Proceedings of the 2003 International Conference on Information and Communication Technologies (ICT 2003), April 8-10, 2003
Edited by Dobri Atanassov Batovski, published by Assumption University, Thailand.

Fair Intelligent Congestion Control over DiffServ: A Resource Discovery and Control Scheme for DiffServ

Doan B. Hoang and Ming Li
Faculty of Information Technology
University of Technology Sydney,
Sydney, NSW 2007, Australia

Abstract

Current DiffServ architecture lacks mechanisms for network path discovery with specific service performance. It is recommended in RFC 2990 [1] that an admission control function be defined which can determine whether to admit a service differentiated flow along the nominated network path. This paper proposes Fair Intelligent Congestion Control over DiffServ (FICC-DS) for addressing this issue. The FICC-DS scheme involves a resource discovery (RD) loop, a fair share estimation algorithm, a congestion condition estimation algorithm, and a source admission control algorithm. Simulation results demonstrate that the FICC-DS achieves high network resource utilization and fairness performance than the standard DiffServ. In particular, the FICC-DS virtually eliminates the unfairness properties of TCP traffic class with respect to round trip time (RTT). It also allocates bandwidth fairly between TCP and UDP classes when they share a link over a DiffServ region.

1. Introduction

Today's Internet only provides best-effort service for all traffic. Traffic is processed as quickly as possible but there is no guarantee as to timeliness or actual delivery. The network makes no attempt to differentiate its service response between the traffic streams generated by concurrent users of the network. This means that the network is not able to guarantee the level of service required by an application that demands more stringent response in terms of delay, jitters, bandwidth, etc. Work on QoS-enabled IP networks has led to two distinct approaches: the Integrated Services architecture (IntServ) and the Differentiated Service architecture (DiffServ).

The goal of IntServ is to allow end-to-end QoS to be provided to applications [2]. The IntServ architecture needs an explicit setup mechanism to convey information to routers so that they can provide the requested services to flows. The resource requirements (computational processing and memory consumption) for running per-flow resource reservations on routers increase in direct proportion to the number of separate reservations that need to be accommodated. The use of per-flow state and per-flow processing is thus not cost effective, or even feasible across the high-speed core of a network.

On the other hand, DiffServ proposes a scalable service discrimination model without requiring any per-flow state management in the core network. DiffServ focuses primarily on aggregate flows and differentiates between service classes rather than providing absolute per flow QoS measures. DiffServ networks classify packets into one of a small number of aggregate flows or "classes", based on the DiffServ code point (DSCP) in the packet's IP header [3]. At each DiffServ router, packets are subject to "per-hop behavior" (PHB), which is invoked by the DSCP. DiffServ moves the complexity of providing QoS out of the core and into the edges of the network where it is feasible to maintain a restricted amount of per-flow states. While the aggregated behavior state of the Differentiated Services architecture offers excellent scaling properties, it does not guarantee end-to-end QoS for applications. The lack of end-to-end signaling facilities makes such an approach one that cannot operate in isolation within any environment [1]. In brief, RSVP/IntServ is not scalable and DiffServ does not guarantee end-to-end QoS for applications. Further more, DiffServ still has a bias against long RTT TCP classes and also cannot constrain misbehaving classes, e.g., UDP traffic, to its fair share.

As reported in [1], "the outcome of the considerations of these two approaches to QoS architecture within the network is that there appears to be no single comprehensive service environment that possesses both service accuracy and scaling properties." A framework for RSVP/IntServ operation over DiffServ networks has been proposed to provide both scalability and end-to-end QoS [4]. For this integration framework to be realized, mechanisms for conveying information about resource availability in a DiffServ network region to boundary routers and some form of signaling from the boundary to the client application must be developed. Currently there is no robust mechanism for network path discovery with specific service performance attributes. No existing mechanisms exist within either IntServ or DiffServ architectures to query the network for the potential to support a specific service profile [1].

This paper proposes a scheme called FICC-DS (Fair Intelligent Congestion Control over DiffServ) to address these issues. The scheme operates in the form of a resource discovery between edge routers of a DiffServ network region.

Our key technique is to estimate traffic class fairness by using Resource Discovery mechanism. To achieve fairness bandwidth allocation, we introduce fair intelligent resource discovery algorithm. In particular, we introduce an algorithm to estimate bandwidth fairshare of DiffServ classes and a resource management factor (RMF) to allow an admission control mechanism to adjust its admission rate according to the availability of resources within the DiffServ region.

