. We denote the topology of a DiffServ domain as a set of DiffServ routers (r_i) interconnected with unidirectional routing path. A unidirectional link (l_i) exists from router r_1 to r_2 if and only if there is a direct routing path from r_1 to r_2. Thus, a topology of a DiffServ domain, T can be denoted as follows.

$$T = R \cup L, \text{ where } R = \{ r_1, r_2, r_3, \ldots, r_m \}, \text{ a set of } m \text{ routers}$$
$$L = \{ l_1, l_2, l_3, \ldots, l_n \}, \text{ a set of } n \text{ links}$$

In addition to the topological modeling, the dynamic traffic information should be modeled as well. At a certain moment, there is a traffic flows between a set of edge routers within the DiffServ domain. We denote this as a DiffServ flows. Since a DiffServ domain carries different classes of DiffServ traffic, a DiffServ flow of class c_i, $D(c_i)$ can be represented as follows.

$$D(c_i) = (T_i, Q_i), \text{ where } T_i = R_i \cup L_i \subseteq T$$
$$Q_i = \{\, Q(c_i, r_k) \mid r_k \in R_i \,\}$$

T_i is a subset of topology T and contains a set of routers and links, where the DiffServ flow of class c_i is detected and monitored.

Edge-to-edge DiffServ flow of concern is a set of IP packets flowing from one edge router to another edge router with the same DSCP value in a certain period of time. Thus, the edge-to-edge DiffServ flow of class c_i, source edge router r_s, and destination edger router r_d can be denoted as $D(c_i, r_s, r_d)$ and represented as the following equation.

$$D(c_i, r_s, r_d) = (T_{i,r_s,r_d}, Q_{i,r_s,r_d}), \text{ where } T_{i,r_s,r_d} = R_{i,r_s,r_d} \cup L_{i,r_s,r_d} \subseteq T_i$$
$$Q_{i,r_s,r_d} = \{\, Q(c_i, r_k) \mid r_k \in R_{i,r_s,r_d} \,\}$$

T_{i,r_s,r_d} is a subset of DiffServ flow T_i and represents the routing path from source edge router, r_s, to destination edge router, r_d.

Fig. 4 depicts mapping relationships of three hierarchical models. Topology model contains a set of DiffServ flows of different DiffServ classes, and a DiffServ flow model contains a set of different source/destination edge-to-edge DiffServ flows. The graphical representation reveals how to extract an edge-to-edge DiffServ flow from a topology of a DiffServ domain.

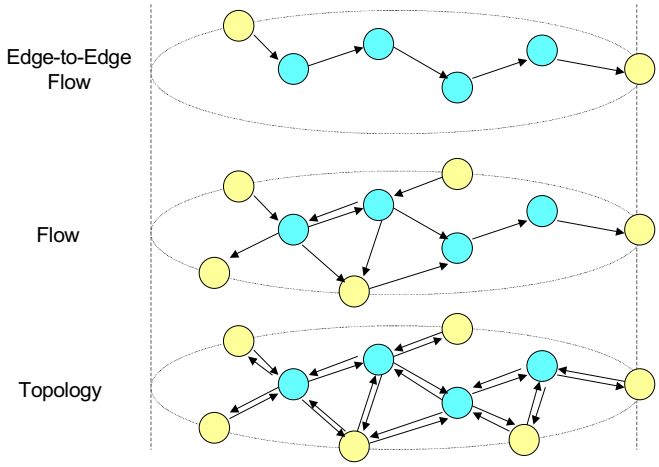

Fig. 4. Mapping Relationships of Topology, Flow, and Edge-to-Edge Flow

When managing DiffServ networks, it is beneficial to possess information on edge-to-edge DiffServ flows in a DiffServ domain. First, edge-to-edge DiffServ flows can be easily mapped to various network services provided by a DiffServ domain. Topology, status, and performance of the given network service is easily retrieved from the edge-to-edge DiffServ flows information. Thus, service management operations can be simplified and various high-level management operations, such as

accounting and billing, SLA negotiation and monitoring, can be constructed on the edge-to-edge DiffServ flows. Secondly, routing decisions can be combined with traffic conditioning decisions. Without edge-to-edge DiffServ flows, routing is totally independent of traffic conditioning. This might improve the performance of routers, but the separation makes traffic conditioning ignorant of the routing topology. If traffic conditioning decisions can be made with the knowledge of routing or vice versa, better decisions would be made. The edge-to-edge DiffServ flows combine the routing and the traffic conditioning to assist both decisions to become more effective and efficient. Lastly, runtime dynamics of network traffic are represented in edge-to-edge DiffServ flows. Dynamically changing behaviors can be classified in the DiffServ flows, and administrators can easily understand and manipulate network traffic by managing DiffServ flows, not by managing individual routers.

