

# Fair Intelligent Congestion Control

for both congestion control and bandwidth fairshare

Doan B. Hoang

Advanced Research in Networking Laboratory  
FIT, University of Technology, Sydney  
Broadway, Australia  
dhoang@it.uts.edu.au

**Abstract—** Most QoS issues are originated from two fundamental problems: network congestion and unfair network resource allocation. This paper introduces a new congestion control scheme, the Fair Intelligent Congestion Control (FICC) to handle these problems. The key design features of FICC include fairness, scalability, and efficiency. Simulation results demonstrate that FICC handles congestion effectively in that it prevents excessive delays at very high throughput. FICC allocates bandwidth fairly among competitive sources. FICC is scalable in that it only requires simple per-queue rather than per-aggregate accounting.

**Keywords-** congestion control, explicit feedback, bandwidth allocation, rate control, DiffServ.

## I. INTRODUCTION

Network QoS requirements for an application are defined in terms of the allocated resources (bandwidth), latency (delay and jitter), loss (bits, packets, or other units), and availability. A network that supports QoS is a network that presents users with differentiated levels of QoS responses and allows users to make choices according to their requirements.

The current Internet is, however, unable to support emerging applications such as premium quality audio or video communications, stock trading transactions and interactive games, because of its undifferentiated and unpredictable service responses.

Almost all QoS issues are originated from two fundamental problems: network congestion and unfair network resource allocation.

When a network becomes congested, it is unable to provide adequate resources required by the application. The causes of congestion may include inadequate network dimensioning, lack of admission control, lack of congestion control, and unpredictable converging traffic patterns.

Even in the absence of congestion conditions, a network may not be able to allocate bandwidth fairly among different aggregates due to the varied attributes of IP traffic such as packet size, round trip time, and traffic type (flow-controlled TCP, uncontrolled UDP, etc.).

To address the QoS issues, the Integrated Services (IntServ) [1] and Differentiated Services (DiffServ) [2] architectures were proposed by the Internet Engineering Task Force (IETF)

some ten years ago. The goal of IntServ is to allow end-to-end QoS to be provided to (per-flow) applications. On the other hand, DiffServ proposes a scalable service discrimination model without requiring per-flow state management in the core network. It has become clear that neither IntServ nor DiffServ provides the answer. IntServ can support end-to-end QoS but is not scalable, and DiffServ is scalable but cannot guarantee end-to-end QoS for applications. However, it is generally believed that DiffServ is more likely to be implemented in the Internet core because of its simplicity and scalability.

Quoting RFC 2990 [3], “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.” Attempts to employ IntServ at the user access ends and DiffServ in the network core have not been realized due to additional complexity in providing the support for IntServ mechanisms over the DiffServ domain.

For an end-to-end framework to be realized, mechanisms for conveying information about resource availability in a DiffServ network region (from core routers) to edge routers and some form of signalling from the edge to the client application *must be developed*. It is also recommended in RFC 2990 [3] that an admission control function be defined which can determine whether to admit a service differentiated flow along the nominated network path..

We have recently proposed an end-to-end QoS architecture called End-Diff [4], that takes into account both interdomain and intradomain problems as well as the RFC 2990’s recommendations. However, this paper only addresses the congestion control and the unfair network resource sharing problems.

FICC is introduced as a simple but effective rate-based congestion control scheme that addresses both fair bandwidth sharing and congestion problems encountered in current QoS architecture. FICC is scalable since it is only applied on each interface of a router. FICC intelligently predicts per-queue fair share for all traffic aggregates. FICC uses feedback control to keep the router operating at a desirable operating point at all times. FICC allows overselling bandwidth when the network is not congested to make efficient use of the network resources. FICC estimates the degree of congestion as a function of the fraction of the buffer that is occupied, which is a more appropriate measure of congestion than measures based on

fixed occupancy thresholds that do not depend on the total buffer size.