The paper is organized as follows. Section 2 briefly describes some related work. Section 3 presents our resource discovery and control scheme. Section 4 describes simulation results. Section 5 & 6 discuses implications of the results and concludes with suggestions for future work.

2. Related work

Over the last two years, several research efforts have been made to address the problem of finding robust mechanisms for network path discovery with specific service performance.

De Meer et al. [5] provided an analysis of existing IP quality of service solutions and the implied signaling issues. It is pointed out that an improvement to the QoS DiffServ architecture could be achieved by providing congestion signaling from within a DiffServ domain to the boundary between the two administrative domains. It is also believed that feedback information and signaling is needed in the next generation of a DiffServ architecture that delivers its specified classes of service by a combination of resource provisioning and cooperation with the subscribers. Our proposal addresses these issues.

Jeong et al. [6] proposed a set of router-based QoS mechanisms including queue policy, resource reservation and metering using the enforcement of traffic profile. The proposed queue policy is to ensure that UDP flows get required bandwidth and TCP flows are protected from unresponsive UDP flows. The proposal only considered simple buffer partitions for allocating bandwidth. Our project addresses directly the resource discovery mechanisms.

Gerla et al. [7] considered bandwidth feedback control of TCP and real time sources in the Internet. Our feedback loop is somewhat similar in spirit, however, our feedback framework is more explicit in that it allows allocation of bandwidth fairly among classes, and among aggregates within a traffic class, rather than controlling TCP flows.

In Endpoint Admission Control (EAC) [8], the explicit decisions whether to accept or refuse a connection request were taken by edge devices, rather than by devices within the network. The driving idea of EAC schemes is to convey the congestion status of network nodes to the end-points. The idea is sound but may not be adequate to control the connection's QoS. Our approach employs feedback information explicitly from both the endpoints and the core routers.

Bianchi [9] suggested a solution in which a DiffServ is always associated with probing channel of the same class but at a lower priority. If packets on the probing channel make it to the egress router of the DiffServ region and back to ingress router, the region is deemed not congested. The idea is to push traffic control to the edge and basing the connection requests acceptance/refusal on packet loss detection. A drawback of the approach is that the feedback may not be adequate to allow an edge device to control the QoS measure tightly. The study also raises the case for defining new "paired" PHB in DiffServ architecture.

In the next section we present our scheme necessitates a feedback loop for resource discovery and mechanisms for sharing bandwidth and avoiding congestion.

3. Resource discovery and control mechanisms

In this paper we focus on the DiffServ region. The basic idea of the control and discovery mechanism is depicted in a schematic diagram in the middle of Figure 1.

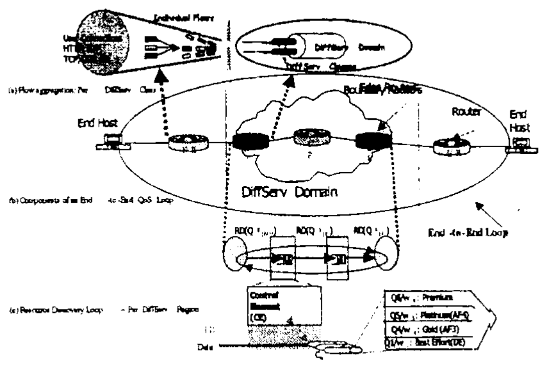

Figure 1. A reference IntServ/DiffServ and feedback control loop between edge routers of the DiffServ region.

For each traffic class, the source edge router (SER) introduces a Resource Discovery (RD) packet at a rate proportional to the traffic rate for that traffic class destined for a destination edge router (DER).

The RD packets contain a vector of QoS parameters as well as traffic parameters for the SER-DER pair and is marked as highest priority to avoid dropping. It should be noted that we plan to implement the feedback loop and its associated algorithms as a control service running on the control plane of a programmable router, hence the operation of the loop does not interfere with the forwarding plane of the router.

Openet [10] is such a programmable networking platform that allows the router's control plane to inspect its available resources and to introduce network services dynamically.

Followed subsections will describe the edge and core router behaviors, class based available network resource algorithms, congestion condition calculation, admission control mechanism and overhead in details.