4 Distributed QoS Measurement and Edge-to-Edge QoS Aggregation of DiffServ Flows

When managing edge-to-edge QoS of a DiffServ flow, we can use two different approaches. One is measuring QoS at each edge point, and the other is measuring QoS in every routing point on routing path. Our approach is a distributed measurement of QoS information at every transit router between two edge routers. And the locally measured QoS information is aggregated by predefined rules. The distributed QoS measurement and aggregation method is explained in Fig. 5.

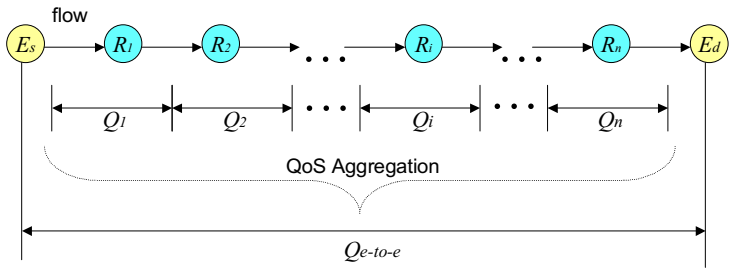

Fig. 5. Distributed QoS Measurement and Aggregation

Following the routing path from the source edge router, E_s, to the destination edge router E_d, there are n core routers from R_1 to R_n located serially. Each core router can monitor and collect QoS information of traffic flows and keep the information as Q_i. The distributed QoS observation from one router does not interact with other routers so that every transit router gathers QoS information independently. When the edge-to-edge QoS is needed, the locally-observed QoS are aggregated hop by hop.

Information obtained from edge-to-edge DiffServ flows consists of two parts: topology and performance. Topology information represents router-to-router connectivity. A path from a set of source edge routers to a set of destination edge routers must be provided. Performance information represents a number of performance parameters of a given DiffServ path. The performance information can

be obtained by combining performance parameters of each router in a DiffServ path.

4.1 Construction of Edge-to-Edge DiffServ Flows

An edge-to-edge DiffServ flow is required to have topology and performance information. Topology information is constructed from the routing tables in each router and performance information is constructed from DiffServ MIB [17] values in each DiffServ router. Constructing edge-to-edge DiffServ flows thus consists of two phases, as in Fig. 6. First, the topology generator produces topology information as a linked list of a set of routers and the performance analyzer aggregates performance parameters of each router in the routing path by using topology information. MIB II [18] and DiffServ MIB are used to construct the edge-to-edge DiffServ flows.