In this paper we describe the design of FICC, present its performance within the context of an enhanced DiffServ architecture. Simulation results demonstrate that FICC is fair and effective. FICC is fair in that it allocates available bandwidth fairly among traffic classes that can use the allocation. FICC is simple and robust in that it requires very few parameters and is not sensitive to small variations of these parameters. FICC is scalable in that it does not require per-flow computation and accounting. It is effective in that it minimizes average packet delays and delay variations while maintaining high throughput.

The rest of the paper is organized as follows. Section 2 describes an enhanced architecture for which FICC operates. Section 3 details the design and development of the Fair Intelligent Congestion Control (FICC). Section 4 presents the results of FICC simulations over the enhanced DiffServ. Section 5 concludes the paper by summarizing the contributions and suggesting directions for further research.

## II. FICC-DIFFSERV

Since our FICC scheme is designed to operate over a scalable QoS architecture, and specifically the DiffServ. The original DiffServ architecture, however, lacks a standardized admission control scheme, and does not intrinsically solve the problem of controlling congestion in the Internet.

In [5], we proposed FICC-DiffServ that has a number of new and distinctive features to overcome the problems of the original DiffServ: no sophisticated PHB scheduling regimes are required at routers (FIFO is sufficient), a resource discovery loop (RD loop) for discovering available resources along a path, and a per-class (or per aggregate) admission control to the domain can handle admitted traffic according to their QoS requirements (Figure 1).

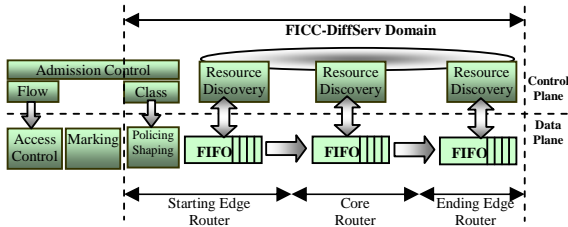


Figure 1. FICC-DiffServ Architecture

Inside the FICC DiffServ domain, two sets of components are implemented: basic DiffServ components and QoS components. The two basic DiffServ components, *policing, shaping and marking*, inherit partly from the original DiffServ model. The three QoS components – the *traffic monitor*, the *admission control agent* and the *resource discovery agent* – are located in the **control plane** of the existing IP network infrastructure to support QoS capabilities of the framework. The first two are implemented only in the ingress router, whereas, the third involves in all routers along the network path.

The *traffic monitoring* agent (only at the ingress) observes incoming traffic of each class inside a DiffServ domain to measure the traffic arrival rate.

The admission control agent implements *per-class admission control* functions to prevent congestion and to allocate resources fairly among traffic classes within the domain. It uses available resource information updated from resource discovery agent to calculate an Explicit Rate (ER) for each class. Per-flow admission control guarantees the fairness among individual flows within the same class. It is derived from the per-class (or per-aggregate) admission control using the same FICC principle; however, it is not covered in this paper.

*Resource Discovery (RD)* agent at an ingress router uses the RD packets between itself and an egress router to probe the available resources inside the augmented FICC-DiffServ domain, and report results back to the ingress router for admission control purposes. It should be noted that the RD agent in a core router simply monitors the current forwarding queue length and uses the simple FICC algorithm to update the Explicit Rate on RD packets. The ingress router is responsible for generating and terminating the RD packets. The egress router is responsible for echoing the RD packets. The RD loop operates in the control plane of a router.

## III. FAIR INTELLIGENT CONGESTION CONTROL

This section introduces the Fair Intelligent Congestion Control, its design considerations and goals. The main ideas behind FICC are as follows. First, as Figure 2 shows, FICC tries to maintain the queue size close to a target value  $Q_0$  selected so that the queue does not go empty and, consequently, the output line is always busy. Second, as Figure 2 also illustrates, FICC adjusts the allowed class rate to minimize variations in buffer queue length and in packet delays. Third, FICC attempts to allocate the available bandwidth fairly. Specifically, FICC tries to allocate bandwidth equally among aggregates (DSCPs) with equal status and to distribute the unused bandwidth (left over by constrained aggregates) fairly among the aggregates that can use an additional share. To achieve this objective, FICC oversells bandwidth when the network operates below the target point, as Figure 2 illustrates. Fourth, FICC maintains per-output queue information (not per-aggregate) to make the scheme more scalable. Finally, the FICC algorithm is kept simple to reduce its implementation complexity.