2.1. Edge router behavior

As far as the FICC-DS concerned, the edge router is responsible for classifying packet into class, initiating and maintaining the RD control loop, collecting the feedback information, and exercising admission control algorithm in DiffServ region. The key function for source edge router is to generate the special resource discovery packet, RD, which carries the QoS parameters the network can support so far. The RD packet is generated proportionally to the class traffic rate. The more packets arriving in source edge router, the more RD packets associated with same class will be generated. In this implementation, we set the minimum RD rate is one RD packet in average class round trip time. The RD packet length is 50 bytes and we set counter number is 50. If the data packet size is 500 bytes [11], that is, generate a RD packet every 50 packets. The source edge router and RD packet generation algorithms are described in Figure 2.

Since the exact computation of the class arrival rate is hardly feasible, a class arrival rate estimate $\hat{AR}_i(t)$ is updated upon the reception of every packet using exponential averaging formula as in CSFQ [12]. Using an exponential weight gives more reliable estimation for bursty traffic, even when the packet inter-arrival time has significant variance. If we indicate the arrival time of the k-th packet of class i as T_i^K and its length as $l_i^K(t)$, the new estimate of $\hat{AR}_i(t)$ can be computed as follows:

$$AR_i^{new}(t) = (1 - e^{-T_i^K/K})\frac{l_i^K}{T_i^K} + e^{-T_i^K/K} * AR_i^{old}$$

T_i^K represents the k-th sample of the interarrival time of class i, i.e., $T_i^K = t_i^k - t_i^{(k-1)}$ and K is a constant. RD packet will carry the arrival rate for class and class weight.

```
Source edge router behavior
Initialization
When a new packet arrives,
Classify the packet into class;
If it's time to generate RD packet
        Estimate class arrival rate, class_arrival_rate. Use (1)
        Invoke RD agent to generate RD packet
        Put QoS parameters into RD packet, e.g., class
        arrival rate, class weight, ER_unit
        Send the RD packet to network
Else
        Release the packet according to policy of admission
        control
When backward RD packet arrives,
Update the policy of admission control plane, e.g., leaky bucket rate

Algorithm for RD packet generation
Initial configuration, for ith class
    α=50bytes/RTT(i)
    RD_rate_init(i)=α
    β=500
When a new ith class packet arrives,
        RD_rate(i)=class_arrival_rate(i)/β
        If RD_rate(i) < RD_rate_init(i)
        Then
          RD_rate(i)=RD_rate_init(i)
```

Figure 2. Algorithms performed by the source edge router

2.2. Core router behavior

Each core router along the path consults its QoS state for that class and modifies the parameters of the RD packets it can support accordingly then forwards the RD packets to the next router till to the destination edge router.

To calculate available network resource, we introduce a class-unit concept. The basic idea behind class-unit concept is the fact that class traffic with different target rate and round trip time can be subdivided into smaller unit, which can be manipulated effectively. The smaller the class-unit is, the more sensitive FICC-DS responses to the network variation. In this FICC-DS we choose 1 Mbps target rate as class-unit.

All intermediate core routers perform exactly the same algorithms to calculate mean class-unit fair share rate, available buffer and congestion condition for the class traffic.

In FICC-DS, we aim to operate the routers around a target operating point. The target operating point adopted in this scheme is a pre-set Buffer Utilization Ratio (BUR). The idea is to prevent the output line from being idle by always keeping some amount of data in the buffer; the BUR is at a level that does not introduce excessive packet delay [13].

In estimating the congestion condition of the network, we employed the Fair Intelligent Congestion Control algorithm [13] to compute the

Resource Management Factor, RMF. We introduce Target Point as the percentage level of an output queue buffer at which a router should operate. This level is then translated to a queue level Q0. If the smoothed queue occupancy, queue length is greater than Q0, the RMF is reduced proportionally to the amount of buffer available (i.e., the degree of congestion is proportional to the excess package occupancy above the target Q0). If queue length is less than Q0, RMF performs as an amplifier where the amplification factor is proportional to the degree at which he buffer is underused below Q0). In estimating the fairshare for each DS class (per output queue), the algorithm tracks the usage of all classes by exponentially averaging, but it also imposes a limit when the buffer level rises above the target level. The algorithm is described in Figure 3.

```
When receive RD packet, update AvgRate_unit and resource management factor, (RMF) and calculate explicit rate, ER

Algorithm for Average Fair Share Rate
Initialization
    AvgRate_unit=100Mbps; Initial value for Average Share Rate
    BUR=0.6; buffer utilization ratio
    Q0=BUR*Qsize; target operating point
    β=0.3; Rate control parameter
    α=0.15; Queue Control Parameter
    Weight(i): I-th class weight

AR_unit(i)=AR(i)/Weight(i); Arrival Rate from RD packet
If (QueueLength)< Q0
        AvgRate_unit=(1-β)*AvgRate_unit+β*AR_unit(i)
Else if AR_unit<AvgRate_unit
        AvgRate_unit=(1-β)*AvgRate_unit+β*AR_unit(i)

Algorithm for calculating RMF
if (QueueLength >Q0)
        RMF_unit=(Qsize-QueueLength)/(Qsize-Q0)
else
        RMF_unit= α*(Q0-QueueLength)/Q0 +1

Algorithm for calculating explicit rate, ER
ER_unit=AvgRate_unit*RMF_unit
If (ER_unit < ER value in RD packet)
    Update ER value in RD packet with ER_unit
```

