Achieving End-to-End Throughput Guarantee for TCP Flows in a Differentiated Services Network

Xiaoning He and Hao Che
Department of Electrical Engineering
Pennsylvania State University
State College, PA 16803

Abstract

A challenge for the differentiated services (DS) architecture is how to simultaneously provide end-to-end assured service (AS) for transport control protocol (TCP) sessions and the best-effort service traffic, based on a single queue management. Our simulation studies show that both intradomain and interdomain best-effort traffic can have adverse impact on the interdomain TCP traffic. This paper proposes a technique to achieve desired end-to-end throughput guarantee for TCP sessions. The proposed technique is composed of a series of measures which includes: (a) a path pinning mechanism for AS, which allows aggregated bandwidth reservation for AS at *each intermediate router* in the forwarding path; (b) a packet marking strategy; (c) a dropping policy; (d) two adaptive dropping-threshold algorithms for queue management. The simulation studies demonstrate that with this technique, a high end-to-end service assurance can be achieved for the TCP traffic, while a reasonably high throughput for the best-effort traffic is maintained. The proposed scheme is particularly viable when DS is supported by a multiprotocol label switched network.

Keywords: Differentiated Services,Assured Service for TCP,Performance measurement,Adaptive algorithm design,MPLS

1 Introduction

As the Internet evolves into a global commercial infrastructure, there is a growing need to support quality of service (QoS) to applications. However, the traditional QoS mechanisms based on per micro-flow reservation, such as the integrated services, do not scale well in the core of a network. Recently, a radical approach, known as differentiated services (DS) [1, 8], has attracted much attention. The DS model is based on the assumption that resources are *abundant* in the core and bottlenecks occur only at the border nodes between domains. While offering multiple classes of service (CoSs), the DS model ensures scalability by keeping a stateless core and adhering to the IP (i.e. Internetworking Protocol) hop-by-hop forwarding paradigm. However, a key problem for this model is the conflict between maximizing resource utilization and achieving a high service assurance. The connectionless IP forwarding paradigm fundamentally limits the predictability of resource availability for reservation-based services. Hence, in order to provide high service assurance, enough resources need to be provisioned to all the possible paths in the direction from a source to a destination.

Recently, Stoica and Zhang proposed a premium service model [3, 5]. The authors developed a technique (see [4]) that can pin a path without keeping per-flow state in the core routers and they designed algorithms to achieve end-to-end deterministic performance guarantee for premium service sessions using explicit paths. We believe that to provide end-to-end performance guarantee while achieving high resource utilization, a connection-oriented approach, such as the one proposed by these authors, is necessary. In fact, as we shall see shortly, both intradomain and interdomain best-effort traffic can adversely impact the performance of interdomain TCP traffic.

The focus of this paper is to design algorithms for providing end-to-end service assurance for TCP applications while achieving high resource utilization and maintaining high scalability. The approach taken is to enable connection-oriented AS with aggregated resource reservation. The connection-oriented AS can be enabled by using a route pinning technique proposed in this paper, the one proposed in [4] or by combining DS with a connection-oriented networking architecture such as multiprotocol label switching (MPLS) [6, 7]. The core routers along the path do not keep per-flow state information but only the aggregated bandwidth reservation information for AS sessions as a whole. A key design is to decouple the treatment of the AS traffic from the best-effort traffic by marking the AS traffic, both conformant and non-conformant, differently from the best-effort traffic. With the assured and best-effort traffic sharing a single output port queue, two *adaptive* queue management algorithms are proposed. The first algorithm is based on the aggregated reserved bandwidth for the AS traffic and the second one is based on both the aggregated reserved bandwidth for the AS traffic as well ase the measured level of aggregated conformant AS traffic. The former one is easier to implement than the latter, but it can result in low network revenue when the AS customers are charged by the number of bits they send, rather than the bandwidth they reserve. Simulation results show that the algorithms successfully suppress the negative impact of both interdomain and intradomain best-effort traffic on the performance of the interdomain AS traffic, resulting in a high end-to-end service assurance for the AS traffic.

The rest of the paper is organized as follows. Section 2 presents a background introduction on the current status of the AS design for TCP applications, and an analysis of the related problems. Section 3 describes the proposed scheme including a queue management algorithm based on the aggregated reserved bandwidth. Section 4 gives the experiment results on the proposed scheme. Section 5 proposes a queue management algorithm based on both the aggregated reserved bandwidth as well as the measured level of the aggregated conformant traffic. The performance of this algorithm is then compared with the one developed in Section 3. Finally, Section 5 concludes the paper and presents future research directions.

