ISSN:0975-9646

A Novel Architecture Providing Scalable Quality of Service - Adaptive Services

HarishBabu.Kalidasu , Bhagyalakshmi.Basuvula ,Haripriya.P

Priyadarshini Group of Institutions – Tenali, Andhra Pradesh, India

Abstract- QoS architectures define how routers process packets to ensure QoS service guarantees enforced. Existing QoS architectures such as Integrated Services (IntServ), Differentiated Services (DiffServ), and Dynamic Packet State (DPS) share one common property that the packet structure and the function of the routers are closely connected. Packets of one data flow are treated the same all the time at different routers. We propose to decouple such connection between packet structures and router functions. In our solution, packets carry as much information as possible, while routers process packets as detailed as possible until their load burden prohibits. We call such novel QoS architecture Adaptive Services (A-Serv). A-Serv utilizes our newly designed Load Adaptive Router to provide adaptive QoS to data flows. Treatments to data flows are not predefined but based on the load burden in Load Adaptive Routers. A-Serv overcomes the scalability problem of IntServ, provides better service guarantees to individual data flows than DiffServ and can be deployed incrementally. Our empirical analysis results show that compared with DiffServ architecture, A-Serv can provide differentiated services to data flows in the same DiffServ class and can provide better guaranteed QoS to data flows. Furthermore, AServ provides better protection to data flows than DiffServ when malicious data flows exist.

Keywords: Network QoS, QoS Architecture, Internet, Packet Format

1. INTRODUCTION

Today's Internet only provides best-effort service, where traffic is processed as quickly as possible, and there is no guarantee to the quality of service (QoS) for data flows. Here, a data flow is composed of packets with same flow ID, which is normally represented by 5-tuple (source IP address, destination IP address, transport protocol, source transport protocol port, and destination transport protocol port). QoS refers to the capability of a network to provide better service to selected network traffic. Services provided by QoS can be described by parameters such as delay, packet loss rate, jitter, throughput, etc.

IntServ, DiffServ, and DPS architectures share one common property that packet structures and router functions are closely connected. Therefore treatments to data flows are predefined. In IntServ, per-flow treatment is binded with the 5-tuple flow ID in each packet. In DiffServ, class based aggregated treatments are binded with the DiffServ class ID in each packet. In DPS, treatments are binded with a 17 bit DPS header defined in IP header. In these QoS architectures, packets of one data flow are treated the same all the time at different routers based on their packet header structure. We propose to decouple such connection between packet structures and router functions. Packets will carry as much information as possible, while routers process the packets as detailed as possible until their load burden prohibits. We call such novel QoS architecture Adaptive Services (A-Serv).

In this paper, we first introduce a Load Adaptive Router, which treats data flows differently according to the load burden of the router. Then we propose a novel QoS architecture, Adaptive Services (A-Serv), which uses Load Adaptive Routers as core routers. Based on router's processing capability, A-Serv keeps as much data flow state information as possible at core routers. In A-Serv, a data flow can be treated either as per flow or aggregate traffic in the core routers depending on the core routers' load burden. Behavior of Load Adaptive Routers in A-Serv architecture is presented in pseudo codes. A new protocol is also proposed to be used in A-Serv to provide guaranteed per-flow treatment for data flows with high priority.

A-Serv is free from Intserv's scalability problem because it adjusts its treatments to data flows based on its actual processing, storage and scheduling burden. On the other hand, A-Serv canprovide better service guarantee to individual data flow than DiffServ by trying to make full usage of routers' processing ability to guarantee services of as many as possible data flows. In addition, A-Serv can be deployed incrementally on existing QoS architectures. Our empirical analysis results show that A-Serv can provide differentiated service to data flows in the same DiffServ class and protect data flows better than DiffServ from malicious data flow's interference.

2. RELATED WORKS

In this section, we present our network model, and a brief review of network QoS and existing QoS architectures, such as IntServ [1], DiffServ [2] and DPS [3].

2.1 Network Model

In our network model, all routers in a domain are categorized into edge routers and core routers. Edge routers are boundary points at which data flows enter (ingress edge router) or leave (egress edge router) this domain. Edge routers connect to access networks or to edge routers in other domains. Core routers are internal routers that connect different edge routers in the same domain.