Figure 3. Algorithms performed by the core router

2.3. Admission Control Mechanism

In this initial study, we employed a simple strategy for admission control. The TokenRate(i) associated with a DS class is adjusted depending on the amount of available bandwidth and the degree of congestion. The control plane module is attached to the source edge router using a leaky bucket (σ, ρ) shaper to conform to a Linearly Bounded Arrival Process [14] and the shaper parameters will be updated after receiving new backward RD packet.

$$\int^{t+\tau} \lambda_i(T)dT \le r_i\tau + \sigma \qquad (1)$$

where r_i is the shaper rate limit and σ is the maximum burst size. Currently, we set r_i as explicit rate, ER.

2.4. Overhead discussion

Since the cost of internal computation is negligible compared to the cost of a transmission, we only need to study the overhead caused by extra packet, RD packet. We express the overhead calculation O as in (2).

$$O = \frac{\sum_{i=1}^{n} RD_i}{\sum_{i=1}^{n} RD + Data} \qquad (2)$$

Where n is the class number. We use this ratio to determine how much extra overhead the RD packets caused. In our proposal, the source edge router generates the RD packet and the RD generation rate is proportional to the data transmission rate. From [11], the Internet IP data packet length mainly on 552 and 576 bytes. The RD packet size is 50 bytes, and from Figure 2, RD generation algorithm, we know that the source edge router generates a RD packet every 50 data packets, then it needs bandwidth no more than 0.19%.

3. Simulation results

We use the *ns* [15] network simulator to evaluate the proposed scheme. *NS* is a discrete event simulator for network research. It provides support for TCP, router queuing mechanisms, and various topologies. We have developed several new modules – source boundary, destination boundary and core routers, and the FICC-DS scheme – and incorporated them into *ns*.

The configuration (Figure 4) employed in our simulation is the same one that was used in Fang and Clark [16]. There are 10 sources sending data to 10 distinct destinations through a "DiffServ" region. Each source can be considered as an individual flow or an aggregated flow (or DiffServ class). A class may be an aggregation of TCP and/or UDP flows. The TCP version used in our simulations is TCP Reno, which includes fast retransmission and fast recovery. The packet size is 512 bytes. The receiver's advertised window is configured large enough so that it does not limit the TCP sender's window. We use a constant bit rate, CBR, source to model non-responsive sources (UDP) since a CBR source does not have a congestion control mechanism. The sending rate of CBR is 10Mbps and the achieved throughput of the CBR connection is calculated at the receiver.

We compare the performance of our scheme, FICC-DS, and the DiffServ scheme (the Time Sliding Window (TSW) DiffServ scheme by Fang and Peterson [17]. In particular we want to investigate the throughput performance with respect to RTT of various classes and with different classes of traffic. The Time Sliding Window (TSW) tagging algorithm runs on the boundary routers that tag packets as IN or OUT according to specific service profiles. It has two components: a rate estimator that estimates the sending rate over a certain period of time, and a tagger that tags packets based on the rate reported by the rate estimator. If the source is sending below the target rate, the tagger will tag all packets as IN; if the source is sending above the target rate, the tagger will tag those packets in excess of the target rate as OUT. TSW maintains three variables: Win_length, which is measured in units of time, AvgRate, the rate estimate upon each packet arrival, and T_front, the time of the last packet arrival.

The simulation runs for 200 seconds.