2 Background

The original idea for designing AS for TCP applications was proposed by Clark and Fang [9]. In this model, AS provides better-than-best-effort service. Each AS session using TCP receives a guaranteed minimum bandwidth or target rate. Traffic is policed at every Internet service

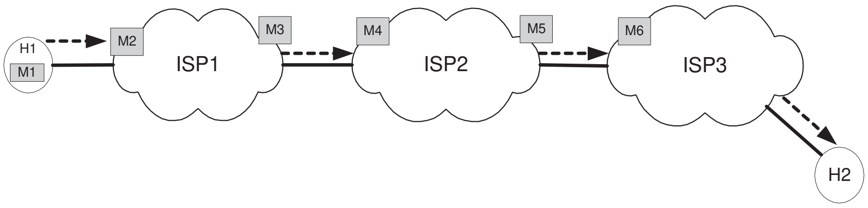

Figure 1: Multi-domain scheme proposed by Clark and Fang

provider (ISP) domain edge node, as shown in Fig. 1. At the edge node, conformant packets are marked as in-profile or IN and non- conformant packets are marked as out-profile or OUT. Both IN and OUT packets are injected into the core of the network, but the OUT packets are treated the same way as the best-effort packets. In each core router, a single first-in- first-out (FIFO) queue is used for both AS and best-effort traffic. A 2-level RED (i.e. random early detection) [12] packet dropping algorithm, called RIO (RED in-and-out) [9] is run based on traffic type. IN packets have a lower dropping probability than the best-effort packets and OUT packets. The best-effort packets and OUT packets have the same dropping priority. At each and every domain boundary, traffic is policed locally, and packets are subject to remarking before being injected into another domain, based on local congestion situation. This model is simple because it requires only a single queue management at each output port of a core router. However, since there is no end-to-end resource provisioning, an end-to-end service assurance is not guaranteed. Several other works on the improvement of this model in an attempt to achieve better service assurance and fairness were proposed [10, 11], all based on a connectionless forwarding paradigm.

The above studies did not consider end-to-end performance of the AS TCP sessions, in the presence of the possible cross traffic, especially, the cross best-effort traffic with small round-trip time (RTT). The cross traffic could occur within a domain or at a domain boundary. Among other issues, a question is whether local control at each domain boundary can guarantee end-to-end performance for AS flows that cross multiple domains. After all, the ultimate performance measure is end-to-end service assurance, local control at each domain boundary, although has merit in its own right, may not guarantee end-to-end performance. To answer this question, we did simulation tests on the approach proposed in [9], using NS-2 from LBNL (Lawrence Berkerley National Laboratory).

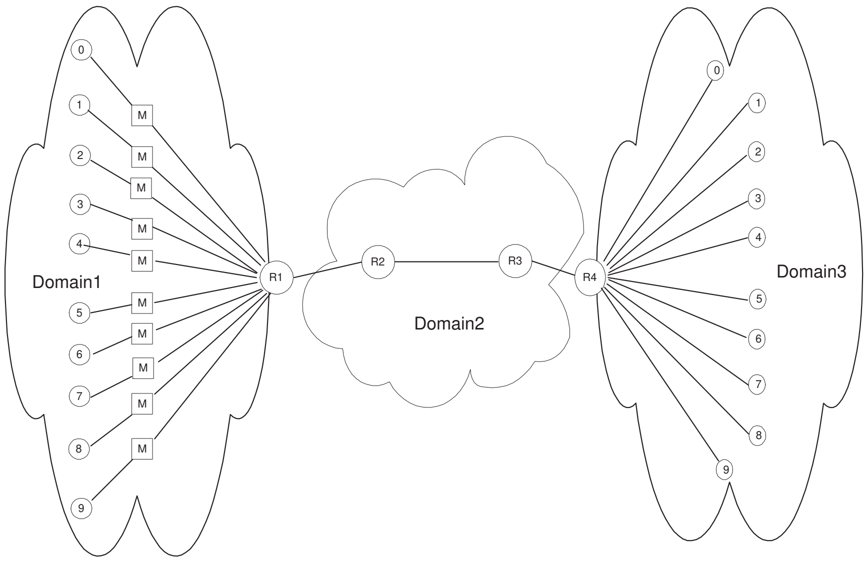

Figure 2: Network setup for experiment 1

The network setup is shown in Fig. 2. It is composed of three domains, with border routers R1 in domain 1, R2 and R3 in domain 2, and R4 in domain 3. The link bandwidth between routers are 33 $Mbps$ each and the buffer size is 50 packets for each output port of a router. There are ten hosts in domain 1 and each has an AS TCP session with one of the hosts in domain 3. Out of these ten sessions, 5 have target rates of 5 $Mbps$ and the other 5 have target rates of 1 $Mbps$. So the aggregated target rate for AS is 30 $Mbps$, which is lower than the link bandwidth between any two routers.

The parameters for RIO are set at $(min_{in}, max_{in}, P_{in}) = (40, 70, 0.02)$ for IN packets and $(min_{out}$, max_{out},$P_{out}) = (10, 30, 0.2)$ for OUT packets ,where min_{in} (min_{out}) is the threshold at which the router starts to drop incoming IN (OUT) packets with probability P_{in} (P_{out}), and max_{in} (max_{out}) is the threshold at which the router starts to drop all the incoming IN (OUT) packets. For more details on RIO, please refer to [9].

Two experiments were performed. In the first experiment, we assume that there is no cross traffic. In the second experiment, we added 50 best-effort TCP flows between nodes R2 and R3 with a RTT of 10 ms each, representing the intradomain cross traffic (we can also interpret it as the interdomain cross traffic coming into domain 2 via R2 and going out to other domains via R3).