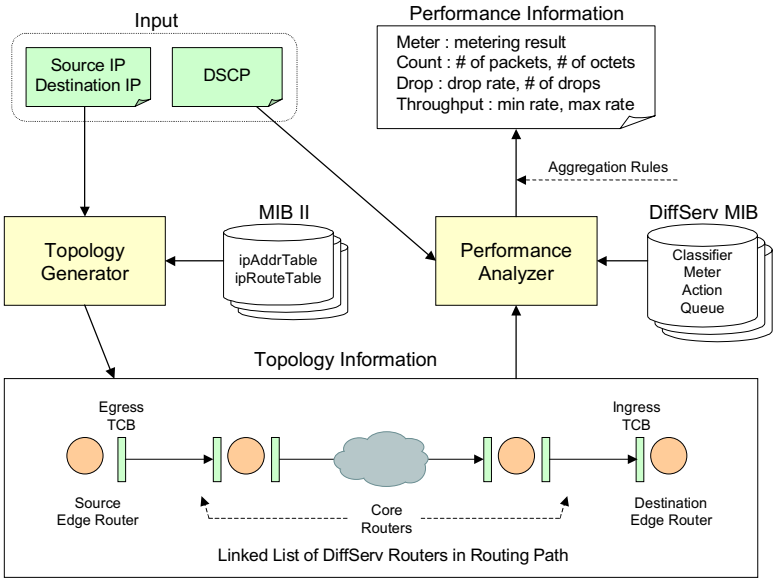

Fig. 6. Construction of Edge-to-Edge DiffServ Flows

Since each DiffServ router supports routing protocols, the router keeps a routing table that contains a list of the next hop routers for a given destination IP address. The MIB II standard has MIB objects for containing the routing table. A central SNMP manager can retrieve the routing table information to construct a whole routing connectivity map in a DiffServ domain. Two MIB tables, an ipAddrTable and an ipRouteTable, can be used to create topology information. The ipAddrTable contains IP addresses of all network interfaces in a router and the ipRouteTable contains the IP routing table that has the next hop host and network interface for a set of destination IP addresses. By combining the two table entries we can obtain every source-to-destination routing path. Given the source and destination IP addresses, the topology generator outputs a linked list of DiffServ routers composing the routing path.

Performance information of edge-to-edge DiffServ flows is obtained from the

DiffServ MIB. Each DiffServ router has performance parameters observed locally. The parameters include metering parameters, counter values, numbers of dropped packets, minimum and maximum rates of packet transmission, and so on. These parameters are calculated and maintained for each DSCP value; that is, the DiffServ MIB of a DiffServ router contains all the performance parameters of DiffServ flows it processes. When a linked list of routers composing a DiffServ routing path is given, the performance analyzer aggregates the values of the parameters from each DiffServ router one by one and produces edge-to-edge performance information of a DiffServ flow.

4.2 Graphical Representation and Edge-to-Edge QoS Aggregation Rules

To represent edge-to-edge DiffServ flows more efficiently, we devised a graphical notation of topology. We show basic graphical notations of DiffServ flows to understand the topological information of each DiffServ flow.
Fig. **7** illustrates four different kinds of DiffServ node representations, composed of vertex and directed edges.

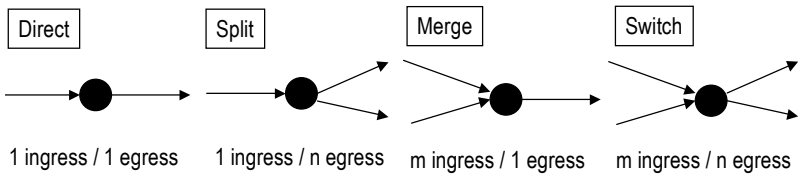

Fig. 7. Graphical Representation of DiffServ Nodes

A direct node receives the DiffServ flow from one direction and forwards it to only one direction. A split node receives the DiffServ flows from one direction but forwards them to two or more directions. A merge node receives the DiffServ flow from multiple directions and combines them to one outgoing direction. A switch node splits and merges at the same time from multiple incoming directions to multiple outgoing directions. The reason why we classify different types of DiffServ nodes is that different kinds of network nodes require different parameter aggregation rules.

The edge-to-edge QoS aggregation rules are used when the performance analyzer construct edge-to-edge performance information following the routing path. Locally observed performance parameters from DiffServ MIB are aggregated differently according to the kind of node in the routing path.

Table **1** summarizes the different aggregation rules for each parameter and node combination. Three performance parameters, throughput, delay, and number of drops, are explained as examples.

Table 1. Aggregation Rules for Different Parameter / Node Combination

	direct	split	merge	switch

throughput	`find_min`	`find_next_hop`
		`find_min`
			`calculate_`
			`portion`
			`find_min`
				`find_next_hop`
				`calculate_`
				`portion`
				`find_min`

delay	`add`	`find_next_hop` `add`	`add`	`find_next_hop` `add`

number of drops	`add`	`find_next_hop`
		`add`
			`calculate_`
			`portion`
			`add`
				`find_next_hop`
				`calculate_`
				`portion`
				`add`