2.2 Network QoS

With the appearance of more and more network applications requiring contents to be delivered with ensured quality, best effort service is not capable of handling all kinds of network applications anymore. Quality of Service (QoS) refers to the capability of a network to provide better service to selected network traffic or data flows. QoS is the ability of a network element (e.g. an application, host or router) to have some degree of assurance that its traffic and service requirements can be satisfied [13]. Components in QoS include 1) QoS architectures, 2) admission control, 3) scheduling and policing, and 4) QoS routing. QoS architectures define how routers process packets to ensure that QoS service guarantees are enforced. In order to provide QoS on the Internet, many QoS architectures have been proposed such as Integrated Services (IntServ) [1], Differentiated Services (DiffServ) [2] and Dynamic Packet State (DPS) [3], etc. To fulfill QoS requirements, admission control processes are used.

2.3 QoS Architectures

IntServ architecture is characterized by resource reservation for each data flow through RSVP signaling protocol [4, 5]. A data flow requesting specific QoS guarantees is required to initiate asetup procedure using RSVP. RSVP sets up “soft states” in each router along the path from source to destination specifying the resource requirements of the initiating data flow. The reservations remain valid as long as the data flow is active, and expire if not refreshed periodically. All routers, including edge routers and core routers, keep the per-flow state information and allocate resources (such as buffer space and link bandwidth) to each passing data flow. This per-flow service model achieves the maximum flexibility as it can meet QoS requirements of individual flows. In IntServ, Packets are identified by their flow ID (5-tuple with 104 bits in IPv4 or 296 bits in IPv6 [6]).

3. NEW ARCHITECTURE USING LOAD ADAPTIVE ROUTER

In this section, we first introduce a newly designed Load Adaptive Router. Load Adaptive Routers can be deployed as core routers in single or multi QoS domains and can provide scalable, differentiated and adaptive services. Then, we use Load Adaptive Routers to design a new adaptive QoS architecture, A-Serv. Behavior of routers in A-Serv is presented by pseudo code. A protocol is designed to provide end-to-end per-flow treatment to data flows.

3.1 Load Adaptive Routers

Load adaptive routers are designed to be deployed as core routers in QoS capable domains as shown in Figure 1.

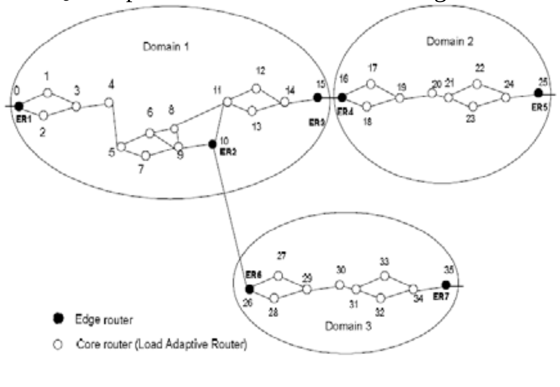


FIGURE 1: Load Adaptive Router in QoS Capable Domain

ER1~ER7 are 7 edge routers and the rests are core routers, which can be implemented as load adaptive routers.

The key idea of this design is to decouple packet structure from the way packets are processed in core routers. In IntServ, routers always treat packets based on their flow ID. By identifying the data flow each packet belongs to, the packet is treated based on the resources reserved by that data flow. In DiffServ, packets are treated in core routers based on their DiffServ class ID in header. In DPS architecture, packets are processed by routers based on the scheduling information in packet headers. In our newly designed load adaptive router, a packet can be treated differently in routers depending on the load of each router while the packet carries as much information as needed for all possible processing.

Here we use the packet header formats of TCP/IP packet in IPv4 and IPv6 shown in Figure 2 to illustrate the load adaptive idea. One thing worth mentioning is that the load adaptive idea itself doesn’t involve any assumption from current IP network and it is not limited in TCP/IP packet and can be easily adapted to other existing or future network standards. When the load adaptive scheme is implemented in TCP/IP network, packets can be treated based on their flow ID (104 bits in IPv4 or 296 bits in IPv6 in Figure 2) as a per flow, which is equivalent to IntServ, or be treated based on class ID (6 bits in DS field in Figure 2) as aggregate traffic, which is equivalent to DiffServ, or anything in the middle by identifying a number of bits in the packet header (the number of bits between 6 and 104, which can be inserted in the option field in IPv4 and IPv6). Instead of having 2-8 classes as in DiffServ, we could have a class to contain only one data flow if that data flow is more important. Instead of having one per-flow state for each data flow, we could aggregate data flows together into different kinds of classes, which are then treated as aggregate traffic, but not necessarily to be the DiffServ classes with 6 bit DiffServ class ID.