Connecion	RTT(ms)	Target Rate	Test1	Test2
0	20	5	5.22/5.14	3.64/3.50
1	20	1	2.97/2.81	0.97/0.89
2	40	5	4.05/3.94	2.77/2.74
3	40	1	1.98/1.89	0.95/0.88
4	50	5	4.02/3.97	3.02/3.00
5	50	1	2.09/1.90	0.83/0.77
6	70	5	3.65/3.62	2.63/2.60
7	70	1	1.61/1.48	0.81/0.75
8	100	5	3.46/3.43	2.63/2.59
9	100	1	1.20/1.15	0.65/0.60
Total		30	30.25/29.37	18.91/18.33

Table 1: Throughputs/Goodputs for experiments 1 and 2

Table 1. summarizes the results for the two experiments. Let's first focus on the case without cross best-effort traffic. As one can see, the sessions with the lower target rate tend to achieve throughputs/goodputs higher than the target rate, whereas the sessions with the higher target rate tend to receive throughputs/goodputs lower than the target rate. This phenomenon was also observed in the previous papers [9, 2]. The related fairness issues were addressed in [2]. Here we focus on the aggregated throughput and goodput. As one can see, RIO achieves rather high aggregated service assurance with the aggregated throughput slightly higher and goodput slightly lower than the aggregated target rate. However, the situation becomes quite different in the presence of the cross best-effort traffic. One can see that the impact of the cross traffic on the end-to-end performance of the AS traffic is tremendous. Most of the AS sessions fall short of their target rates. Even worse, the achieved aggregated throughput/goodput are only about two third of the target value.

In summary, RIO can not guarantee end-to-end service assurance for the AS TCP sessions in the presence of cross best-effort traffic. One major reason behind this quality deterioration is that dropping an OUT packet and a best-effort packet have quite different impact on the performance of the two traffic types. Another major reason is that RIO is a static algorithm in the sense that the control parameters are set at fixed values, regardless of the AS traffic volume. We shall study these issues in more detail later.

3 Proposed Scheme

In this section, we propose a connection-oriented AS (COAS) as a first step towards a comprehensive solution to the above mentioned issues.

3.1 Path Pinning

COAS departs from the previous approach in that it introduces connection-oriented end-to-end AS for TCP applications. With COAS, there is no need to assume the resource abundance in the core of a domain, nor is there a need to distinguish domain border routers from core routers. An end-to-end connection setup and resource reservation (using, e.g., RSVP) is performed for each AS session. However, an immediate question is how to enable a connection-oriented service while retaining high scalability in the core. The answer is to pin the routed path for the life-time of an AS TCP session, such as the one proposed in [4]. It is worth mentioning that in an MPLS enabled Internet, connection-oriented DS can be easily support without designing separate path pinning mechanisms. Since MPLS is a connection-oriented service architecture based on label swapping, service classes can be easily supported by embeding multiple CoS-based trunks in a label switched path [6, 7]. In what follows, we propose a new route pinning mechanism and also briefly introduce the route pinning mechanism proposed in [?].

The idea is similar to the IP source routing in the sense that the path information is kept in each data packet of a session, not in the routers along the path. However, the difference is that the path information is represented by the output port numbers, not the IP addresses of the routers in the path and that it is the edge node of the source, not the source itself where this information is attached to the packet of a TCP session. The idea is to map the IP address of each router or router interface to a port number with local significance. When a signaling packet reserves resources along the path all the way back to the edge node of the source, it also copies the input port number of each router, or equivalently, the output port number for data flow. The edge node then caches the port-numbers-to-connection binding information in its cache table. When the edge node receives a data packet of an AS session, it attaches the port list in the options field of the IP header for packet forwarding. A router in the path forwards a packet by first checking the IP header to see if the AS bit is set. If it is, the router immediately reads the corresponding output port number in the IP options field for forwarding. If we assume that any core router has less than 256 interfaces with any other core routers and edge routers, and the maximum number of hops for any routed path is less than 30. Then the added overhead will be less than 30 bytes which is reasonably small. Notice that this scheme requires no change to the routing table nor the forwarding table. The only change is that a core router needs to associate its interfaces to all the other routers with a locally significant sequence of port numbers.

However, there are some related issues which need to be addressed. First, since the number of hops differs from one session to another and thus the number of port numbers in the port list varies from one session to another, whether to use a variable length or fixed length IP header needs to be decided. A simple solution is to use a fixed length with 30 bytes allocated for a port list. It is very unlikely that a core router will interface with more than 256 other routers. Also an end-to-end route is unlikely to exceed 30 hops. Second, there is an issue of robustness with respect to route changes. In general, path pinning mechanism usually cause transient packet loss when routes change and the information has not propagated back to the source. This issue can be solved by allowing a core router to immediately change a AS packet to a best-effort packet when it sees the output port the packet is destined to is down. Of course, this implies that in addition to the port list, each AS packet also needs to carry the destination IP address.