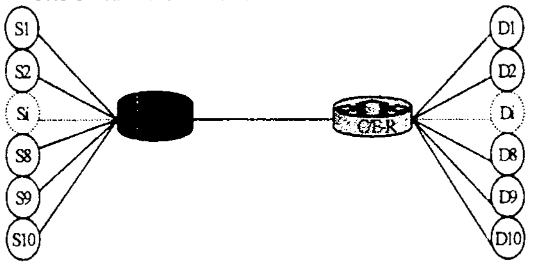

Figure 4. Topology used in simulation

We will evaluate our proposal with different situations, TCP traffic with different Round Trip Time and the same target rate, UDP traffic with the same target rate, and TCP mixed with UDP traffic.

3.1. TCP Traffic with different RTT and same target rate

In this scenario, all of 10 TCP classes have the same target rate but different RTTs. Normal DiffServ scheme do not solve the well-known unfairness of TCP protocol. Those classes with shortest RTT (1 and 2), 20ms, claim the most bandwidth and those with longest RTT (3 an 4), 100ms, claim the least.

Figure 5 compares the throughput performance of our FICC-DS and DiffServ. It is seen that FICC-DS provides much better fairness to all TCP classes irrespective of their Round Trip Time. This results comes from the fact that FICC-DS estimate a class fair share based on its usage rather than the time it takes to update the sending window.

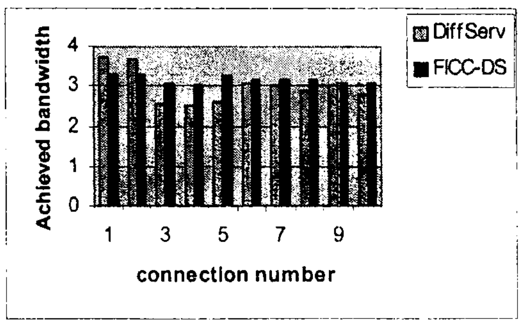

Figure 5. FICC-DS eliminates the effect of RTTs

3.2. TCP mixed UDP with same target rate

In this scenario, there are 10 classes traffic all with the same RTT and same target rate, six of them are TCP traffic and two are UDP traffic (5 and 10). Clearly, without being restrained by the TCP congestion control algorithms, the two UDP classes claim excessive amount of bandwidth.

Figure 6 demonstrates that FICC-DS is able to keep the two UDP traffic classes in check by allocating them their proper bandwidth shares. All classes received very much the same bandwidth share as they all have the same target rate.

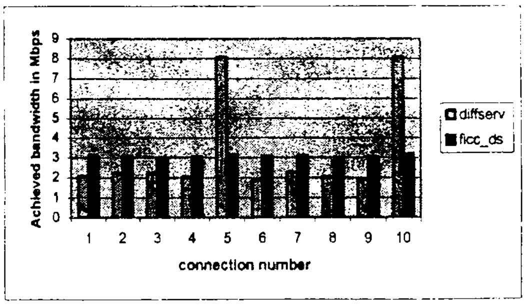

Figure 6. TCP/UDP classes with the same target rate and RTT

3.3. UDP traffic with same RTT and same target rate

Figure 7 shows that FICC-DS again allocates bandwidth fairly among all UDP classes with same RTT and same target rate. The DiffServ and FICC-DS achieve the similar fairness.

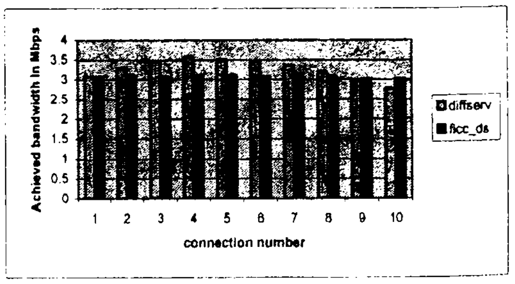

Figure 7. Throughput for all UDP classes

4. Discussion

The scheme proposed in this paper has the same TCP adaptation spirit, but it is applied to DiffServ classes (rather that TCP flows) and it uses explicit feedback information rather than implicit congestion notification.

One issue with the scheme is when to generate an RD packet. We assume that RD packet is generated as long as there exists a DS class between a Source-Destination Boundary Router pair. All RD packets after the first one can be considered as periodic refreshed packets. In this implementation, we estimate the fairshare of all DS classes within the core routers and that is the reason why we generate RD packets at a rate proportional its class traffic rate. Alternatively, we can leave the task of fairshare estimation to the edge routers and then we can reduce the overhead by generating RD packets less frequently. An adaptive scheme may be appropriate to determine an optimal refreshing frequency.