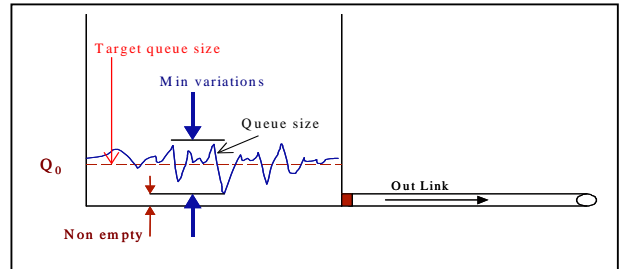


Figure 2. Design considerations at a router

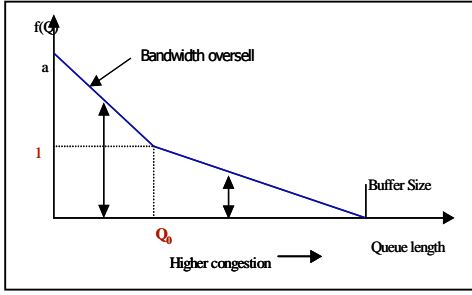


Figure 3. Selection of a queue control function

#### A. Maintaining the network at a desired level of operation

*Fairness.* Most congestion control schemes divide network operational status into two states: congestion state and non-congestion state. Normally a simple router buffer threshold is used to mark the transition from the non-congestion state to the congestion state. Some schemes may include a hysteresis by using two thresholds, a high threshold for the transition from non-congestion to congestion and a low threshold for the transition from congestion to non-congestion. Almost all congestion control schemes become active only when they detect that the network is in the congestion state and do not operate when the network is in the non-congestion state. This mode of operation results into several important characteristics. Firstly, the network is not efficiently utilized. Indeed, network congestion is usually detected locally, as some parts of the network may be congested while others may not be. Traffic passing from a congested part of a network is constrained by the congestion limit and may not be able to utilize its share of bandwidth in the non-congested part. FICC employs an overselling scheme that encourages and allows unconstrained aggregates to take up the unused bandwidth made available by constrained aggregates. This scheme improves the network bandwidth utilization (Fig. 3). Secondly, unfairness is introduced because aggregates that traverse more links are more likely to experience congested routers, and consequently, are more severely restricted than aggregates that traverse fewer links. Furthermore, if the congestion threshold is not chosen appropriately, the network oscillates widely between extremely busy conditions and idle conditions, which is not efficient and leads to large variations in packet delays.

*Maintaining a stable operating point instead of using a congestion threshold.* FICC defines a target operating point instead of a congestion threshold. FICC dynamically maintains the network operation around this target. In contrast, all threshold congestion mechanisms reduce the aggregate rates to bring the buffer occupancy below the threshold. They do not operate further when the buffer occupancy falls below the threshold. In other words, threshold congestion schemes do not try to maintain a target operating point. FICC, however, has a fixed point (the target operating point) and all aggregates work together to converge to this fixed point. Explicitly, FICC is always operational whether the network is below or above the target operating point. Consequently, with FICC, a queue length value is really an indication of the network operation level. An appropriate rate allocation scheme is one that takes this indication into account. Large queue length values imply

the network is overloaded, risking long delays and buffer overflow; small values imply light load and link underutilization. Therefore, it is desirable to aim for a target operating point where the queue length is acceptable for both throughput and delay.