The route pinning mechanism proposed in [4] is to generate a path label which is the XOR over the IP addresses of all the routers along the path, and then associate this label with each packet of a flow that goes along that path. The path label is iteratively re-computed based on the XOR operation between the packet's current label and the node IP address. The initial path label is computed at the edge node and the session-to-label binding information is kept in a cache table for initial label binding and packet forwarding. The drawback of this approach is that it does not guarantee the uniqueness of the labels for different paths, although the chance of error is likely to be very small. This approach also requires that the routing table and forwarding table to be changed to include the path- label-to-next-hop binding information.

3.2 Marking Policy

For RIO, both OUT packets of AS flows and the best-effort packets have the same dropping priority. However, dropping an OUT packet and a best-effort packet could impact an AS flow and a best-effort flow in quite different ways. For example, for the case in Section 2, since the AS TCP sessions have much larger RTTs than the cross best-effort TCP flows, when there is a packet drop, it takes much longer time for an AS TCP session to open up its window than a best-effort flow does. As a consequence, most of the time, the best-effort traffic takes up a large portion of the output buffer no matter how much bandwidth is reserved for the AS traffic.

In order to suppress the impact of the best-effort flows, it is better to distinguish between OUT packets of an AS flow from the best-effort packets and treat them in different ways. This would allow a certain degree of decoupling of the two types of flows. In this paper, we modify RIO to allow three types of packets instead of two. The packet marking policy can be described as follows: (a) mark conformant AS packets as AS; (b) mark non-conformant AS packets as EX; (c) mark the best-effort packets as BE. In the present approach, a packet is marked only once at the edge node closest to the sender. There is no further remarking at the domain boundaries. How to treat the packets with different marks is the focus of the following subsection.

3.3 Dropping Policy and Adaptive Dropping-Threshold Algorithm

In this subsection, we design a dropping policy to suppress the BE traffic in a proper way so that it will give its way to the AS traffic.

We use the following dropping policy which generalizes RIO:

If (*Packet Type is AS*)

Process the packet in the same way as RIO processes an IN packet

Else if (*packet Type is BE and queue length of the best-effort packets* $> K_{be}$)

Drop the BE packet

Else

Treat EX and BE packets the same way as OUT in RIO

The only difference between this policy and RIO is that in this policy, a threshold K_{be} is imposed to upper bound the number of best-effort packets in the queue. To suppress the best-effort traffic without starving it, a proper design of K_{be} is crucial.

Intuitively, K_{be} should be a function of the total buffer size K, the link bandwidth B, and the aggregated reserved bandwidth B_{as} for the AS sessions at any given time t, i.e., $K_{be} = f(K, B, B_{as})$. Also, $f(K, B, B_{as})$ should be a monotonically decreasing function of B_{as} with boundary conditions $K_{be} = 0$ at $B_{as} = B$ and $K_{be} = K_t$ at $B_{as} = 0$. However, since the optimal buffer allocation problem that is traffic dependent is a hard one, we take a practical approach by designing an adaptive algorithm based on a given functional form. Note that the algorithm is adaptive in nature simply because B_{as} is a function of time. Also note that the approach only requires that each router keeps and updates a B_{as} value for each output port and thus high scalability is retained. With the path pinning technique proposed in Section 3.1, B_{as} can be easily updated at a router upon each arrival of either a connection setup or teardown packet.

To find a suitable functional form, we resort to the simulation tests, which will be done in the following section. We shall confine ourselves to the following group of functions:

$$K_{be} = K\left(\frac{B - B_{as}}{B}\right)^n, \quad \text{for } n\text{=1,2,3,...} \tag{1}$$

3.4 Establishment Procedure

In our approach, we treat the domain border routers the same way as the core routers and do not distinguish domain border routers from the core routers. An end-to- end connection setup and teardown for resource reservation and resource release, respectively, are required for an AS session. The intermediate routers keep and update the aggregated bandwidth reservation for AS at each output port by processing connection setup and teardown packets. Each intermediate router is also responsible for assigning the output port number to each connection setup packet. In addition to the roles an edge node plays in the traditional approach, it is also responsible for initiating the connection setup and teardown, and caching and assigning the port number list to each AS session coming into the network. In our approach, since the resource reservation is end-to-end, there is no need to do any packet remarkings at the border routers. The border routers play exactly the same role as the core routers. They run only the packet dropping algorithm proposed in the previous subsection.