With our approach the RD parameters discovered are not global since the loop only has a partial view of the network. Global information can be obtained with QoS-based routing where QoS parameters and cost functions are flooded to all routers in the networks frequently. However, QoS routing suffers several serious problems. Firstly, it is an NP-complete problem when multiple cost metrics are employed. Secondly, QoS routing suffers instability problem: the cost metric changes every time a connection is admitted to the network and impossible for each router to maintain a consistent global database required for an QoS algorithm. Thirdly, it is costly in terms of bandwidth used for flooding QoS information to all routers. With FICC-DS, a core router only has to keep state information of DiffServ classes (corresponding to at most 32 DSCP code points), and several variables per router output queue. We believe this is acceptable for achieving tighter service responses over a DiffServ region.

5. Conclusion

The paper proposes FICC-DS scheme to address issues raised in RFC 2990 [1], namely, lacking a standardized admission control scheme, lacking mechanisms for conveying information about resource availability in a DiffServ network region to boundary/edge routers, and does not intrinsically solve the problem of controlling congestion.

Simulation results demonstrate that FICC-DS achieves better throughput and fairness performance than standard DiffServ. Our next step is to investigate the implementation of FICC-DS over DiffServ-routers' control plane.

References

[1] G. Huston, "Next steps for the QoS architecture," in *IETF RFC2990*, 2000.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet architecture: an overview," in *IETF RFC1633*, 1994.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," in *IETF RFC2475*, 1998.

[4] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, E. Felstaine, and J. Wroclawski, "Integrated Service over DiffServ networks," in *IETF RFC2998*, 2000.

[5] H. De Meer, P. O'Hanlon, G. Feher, N. Blefari-Melazzi, H. Tschofenig, G. Karagiannis, D. Partain, V. Rexhepi, and L. Westberg, "Analysis of Existing QoS Solutions," in *IETF Internet Draft draft-demeer-nsis-analysis-02.txt*, 2002.

[6] S. H. Jeong, H. Owen, J. Copeland, and J. Sokol, "QoS support for UDP/TCP based networks," *Computer Communications*, vol. 24, pp. 66-77, 2001.

[7] M. Gerla, W. Weng, and R. L. Cigno, "Bandwidth feedback control of TCP and real time sources in the Internet,," Proceedings of IEEE GLOBECOM, San Francisco, CA, USA, 2000.

[8] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance," ACM SIGCOMM, Stockholm, Sweden, 2000.

[9] G. Bianchi and N. Blefari-Melazzi, "Per Flow Admission Control over AF PHB Classes," in *IETF Internet Draft draft-bianchi-blefari-admcontr-over-af-phb-00.txt*, 2000.

[10] T. Lavian, P. Wang, F. Travostino, S. Subramanian, D. B. Hoang, V. Sethaput, and D. Culler, "Enabling Active Flow Manipulation In Silicon-based Network Forwarding Engines," *Journal of Communications and Networks*, pp. 78-87, 2001.

[11] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet traffic patterns and characteristics," *IEEE Network*, 1997.

[12] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks," Proceedings of the ACM SIGCOMM'98, Vancouver, Canada, 1998.

[13] D. B. Hoang, X. Yu, and D. D. Feng, "Fair Intelligent bandwidth allocation for rate adaptive video traffic," *Computer Communications special issue on support of multimedia communications over the Internet*, vol. 23, pp. 1425-1436, 2000.

[14] R. L. Cruz, "A calculus for network delay and a note on topologies of interconnection networks," University of Illinois, 1987.

[15] LBL, "NS-2, Network simulatior," 2 ed: LBL, 2000.

[16] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 362-373, 1998.

[17] W. Fang and L. Peterson, "TCP mechanisms for Diff-Serv architecture," Princeton University Technical Report 605-99, 1999 1999.

the achieved throughput of the CBR connection is calculated at the receiver.