*Target Operating Point.* In concrete terms, operating at the target operating point means that FICC maintains each of the router buffer queues at a fixed percentage of the buffer capacity. We call this level the Buffer Utilization Ratio (BUR). The motivation behind this idea is to maximize the network throughput by ensuring that the link associated with the buffer is never idle unnecessarily (Fig. 1). The Buffer Utilization Ratio (BUR) of an output queue of a router is defined as the target percentage of buffer capacity that should be occupied. When the target is met, the queue occupancy is  $Buffer\_Size * BUR$ . This target occupancy is designed to avoid link underutilization and the remaining buffer capacity  $Buffer\_Size * (1-BUR)$  is available to absorb packets that might arrive in the queue when the network becomes highly loaded. While BUR defines the target buffer operating point, the corresponding target queue length  $Q_0 (= BUR * Buffer\_Size)$  is often referred to instead of BUR in the description of FICC algorithm. It should also be pointed out that BUR only indicates the desirable long-term operational level. The actual buffer utilization fluctuates around this level. Section 4 discusses the effects of various parameter settings.

#### B. Queue Control Function

FICC determines the explicit rate (ER) of an aggregate (the maximum rate at which the network informs the source of the aggregate that it can support) based on the Mean Allowed Class Rate (MACR), which is used to measure the estimated fair share of the aggregate. This MACR in turn is based on the queue length at the router. It is thus essential to relate appropriately the buffer queue length to the degree of network congestion. FICC employs a function of the queue size queue, called the “queue control function” to express such a relationship. Since the queue builds up and drains out continuously, the congestion function should be continuous to regulate smoothly the queue fluctuations through the computed ER values. Sophisticated queue control functions may help to fine-tune the performance of the congestion control algorithm, but simpler functions are often preferred for simplicity of implementation. For simplicity, we selected the piecewise linear congestion function  $f(Q)$  shown in Figure 3. This function is defined as follows.

$$f(Q) = \frac{Buffer\_Size - Q}{Buffer\_Size - Q_0} \quad \text{for } Q > Q_0; \quad (1)$$

$$f(Q) = \frac{(a-1) * (Q_0 - Q)}{Q_0} + 1 \quad \text{for } Q \leq Q_0.$$

The function is specified by three parameters:  $a$ ,  $Q_0$  and  $Buffer\_Size$ .

#### C. Estimation of Allowed Class Rate and Explicit Rate Feedback

The aim is to maintain the router buffer queue length at a target operating level. In order to do so the router needs to

estimate its available bandwidth and advise the traffic sources appropriately. The router estimates the current traffic rate of all aggregates passing through it and allocates the available bandwidth fairly among its aggregates. FICC uses the MACR for this purpose. The MACR (Mean Allowed Class Rate) is an average of recent values of the current allowed class rates as indicated in the ACR field of the forward RD packets that the active source aggregate agent generates. The router computes MACR by a low pass filter of these ACR values. Specifically, upon receiving a RD, the router updates MACR as  $MACR = MACR + \beta * (ACR - MACR)$ , where  $\beta$  is the exponential average factor. In FICC, however, the value of MACR is a true exponential running average of the current load from all aggregates only when the network operates below the target operating point. When the network exceeds the target operating point, FICC does not allow MACR to increase further. That means that MACR does not track any ACR value larger than the current MACR when the queue is congested. This rule prevents all those aggregates whose ACRs are already equal or larger than the current MACR to increase their rates further, thereby preventing further loading of the network. This modification helps prevent excessive congestion. The router determines the Explicit Rate for each aggregates by calculating  $f(Q)*MACR$ . When the network operates above the target operating point, all aggregates have to reduce their rates (since  $f(Q) < 1$ ) to the same explicit rate and the throttling is performed fairly. However, when the network operates below the target operating point, all aggregates are allowed to increase their rate by a factor greater than 1 (that is what we mean by overselling), which enables aggregates that are capable of using the available bandwidth to take advantage of it.

#### D. Explicit Rate Algorithm of FICC

TABLE I. DESCRIPTION OF FAIR INTELLIGENT CONGESTION CONTROL

FICC calculates the explicit rate as follows.