4 Simulation Results and Analysis

	Target Rate	RIO	COAS(n=1)	COAS(n=2)	COAS(n=3)
1	5	6.20/23.12	6.20/22.55	6.30/22.37	6.95/21.85
2	10	8.66/20.14	8.74/20.08	8.72/20.25	10.06/19.01
3	15	10.73/18.32	10.81/18.26	12.77/16.60	16.86/12.88
4	20	13.03/16.22	13.33/15.98	18.71/11.12	24.20/5.77
5	25	15.61/13.94	16.97/12.87	24.76/4.70	27.64/2.92
6	30	17.25/12.47	24.15/6.28	28.50/1.47	28.31/1.38

Table 2: Aggregated throughputs/goodputs for COAS and RIO

Two simulation experiments were performed to test the performance of COAS in comparison with RIO. Note that RIO here specifically refers to the RIO packet dropping algorithm, not the AS architecture proposed in [9]. A packet forwarding path is fixed using the path pinning

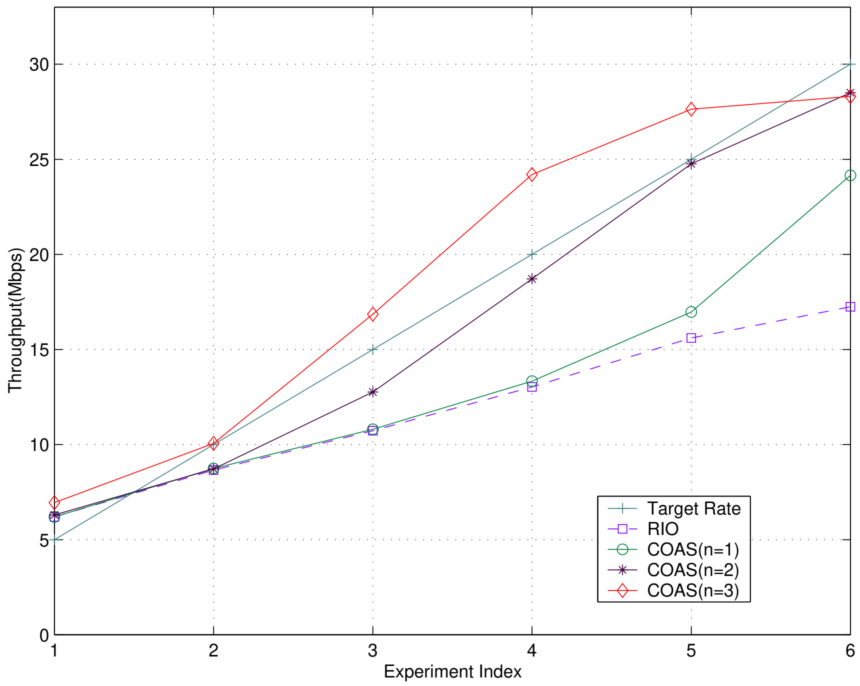

Figure 3: Aggregated throughputs/goodputs for COAS and RIO

mechanism proposed in Section 3.1.

The first experiment is based on the network setup in Fig. 2. We consider the case where there are 50 cross best-effort TCP flows between R1 and R2. With the other parameter settings identical to the ones in Section 2, we assume the 10 AS sessions have the same target rate and six different aggregated target rates are considered, i.e., 5, 10, 15, 20, 25, and 30 $Mbps$. The experiment is performed for COAS with $n = 1, 2, 3$ in (1) and also for RIO. The results for aggregated AS goodputs are shown in Fig. 3 and Table. 2. Also listed after slashes in Table. 2 are the goodputs for the aggregated cross best-effort traffic. From Fig. 3, one can see that COAS outperforms RIO at all three n values, with the best performance achieved at $n = 2$. One can also see that RIO offers acceptable performance only when the aggregated target rate for AS is small. From Table 2, one can clearly see that the cross best-effort traffic greatly degrades the performance of the AS traffic as its aggregated target rate increases. On the contrary, at $n = 2$, COAS successfully suppresses the cross traffic and keep the achieved goodput close to its target rate even with high bandwidth reservation. We observe that even at target rate $B_{as} = 30$ $Mbps$, the cross best-effort traffic can still achieve 1.47 $Mbps$ goodput, which suggests that COAS