There are four different operations for building up aggregation rules. One aggregation rule consists of one or several individual operations according to the kind of parameter and node combination. A `find_min` operation finds a smaller value between the ingress parameter and the egress parameter. Since the edge-to-edge throughput of a DiffServ flow is bounded by the link of the minimum throughput, a `find_min` operation is used to extract the throughput parameter. An `add` operation adds the egress parameter to the ingress parameter, so that the edge-to-edge parameter can be accumulated to the end of routing path. The delay of a link and number of dropped packets can be calculated by summing up all the values on the routing path. For the nodes with multiple egress links, such as split and switch nodes, a `find_next_hop` operation must be performed before other operations to specify one egress link that the edge-to-edge DiffServ flow of concern actually uses for the next routing path. For the nodes with multiple ingress links, such as merge and switch nodes, a `calculate_portion` operation calculates the fraction of the ingress traffic occupied in the egress traffic. The calculation can be represented by the amount of current throughput of DiffServ flow in the ingress link divided by the throughput of the next hop egress link. The following equation provides the value of the fraction from the operation.

$$\text{portion} = \frac{\text{current throughput of DiffServ flow in the ingress link}}{\text{throughput of the next hop egress link}}$$

The `calculate_portion` operation is important for extracting the edge-to-edge DiffServ flow from mixed DiffServ flows in a link. If there are multiple ingress links and the DiffServ node does not distinguish different edge-to-edge DiffServ flows because of the same DSCP value, every egress link might have a mix of different edge-to-edge DiffServ flows. However, since we know the current amount of throughput of the edge-to-edge DiffServ flow in the ingress link and the DiffServ node treats packets the same way at the egress link, we can assume that performance parameters, such as throughput and number of the dropped packets in the egress link,

are divided according to the fraction of the amount of ingress link over the amount of egress link. For example, if the throughput of edge-to-edge DiffServ flow of concern is 10Mbps in the ingress link, while the throughput of the next hop egress link is 20Mbps, we assume that half of the egress link traffic comes from other edge-to-edge DiffServ flows. In this situation, the result of the `calculate_portion` operation is 0.5, and so the throughput and number of dropped packets of the egress link should be multiplied by the fraction to extract parameters of edge-to-edge DiffServ flow of concern. A certain kind of performance parameters, such as delay parameter, does not use the `calculate_portion` operation because of its inherent characteristics.

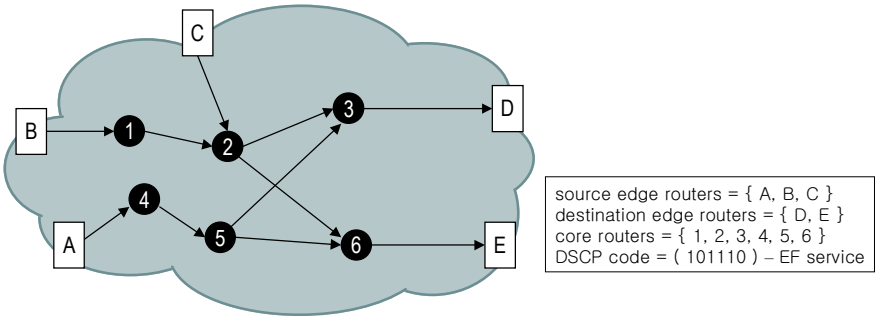

Fig. 8. Example of Edge-to-Edge DiffServ Flows

Fig. 8 is a graphical representation of an example snapshot of DiffServ flows of class EF. In this snapshot, three source edge routers (A, B, and C) and two destination edge routers (D and E) exist. Six core routers (labeled 1 through 6) are interconnected with each other by directed edges representing IP routing paths. According to our graphical representation, router 1 and 4 are direct nodes, router 5 is a split node, router 3 and 6 are merge nodes, and finally router 2 is a switch node.

The edge-to-edge DiffServ flow from edge router A to edge router D is constructed as follows. According to IP routing tables, the topology generator extracts the routing path from router A to router D as A – 4 – 5 – 3 – D. QoS information observed by the core routers are aggregated as follows. Since router 4 is a direct router, performance parameters are simply aggregated at this point. Next, router 5, the DiffServ flow is separately forwarded to core routers 3 and 6, and the performance parameters from router A to router 4 are divided to reflect the division. Router 3, a merge router, combines DiffServ flows from 2 and 5, and the performance parameters from router 3 to router D is divided to reflect the aggregation.