##### Parameters

- \*  $\beta$  : The average ratio
- \* BUR : Buffer Utilization Ratio
- \*  $a$  : Congestion function parameter

##### Per Queue Variable

- MACR: Mean Allowed Class Rate
- DPF: Down Pressure Factor
- $Q_0$ : Target Queue Length

##### Initialization

$$Q_0 = BUR * BufferSize$$

##### At each router's network interface

```

if (receive RM (CCR, ER, DIR = forward))
  if (QueueLength > Q0)
    if (ACR < MACR)
      MACR = MACR +  $\beta$  * (CCR - MACR)
    else
      MACR = MACR +  $\beta$  * (CCR - MACR)
  if (receive RM (CCR, ER, DIR = backward))
    if (QueueLength > Q0)
      DPF = -----
      BufferSize - QueueLength
      BufferSize - Q0

```

```

else
  DPF = -----
  (a-1)*(Q0 - QueueLength)
  Q0
ER = max (MCR, min (ER, DPF * MACR))

```

## IV. SIMULATION STUDY

### A. Simulation Environment

Our simulation is based on the network simulator NS2 version 2.27 developed by VINT-Project. In the simulation environment, we develop a new control plane, which is capable of (1) reading the status from a normal FIFO buffer in the data plane and (2) running RD.

The simulation results are presented regarding the fairness, average packet delay, link utilisation, queue length, and traffic goodput. For each simulation, a set of results for normal DiffServ and FICC-DiffServ is presented for the purpose of comparison and discussion. For the normal DiffServ results, we use the basic DiffServ module developed by Shallwani et al [6].

### B. Simulation results

#### 1) Peer-to-peer configuration

The DiffServ domain applies four different classes of service, which are gold, silver, bronze and best effort. The different priorities for gold, silver, bronze and best-effort classes are 40%, 30%, 20% and 10% respectively.

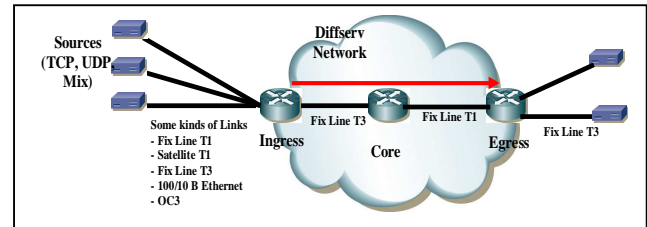


Figure 4. Peer-to-peer network configuration

This experiment is dedicated to TCP traffic, because the TCP source behaviours are affected by the RTT. The experiment is also performed to check the fairness of both the flow-level and class-level. To check the class fairness, all the silver, bronze and best-effort traffic arrive through fix line T1 links, which are 60 ms in delay, whereas the gold traffic come to the DiffServ through satellite T1 links with 500 ms in delay. Additionally, the flow fairness experiment is obtained by using the different type of T1 links for different flows in Gold class. The results are illustrated in Figure 5. It can be seen that the gold users using the satellite T1 link with longer RTT gain less goodput than the silver users. However, the FICC-DiffServ eliminates that problem, sharing resources for each class equally

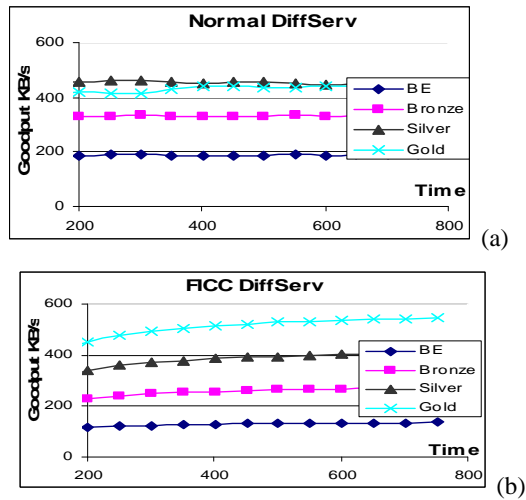


Figure 5. Class-level fairness – different RTTs

### 2) Parking Lot Configuration

In this topology, there are four traffic from four ingress routers (sE1, sE2, sE3 and sE4), all destined to the same network interface in egress router dE1. Traffic from sE1 and sE2 starts at the first core router C1 in the chain and traverses through all core routers. One of the remaining traffic starts at each ingress router in the chain. All the links have the same 10Mbps in bandwidth. The fair share for each traffic from different ingress routers is expected to be 25% in the congestion link between C3 and dE1.