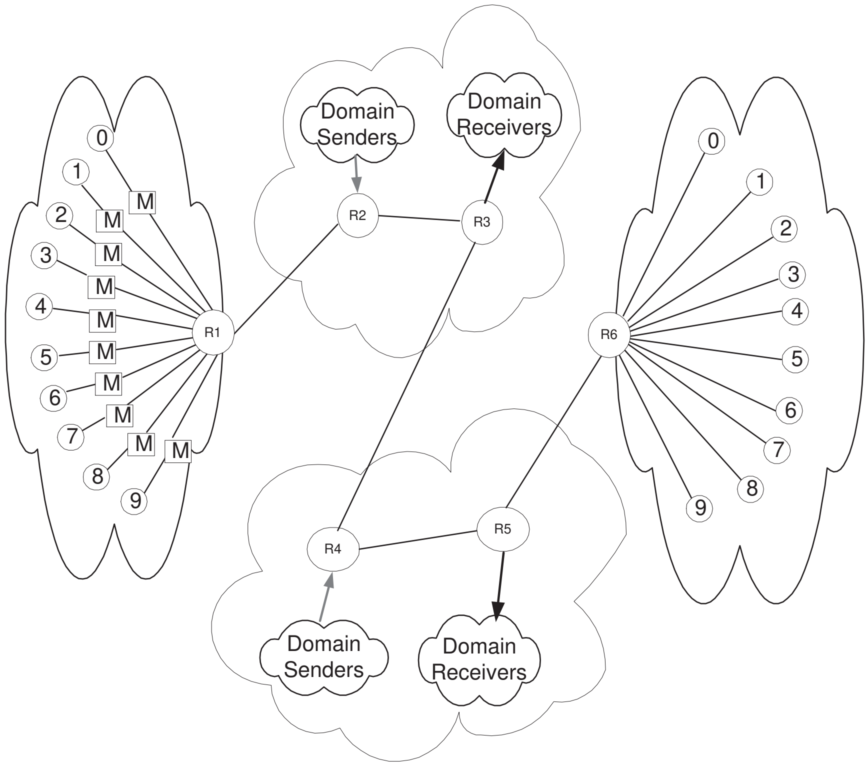

Figure 4: Network setup for experiment 2

not only provide high service assurance for the AS traffic but also a reasonable throughput performance for the best-effort traffic. Note that for all the experiments throughout the paper, a simple discrete control scheme of K_{be} is used, i.e., the largest integer which is no greater than the calculated K_{be} is used as the threshold for dropping best- effort packets. However, to avoid $K_{be} = 0$ due to the discrete control of K_{be}, one best- effort packet is allowed to be buffered with probability equal to K_{be} when the calculated K_{be} is smaller than 1.

To further examine the performance of COAS, we consider a network setup with one more domain in the data path as shown in Fig. 4. In this experiment, There are 10 AS TCP sessions between the hosts in domain 1 and domain 4, where domains are numbered increasingly from the left to the right. Their target rates and RTTs are listed in Table 3. The aggregated target rate is 30 $Mbps$. The link capacities between R1 and R2, R2 and R3, R3 and R4, and R5 and R6 are all 33 $Mbps$. The link capacity between R4 and R5 is 50 $Mbps$. There are 30 cross best-effort flows from R2 to R3 in domain 2 and 30 cross best-effort flows from R5 to R6 in domain 3. The simulation is performed for COAS with $n = 2$ and RIO. Both throughput and goodput are measured. This time we want to test the performance of each session and the results are listed in Table 3, with a slash separating the throughput from the goodput.

Connecion	RTT(ms)	Target Rate	RIO	COAS
0	20	5	3.36/3.32	4.66/4.48
1	20	1	0.89/0.82	2.72/2.62
2	40	5	3.17/3.06	4.37/4.33
3	40	1	0.62/0.57	2.01/1.94
4	50	5	2.85/2.81	3.98/3.89
5	50	1	0.77/0.69	1.96/1.83
6	70	5	2.39/2.37	3.75/3.68
7	70	1	0.78/0.72	1.51/1.43
8	100	5	2.83/2.80	3.01/2.99
9	100	1	0.49/0.46	1.26/1.23
Total		30	18.13/17.62	29.21/28.42

Table 3: Individual throughput/goodput for COAS and RIO

	In Dmain 2	In Domain 3
RIO	14.40/12.38	29.25/27.46
COAS	1.74 /1.51	19.30/17.48
Total Rate with RIO	32.53/30.00	47.38/45.08
Total Rate with COAS	30.95/29.93	48.51/45.90

Table 4: Aggregated throughputs/goodputs of best-effort traffic for COAS and RIO, and the total throughputs/goodputs for COAS in domain 2 and domain 3

As one can see, COAS outperforms RIO for all the AS sessions and again, it offers superior performance to RIO in terms of aggregated throughput guarantee. However, without isolations among AS sessions themselves, the fairness issue still exists when COAS is used.

Finally, to see the performance of the best-effort traffic, Table 4 lists the aggregated throughput/goodput for the cross best-effort traffic in both domains. As expected, RIO fails to suppress the cross best-effort traffic and leads to a bandwidth over utilization by the best-effort traffic. On the contrary, COAS gracefully suppresses the best-effort traffic in both domain 2 and domain 3, offering rather high goodputs at about 1.5 $Mbps$ and 17.5 $Mbps$ in the respective domains. Hence, COAS can locally suppress cross traffic, resulting in a near-optimal global resource utilization. In fact, for both experiments, every bottleneck link achieves a link utilization as high as 95 %.

5 Measurement-Based Queue Management Algorithm

In COAS, only the aggregated bandwidth reservation information is used to calculate K_{be}. In practice, however, an AS session may not always fully use its reserved bandwidth. When this happens, COAS can result in low resource utilization and cause unfairness issue between the AS traffic and best effort traffic. Low resource utilization due to AS under utilization of its reserved bandwidth may not be a problem when an AS customer is charged by the reserved bandwidth multiplied by the session duration, since in this case the cost is recovered, whether or not the session is active. However, if an AS customer is charged by usage or the number of bytes he/she sends, then COAS can result in a loss of revenue for an ISP whose network runs COAS. Hence, we need to explore measurement-based approaches. In what follows, we propose a simple measurement- based queue management algorithm for COAS, called the measurement-based COAS or simply MCOAS.