We compare the performance of our scheme, FICC-DS, and the DiffServ scheme (the Time Sliding Window (TSW) DiffServ scheme by Fang and Peterson [17]. In particular we want to investigate the throughput performance with respect to RTT of various classes and with different classes of traffic. The Time Sliding Window (TSW) tagging algorithm runs on the boundary routers that tag packets as IN or OUT according to specific service profiles. It has two components: a rate estimator that estimates the sending rate over a certain period of time, and a tagger that tags packets based on the rate reported by the rate estimator. If the source is sending below the target rate, the tagger will tag all packets as IN; if the source is sending above the target rate, the tagger will tag those packets in excess of the target rate as OUT. TSW maintains three variables: Win_length, which is measured in units of time, AvgRate, the rate estimate upon each packet arrival, and T_front, the time of the last packet arrival.

The simulation runs for 200 seconds.

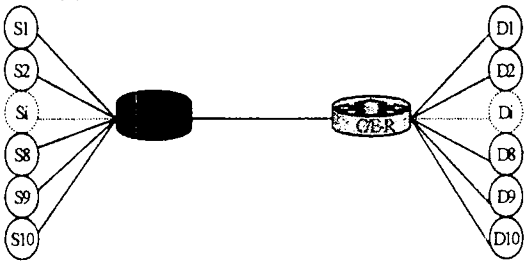

Figure 4. Topology used in simulation

We will evaluate our proposal with different situations, TCP traffic with different Round Trip Time and the same target rate, UDP traffic with the same target rate, and TCP mixed with UDP traffic.

3.1. TCP Traffic with different RTT and same target rate

In this scenario, all of 10 TCP classes have the same target rate but different RTTs. Normal DiffServ scheme do not solve the well-known unfairness of TCP protocol. Those classes with shortest RTT (1 and 2), 20ms, claim the most bandwidth and those with longest RTT (3 an 4), 100ms, claim the least.

Figure 5 compares the throughput performance of our FICC-DS and DiffServ. It is seen that FICC-DS provides much better fairness to all TCP classes irrespective of their Round Trip Time. This results comes from the fact that FICC-DS estimate a class fair share based on its usage rather than the time it takes to update the sending window.

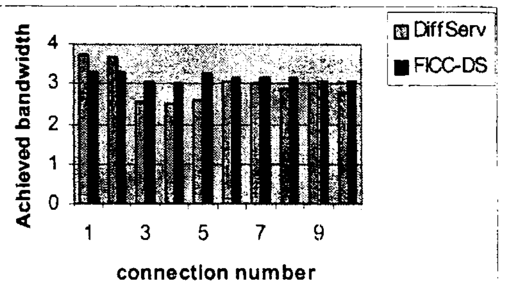

Figure 5. FICC-DS eliminates the effect of RTTs

3.2. TCP mixed UDP with same target rate

In this scenario, there are 10 classes traffic all with the same RTT and same target rate, six of them are TCP traffic and two are UDP traffic (5 and 10). Clearly, without being restrained by the TCP congestion control algorithms, the two UDP classes claim excessive amount of bandwidth.

Figure 6 demonstrates that FICC-DS is able to keep the two UDP traffic classes in check by allocating them their proper bandwidth shares. All classes received very much the same bandwidth share as they all have the same target rate.

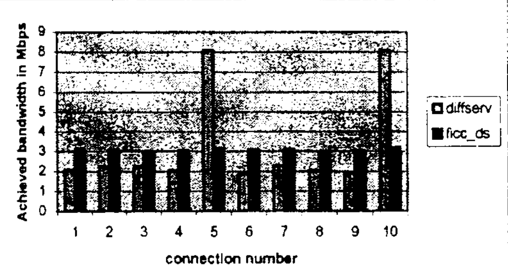

Figure 6. TCP/UDP classes with the same target rate and RTT

3.3. UDP traffic with same RTT and same target rate

Figure 7 shows that FICC-DS again allocates bandwidth fairly among all UDP classes with same RTT and same target rate. The DiffServ and FICC-DS achieve the similar fairness.

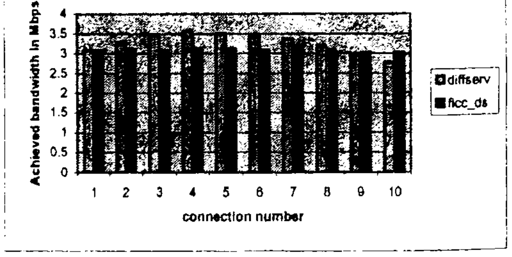

Figure 7. Throughput for all UDP classes

4. Discussion

The scheme proposed in this paper has the same TCP adaptation spirit, but it is applied to DiffServ classes (rather that TCP flows) and it uses explicit feedback information rather than implicit congestion notification.