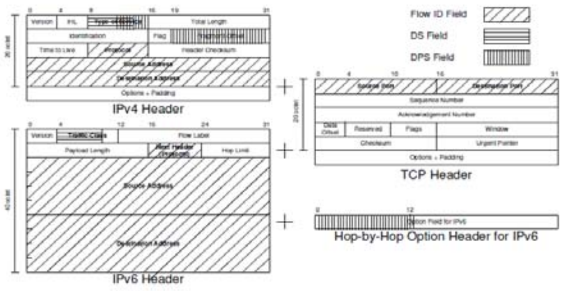


FIGURE 2: 5 -Tuple Flow ID, DS and DPS Field in TCP/IP Header

3.2 A-Serv: Adaptive Services Provided by Load Adaptive Router

In this subsection, we introduce a new QoS architecture called A-Serv, which utilizes load adaptive routers as core routers to provide Adaptive Services to data flows. In A-Serv, edge routers insert default data flow aggregation information (such as class ID) into packets’ eader. The default data flow aggregation can be the DiffServ class or any other aggregation scheme defined in the future.

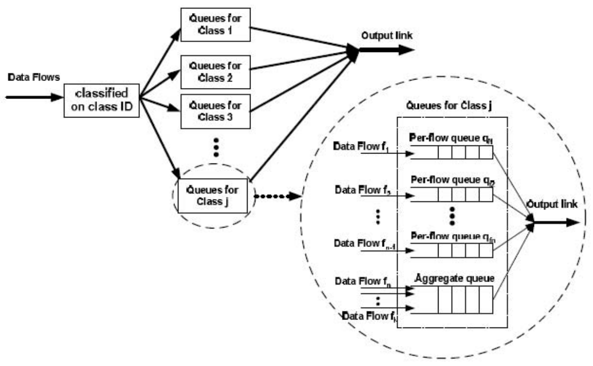


FIGURE 3: Data Flow Treatment in A-Serv

A. Adaptive Service

In A-Serv, data flows' treatments in a load adaptive router are shown in Figure 3. A load adaptiverouter first classifies data flows based on its default class ID. Within each class, a data flow is treated either as per flow or aggregate traffic. For example, data flow f1 is treated as per flow in class j in Figure 3. A separate queue qf1 with desired buffer and bandwidth assignment would be set up for data flow f1. Thus f1 receives guaranteed service at this load adaptive router. Here wesay f1 is treated as per flow (f1 receives per-flow treatment) and qf1 is the per-flow queue for data flow f1. Packets belonging to f1 are inserted into queue qf1 in this router after being identified by their class and flow ID. For data flow fn, all packets of fn would be identified by the class ID only in this router and be inserted into the aggregate queue for class j, which is responsible for forwarding all the data flows being treated as aggregate traffic in class j. Here we say fn is treated as aggregate traffic (fn receives aggregate treatment) in class j. A-Serv is a two level adaptive implementation of load adaptive routers.

B. Load Burden Estimation

Load adaptive routers use load burden to determine treatments to data flows (per-flow or aggregate). In our design, load burden is evaluated through the number of existing queues in load adaptive routers. The number of existing queues affects processing complexity, scheduling complexity and consumed storage space of load adaptive routers. The more existing queues in a router, the more storage space is used to store state information for each queue which records queue's resource reservation; the more time needed to perform classification for incoming packets to decide destination queues to insert the packets; the more time spent on scheduling the packets transmission. For example, in General Processor Sharing (GPS) scheduling the start/finish virtual time for each packet is computed for scheduling purpose. Such computation has complexity of O(logn), where n is the number of queues in scheduling process [68].

However, number of queues is not the only choice for load burden estimation. Other factors that constraint the scalability can also be adopted into the load burden criteria. For example, the load burden criteria could be the utilization of buffers, if the buffer space used to store per flow information becomes the bottleneck and limits the scalability. Also more sophisticated situation in load burden evaluation arises when we consider the data flow's traffic properties such as bustiness, traffic volume, packet size, etc.
In this paper, we only consider the situation when the number of queues in the router cases scalability problem and we use it as load burden criteria in our design.

C. General Functions in a Load Adaptive Router

In this part, we present pseudo codes for general functions in a load adaptive router, includingnew data flow establishment, incoming packet processing and existing data flow's termination.
In A-Serv architecture, there is an upper bound N on the number of existing queues in a load adaptive router. N is decided by the router's processing ability and storage capacity. Different load adaptive routers could have different values for N and this enables gradual deployment and upgrade. In one router r, N is partitioned into nj, j=1,2,..k, when there are k default classes defined in total. nj is the number of queues that could be supported simultaneously in this router for class j. Each default class reserves one queue for aggregate traffic. Therefore, simultaneously at most nj-1 data flows in class j can be treated as per flows.
The resource allocation is shown in Figure 4. The initial resources allocation between classes is determined by the network domain manager or operator. The resources within each class are assigned to per-flow queues and the aggregate queue as we described above.