5 Conclusion and Future Work

A best-effort service model for the Internet is simple and easy to maintain. However, the model does not satisfy various QoS requirements in the Internet, especially when network bandwidth becomes scarce. The Internet is currently facing an urgent need for QoS support from various network users and applications. Differentiated Services (DiffServ) is gaining acceptance as a promising solution towards providing QoS

support in the Internet. When DiffServ is deployed in the backbone networks, managing it is necessary to manage the DiffServ domain by monitoring topology and performance parameters from DiffServ network elements. However, though the operational architecture of DiffServ is becoming mature and stable by the standardization efforts from the IETF DiffServ working group, the management aspect must be refined and extended.

In this paper, we have proposed a method for measuring and constructing QoS of edge-to-edge DiffServ flows within a DiffServ domain. Distributed measurement of QoS information is aggregated by predefined aggregation rules according to topological DiffServ model. By following our methods, the administrators of DiffServ networks can obtain dynamic status of network behaviors very efficiently and also pinpoint bottleneck points and possible solutions when edge-to-edge QoS is not satisfied when congestion occurs.

To verify the suggested methods, we are building a testbed environment with several DiffServ routers configured in Linux platform. We are also simulating more complicated topology and traffic patterns by using ns-2 network simulator [19]. Experiment and simulation results will provide more helpful information to understand our methods and applicability.

A systematic method for representing the proposed DiffServ traffic aggregate information is needed. The proposed construction process and graphical notation must be extended to produce a formal and complete description of the edge-to-edge traffic aggregates. Standardized data formats and extended graphical representations are under research. Further, mathematical formulation on hop-by-hop aggregation for edge-to-edge traffic aggregates requires much more research, especially concerning multiple source and destination edge routers.

References

1. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An Architecture for Differentiated Services,” IETF RFC 2475, December 1998.
2. R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: an Overview,” IETF RFC 1633, June 1994.
3. K. Nichols, S. Blake, F. Baker and D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,” IETF RFC 2474, December 1998.
4. V. Jacobson, K. Nichols, and K. Poduri, “An Expedited Forwarding PHB,” IETF RFC 2598, June 1999.
5. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, “Assured Forwarding PHB Group,” IETF RFC 2597, June 1999.
6. N. Brownlee, C. Mills, G. Ruth, “Traffic Flow Measurement: Architecture,” IETF RFC 2063, January 1997.
7. N. Brownlee, “Traffic Flow Measurement: Experiences with NeTraMet,” IETF RFC 2123, March 1997.
8. Cisco NetFlow white paper, http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm.
9. S. Waldbusser, “Remote Network Monitoring Management Information Base,” IETF RFC 2819, May 2000.
10. S. Waldbusser, “Remote Network Monitoring Management Information Base Version 2 Using SMIv2,” IETF RFC 2021, January 1997.

11. Y. Jiang, C. Tham, and C. Ko, “Challenges and Approaches in Providing QoS Monitoring,” International Journal of Network Management, October 2000, pp.322-334.
12. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, “NetScope: Traffic Engineering for IP Networks,” IEEE Network, Vol.14, No.2, March/April 2000, pp.11-19.
13. Y. Bernet, D. Durham, and F. Reichmeyer, “Requirements of Diff-serv Boundary Routers,” IETF Internet-Draft, draft-bernet-diffedge-01.txt, November 1998.
14. Y. Bernet, A. Smith, S. Blake, and D. Grossman, “A Conceptual Model for Diffserv Routers,” IETF Internet-Draft, draft-ietf-diffserv-model-03.txt, May 2000.
15. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, “The COPS (Common Open Policy Service) Protocol,” IETF RFC 2748, January 2000.
16. R. Yavatkar et al., “COPS Usage for Differentiated Services,” IETF Internet-Draft, draft-ietf-rap-cops-pr-00.txt, December 1998.
17. F. Baker, K. H. Chan, and A. Smith, “Management Information Base for Differentiated Services Architecture,” IETF Internet-Draft, draft-ietf-diffserv-mib-03.txt, May 2000.
18. W. Stalling, SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, 3rd Edition, Addison-Wesley, 1999.
19. ns-2 network simulator, http://www.isi.edu/nsnam/ns/.