One issue with the scheme is when to generate an RD packet. We assume that RD packet is generated as long as there exists a DS class between a Source-Destination Boundary Router pair. All RD packets after the first one can be considered as periodic refreshed packets. In this implementation, we estimate the fairshare of all DS classes within the core routers and that is the reason why we generate RD packets at a rate proportional its class traffic rate. Alternatively, we can leave the task of fairshare estimation to the edge routers and then we can reduce the overhead by generating RD packets less frequently. An adaptive scheme may be appropriate to determine an optimal refreshing frequency.

With our approach the RD parameters discovered are not global since the loop only has a partial view of the network. Global information can be obtained with QoS-based routing where QoS parameters and cost functions are flooded to all routers in the networks frequently. However, QoS routing suffers several serious problems. Firstly, it is an NP-complete problem when multiple cost metrics are employed. Secondly, QoS routing suffers instability problem: the cost metric changes every time a connection is admitted to the network and impossible for each router to maintain a consistent global database required for an QoS algorithm. Thirdly, it is costly in terms of bandwidth used for flooding QoS information to all routers. With FICC-DS, a core router only has to keep state information of DiffServ classes (corresponding to at most 32 DSCP code points), and several variables per router output queue. We believe this is acceptable for achieving tighter service responses over a DiffServ region.

5. Conclusion

The paper proposes FICC-DS scheme to address issues raised in RFC 2990 [1], namely, lacking a standardized admission control scheme, lacking mechanisms for conveying information about resource availability in a DiffServ network region to boundary/edge routers, and does not intrinsically solve the problem of controlling congestion.

Simulation results demonstrate that FICC-DS achieves better throughput and fairness performance than standard DiffServ. Our next step is to investigate the implementation of FICC-DS over DiffServ-routers' control plane.

References

[1] G. Huston, "Next steps for the QoS architecture," in *IETF RFC2990*, 2000.
[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet architecture: an overview," in *IETF RFC1633*, 1994.
[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," in *IETF RFC2475*, 1998.
[4] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, E. Felstaine, and J. Wroclawski, "Integrated Service over DiffServ networks," in *IETF RFC2998*, 2000.
[5] H. De Meer, P. O'Hanlon, G. Feher, N. Blefari-Melazzi, H. Tschofenig, G. Karagiannis, D. Partain, V. Rexhepi, and L. Westberg, "Analysis of Existing QoS Solutions," in *IETF Internet Draft draft-demeer-nsis-analysis-02.txt*, 2002.
[6] S. H. Jeong, H. Owen, J. Copeland, and J. Sokol, "QoS support for UDP/TCP based networks," *Computer Communications*, vol. 24, pp. 66-77, 2001.
[7] M. Gerla, W. Weng, and R. L. Cigno, "Bandwidth feedback control of TCP and real time sources in the Internet,," Proceedings of IEEE GLOBECOM, San Francisco, CA, USA, 2000.
[8] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance," ACM SIGCOMM, Stockholm, Sweden, 2000.
[9] G. Bianchi and N. Blefari-Melazzi, "Per Flow Admission Control over AF PHB Classes," in *IETF Internet Draft draft-bianchi-blefari-admcontr-over-af-phb-00.txt*, 2000.
[10] T. Lavian, P. Wang, F. Travostino, S. Subramanian, D. B. Hoang, V. Sethaput, and D. Culler, "Enabling Active Flow Manipulation In Silicon-based Network Forwarding Engines," *Journal of Communications and Networks*, pp. 78-87, 2001.
[11] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet traffic patterns and characteristics," *IEEE Network*, 1997.
[12] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks," Proceedings of the ACM SIGCOMM'98, Vancouver, Canada, 1998.
[13] D. B. Hoang, X. Yu, and D. D. Feng, "Fair Intelligent bandwidth allocation for rate adaptive video traffic," *Computer Communications special issue on support of multimedia communications over the Internet*, vol. 23, pp. 1425-1436, 2000.
[14] R. L. Cruz, "A calculus for network delay and a note on topologies of interconnection networks," University of Illinois, 1987.
[15] LBL, "NS-2, Network simulatior," 2 ed: LBL, 2000.
[16] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 362-373, 1998.
[17] W. Fang and L. Peterson, "TCP mechanisms for Diff-Serv architecture," Princeton University Technical Report 605-99, 1999 1999.