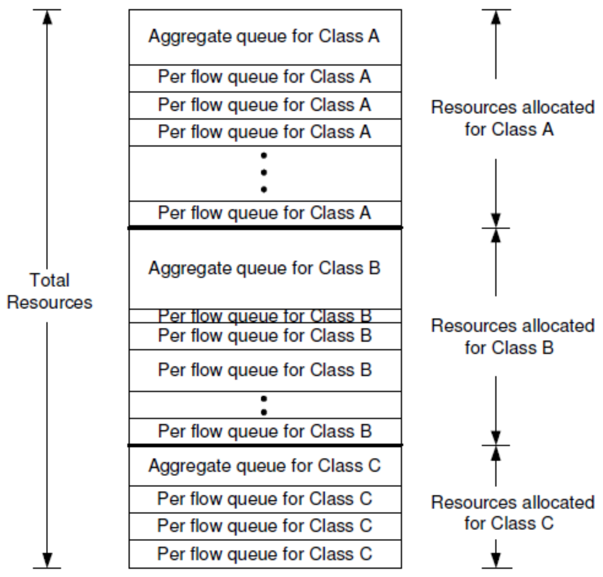
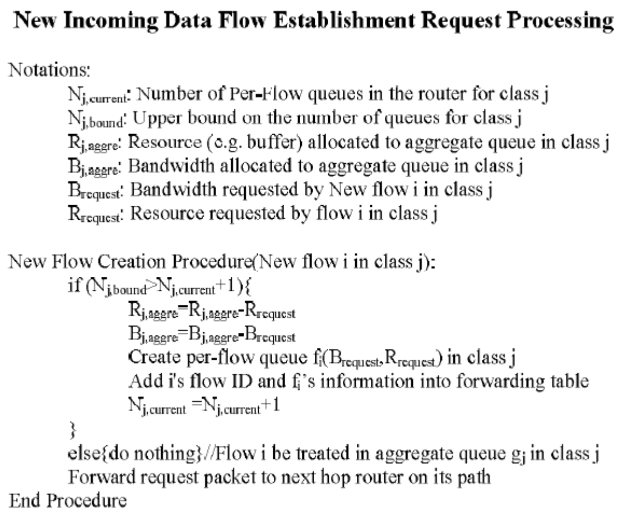

Total Resources
Aggregate queue for Class A
Per flow queue for Class A
Per flow queue for Class A
Per flow queue for Class A
Per flow queue for Class A
Aggregate queue for Class B
Per flow queue for Class B
Per flow queue for Class B
Per flow queue for Class B
Per flow queue for Class B
Aggregate queue for Class C
Per flow queue for Class C
Per flow queue for Class C
Per flow queue for Class C
Resources allocated for Class A
Resources allocated for Class B
Resources allocated for Class C

FIGURE 4: Resource Allocation in A-Serv

When a new data flow needs to be established, a request packet (or the first data packet in the data flow) will be processed in the load adaptive routers on its path following pseudo codes in Figure 5. Each router will try to create a per-flow queue for the new data flow. If the per-flow queue quota, nj, is used up by existing data flows, the new data flow will be treated in aggregate queue.



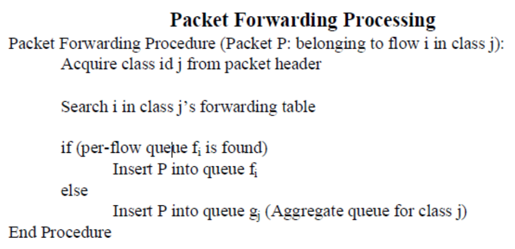
New Incoming Data Flow Establishment Request Processing

```
Notations:
        $N_{j,current}$: Number of Per-Flow queues in the router for class j
        $N_{j,bound}$: Upper bound on the number of queues for class j
        $R_{j,aggre}$: Resource (e.g. buffer) allocated to aggregate queue in class j
        $B_{j,aggre}$: Bandwidth allocated to aggregate queue in class j
        $B_{request}$: Bandwidth requested by New flow i in class j
        $R_{request}$: Resource requested by flow i in class j

New Flow Creation Procedure(New flow i in class j):
        if ($N_{j,bound}$>$N_{j,current}$+1){
                $R_{j,aggre}$=$R_{j,aggre}$-$R_{request}$
                $B_{j,aggre}$=$B_{j,aggre}$-$B_{request}$
                Create per-flow queue $f_i$($B_{request}$,$R_{request}$) in class j
                Add i's flow ID and $f_i$'s information into forwarding table
                $N_{j,current}$ =$N_{j,current}$+1
        }
        else{do nothing}//Flow i be treated in aggregate queue $g_j$ in class j
        Forward request packet to next hop router on its path
End Procedure
```