The only change in MCOAS with respect to COAS is that B_{as} in (**??**) is replaced by $min\{B_{as}, B_{ms}\}$, where B_{ms} is the measured aggregated bit rate for conformant AS traffic averaged in a given time window. With this change in (1), MCOAS allows the best-effort traffic to have a better chance to compete with the AS sessions for sharing the residual bandwidth left over by some other underutilized AS sessions. MCOAS also explores the multiplexing gain of the AS traffic.

First we demonstrate the deficiency of COAS in the presence of AS bandwidth under uti-

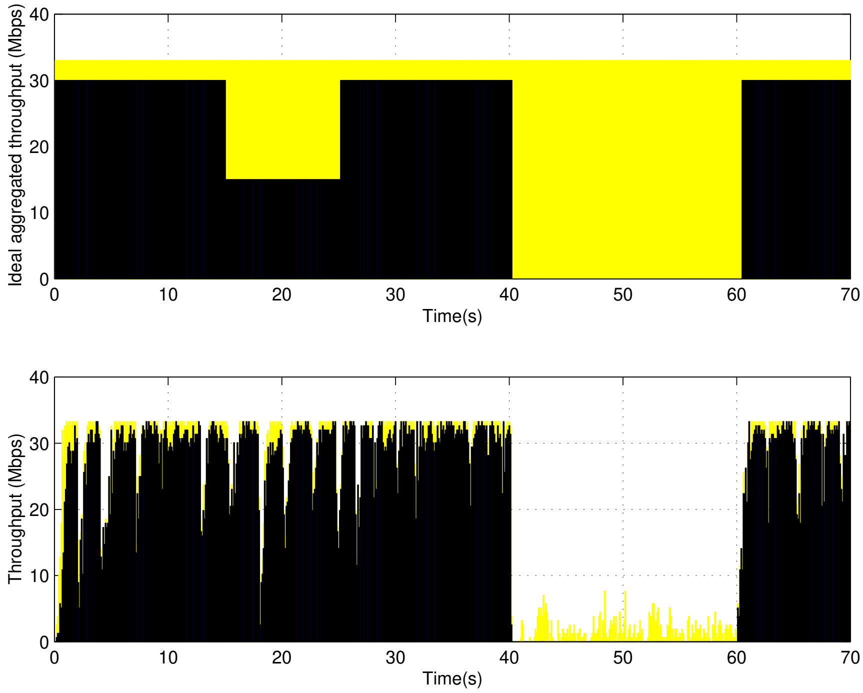

Figure 5: COAS performance with on-off AS sessions

lization. The network scenario in Fig. 2 is used for the experiment. The other network settings are the same as in experiment 1 except that the 10 AS TCP sessions are random on-off sources. The upper plot in Fig. 5 shows the ideal traffic throughput composition when the aggregated conformance AS traffic level varies due to the on-off behaviors of the AS sessions (the black area). The gray area is a mixture of the best-effort traffic and the non- conformant AS traffic. The link bandwidth is fully used for the ideal case. The lower plot in Fig. 2 is the actual throughput composition by running COAS. We notice that the best- effort traffic does not receive a fair share of the residual bandwidth with AS sessions in the time interval between 14 and 25 seconds, where half of the AS sessions are turned off. Note that in this plot, the black area represents the aggregated throughput for AS sessions and the gray area represents the aggregated throughput for the best-effort traffic. Almost all the residual bandwidth has been taken away by the other half of the active AS sessions. Even with all the AS sessions turned off from 40 to 60 seconds, the best-effort traffic cannot take over the available bandwidth, simple because the queue threshold K_{be} is calculated based on the aggregated reserved bandwidth B_{as} for AS sessions, which is unchanged when some or all of the AS sessions are in their off-periods.