FIGURE 5: New Data Flow Establishment

After data flow's establishment, when subsequent data packets come into the router for forwarding, the router will first check the packet's header for class ID, say class j. Then the routerwill search its flow ID in the forwarding table of class j. If there is a match, this packet will be inserted into the corresponding queue for the matched entry in forwarding table. If there is no match, the packet will be inserted into the aggregate queue for class j. Such procedure for packet forwarding is depicted as Figure 6.



Packet Forwarding Processing

```
Packet Forwarding Procedure (Packet P: belonging to flow i in class j):
        Acquire class id j from packet header

        Search i in class j's forwarding table

        if (per-flow queue $f_i$ is found)
                Insert P into queue $f_i$
        else
                Insert P into queue $g_j$ (Aggregate queue for class j)
End Procedure
```

FIGURE 6: Packet Forwarding in A-Serv

4. CONCLUSION AND FURTHER WORK

We introduce a new QoS architecture called A-Serv using load adaptive router to provide Adaptive Services. In the new QoS architecture, we propose to decouple the connection between packet structure and router functions. A-Serv architecture provides different packet treatments to data flows based on router's load burden. A-Serv overcomes the scalability problem of IntServ, provides better QoS guarantee than DiffServ and can be deployed incrementally. The analysis of data flows' performance in A-Serv under both with/without malicious data flow scenarios shows that A-Serv can provide better QoS guarantees than DiffServ without loosing scalability. The differentiated services provided to data flows in the same class in one router offer more options in providing variant and/or prioritized service to end users while maintaining the scalability. Furthermore, the adaptive property of A-Serv simplifies system upgrades. With the development in the router's processing ability, simply changing the per-flow number upperbound enables more data flows to receive guaranteed service without changing network protocols or packet header format.

Our design of A-Serv provides a new adaptive sight in QoS architecture design to deal with the tradeoff between scalability and QoS guarantee. In A-Serv, we constructed a general framework for such design and there are many detailed aspects and options need to be addressed in the future. One research issue is designing priority and scheduling scheme in A-Serv, which extends the load adaptive router's function to support more sophisticated data flow treatment schemes. Another further topic is how to design admission control algorithm to deal with the data flows receiving different services at different core routers. Different options of load burden criteria also need to be explored.

REFERENCES

[1] B. Braden, "Integrated services in Internet architecture - an overview," *Internet RFC-1633*, June 1994.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differential services," *IETF, RFC 2475,* December 1998,

[3] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," *Proc. Of ACM SIGCOMM,* pp. 81-94, September 1999.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) – version 1 functional specification," *RFC 2205*, Internet Engineering Task Force, September 1997.

[5] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource ReSerVation protocol," *IEEE Network*, September 1993.

[6] S. Bradner and A. Mankin, "The recommendation for the IP next generation protocol," *RFC 1752*, Internet Engineering Task Force, June 1995.

[7] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," *RFC 2598*, Internet Engineering Task Force, June 1999.

[8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group,"*RFC 2597*, Internet Engineering task Force, June 1999.

[9] J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, and J.Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding," *IEEE/ACM Transactions on Networking*, vol. 10, Issue 4, pp. 529-540, August 2002.

[10] A. Charny and J.Y. Le Boudec, "Delay bounds in a network with aggregate scheduling," *Proc. Of First International Workshop of QofIS'2000*, Berlin, Germany, September, 2000.

[11] V. Firoiu, J. -Y. Le Boudec, D. Towsley, Zhi-Li Zhang, "Theories and models for Internet quality of service," *Proc. Of the IEEE,* vol. 90, Issue 9, pp. 1565-1591, September 2002.

[12] Network Simulator 2 - NS2, http://www-mash.cs.berkeley.edu/ns.

[13] "The need for QoS," Stardust.com. White Paper, July 1999, http://www.qosforum.com