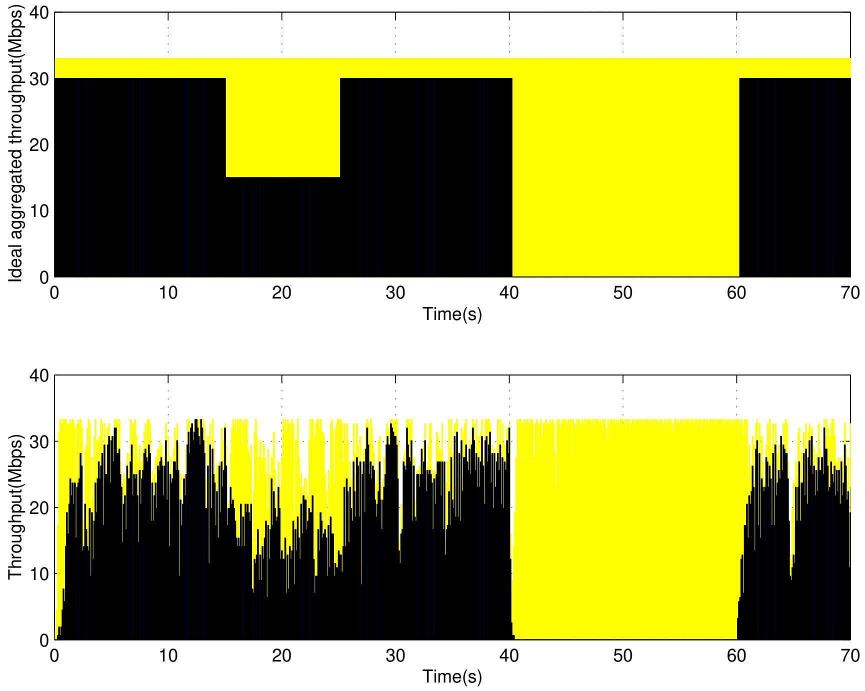

Figure 6: MCOAS performance with on-off AS sessions

To test the performance of MCOAS. We run MCOAS for the same simulation setting. The measurement window size for B_{ms} is fixed at 0.1 second and B_{ms} is updated every 0.1 second. Fig. 6 presents the simulation results. From the lower plot and with a reference to the upper plot, we observe that the best-effort traffic receives a rather fair share of the residual bandwidth with the AS sessions from 14 to 25 seconds and it takes over all the available bandwidth from 40 to 60 seconds. Note that the MCOAS performance is not quite sensitive to the selection of measurement window size for B_{ms} in the range between 0.01 seconds and a few seconds. The added complexity in MCOAS with respect to COAS is the measurement of the conformant AS traffic level or B_{ms} at each output port and periodic updates of K_{be} at each B_{ms} update interval, in addition to the updates upon each arrival and termination of an AS session.

6 Conclusion and Future Work

In this paper, a connection-oriented AS for TCP applications was proposed. The goal is to achieve end-to-end service assurance for TCP applications, a high network resource utilization, and high scalability, simultaneously. A path pinning mechanism was proposed to facilitate scalable flow state maintenance, connection setup, and resource reservation for the AS traffic. Packet marking and dropping policies proposed in [9] were also modified to provide better service isolations. Making use of the information about the current aggregated reserved bandwidth for the AS traffic, we were able to design two simple adaptive dropping-threshold algorithms, i.e., COAS and MCOAS, to locally suppress the best-effort traffic from overloading the AS traffic at each intermediate router. We were able to show that COAS and MCOAS can guarantee a rather high level of end-to-end service assurance for the aggregated AS TCP traffic, while still retaining a reasonably high throughput for the best-effort traffic.

The proposed scheme in this paper is the first step towards a complete solution for providing end-to-end AS for TCP applications. There are still many issues to be solved. One of the issues is that the present scheme does not provide service isolations among AS flows themselves and thus it does not solve the fairness issue. Yeom and Narasimha [11] proposed a scheme to solve this issue, however, at the expense of a much increased complexity at edge nodes. Hence, one of our future work is to address this issue in the context of the proposed scheme.

This paper is only concerned with the design of two CoSs, i.e., AS and the best- effort service, based on a single queue management. However, if more than two CoSs are to be supported, interesting issues arise as to how to allocate network resources among difference CoSs and how to do CoS-based routing. A salient feature of the DS architecture is the decoupling of traffic forwarding behavior from the service design. This feature allows service abstractions that enable future extension to incorporate CoS-based routing. The key to enable service abstractions is to logically segregate network resources, such as link bandwidth and buffer, for different CoSs, creating CoS-based virtual networks. Hence, part of our future research is to design dynamical resource allocation/sharing algorithms for different CoSs in DS. On the basis of these algorithms, the next step is to design CoS-based routing algorithms for DS.

References

[1] D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC2475, Dec., 1998, and the references therein.

[2] I. Yeom and A. Reddy, "Realizing throughput guarantees in a differentiated services network," *Proc. of IWQoS*, June 1999.

[3] I. Stoica and H. Zhang, "Providing Guaranteed Services Without Per Flow Management," *ACM SIGCOMM'99*, Boston, MA, Sept 1999.

[4] I. Stoica and H. Zhang, "LIRA: A Model for Service Differentiation in the Internet," *NOSS-DAV'98* .

[5] I. Stoica, H. Zhang, S. Shanker, R. Yavatkar, D. Stephens, A. Malis, Y. Bernet, Z. Wang, F. Baker, J. Wroclawski, and S. R. Wilder, "Per Hop Behaviors Based on Dynamic Packet States," Internet Draft, draft-stoica-diffserv-dps-00.txt, Feb. 1999.

[6] T. Li and Y. Rekhter, "Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," IETF Draft: draft-li-paste-01.txt, September 1998.

[7] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," IETF RFC 2702, September 1999.

[8] Y. Bernet, J. Binder, S. Blake, M. Carlson, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss, "A Framework for Differentiated Services," Internet Draft <draft-ietf-diffserv-framework-01.txt>, Oct., 1998, and the references therein.

[9] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, p. 362, Aug. 1998.

[10] I. Yeom and A. Reddy, "Impact of marking strategy on aggregated flows in a differentiated services network," *Proc. of IWQoS*, June 1999.

[11] I. Yeom and A. Reddy, "Marking of QoS Improvement," http://dropzone.tamu.edu/~ikjun/papers.html.

[12] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Trans. Networking, Vol. 1, No. 4, p. 397, Aug. 1993.