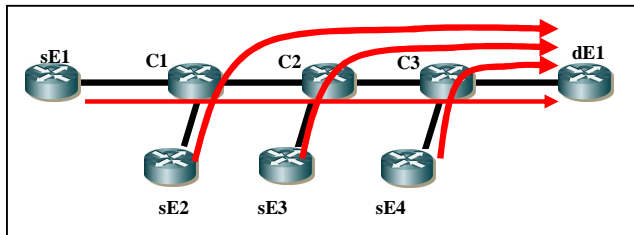


Figure 6. Parking lot network configuration.

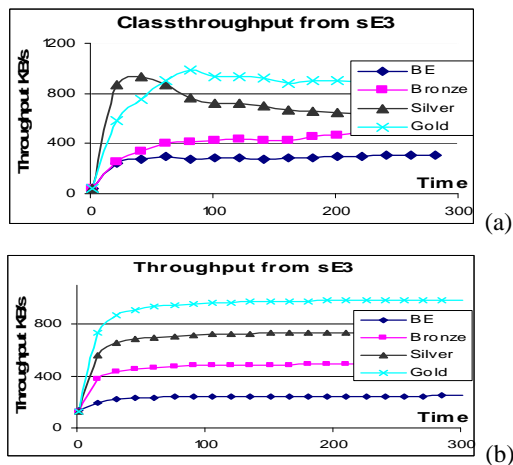


Figure 7. Parking lot – traffic throughput in FICC-DiffServ

Figure 7 shows the throughput per class from edge router SE3 in the normal DiffServ and FICC DiffServ in the case of all TCP traffic. This topology demonstrates that the FICC DiffServ is not only fair but also very efficient. The FICC DiffServ allocates 25% bandwidth for traffic from different edge routers, regardless of the number of routers it goes through. In addition, the traffic pattern in the FICC-DiffServ reflects the stability of the new system. However, in the normal DiffServ, the traffic from different edges received different bandwidth in congestion links due to the different number of hops they traversed. The larger number of links traffic traverses, the less bandwidth it can achieve. Moreover, the fairness level between classes in each ingress router of normal DiffServ domain is also degraded.

## V. CONCLUSION

The paper proposes a new congestion control algorithm: FICC. Instead of trying to reduce the network congestion, FICC aims to maintain a desired target operating point, estimates the accurate level of fair share for all aggregates, and allows bandwidth overselling. The resulting scheme is fair, effective, and simple. The performance of FICC can be summarized as follows. FICC outperforms existing per-queue congestion control algorithms in all situations: in fairness, throughput, packet delays, delay variations, scalability and robustness. In brief, FICC offers a most effective scalable solution to the explicit rate feedback congestion control problem. Recently, FICC has been deployed in End-Diff, a new scalable end-to-end QoS architecture [4]. The architecture consists of three basic components: a FICC-DiffServ architecture for autonomous systems ASs, a QoS-extended BGP between domains [7], and a per-flow admission control.

## REFERENCES

- [1] R. Braden, Clark, D., Shenker, S., "Integrated Service in the Internet architecture," *IETF RFC1633*, 1994.
- [2] D. B. S. Blake, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," *IETF RFC2475*, 1998.
- [3] G. Huston, "Next Steps for the QoS Architecture," *IETF RFC2990*, 2000.
- [4] D. B. Hoang, Phan, H. T., "End-Diff: An End-to-End QoS Architecture," *submitted for publication*, 2006.
- [5] H. T. Phan, Hoang, D. B., "FICC-DiffServ: A New QoS Architecture Supporting Resources Discovery, Admission and Congestion Controls," *International Conference on Information Technology and Applications (ICITA)*, 2005.
- [6] F. Shallwani, Ethridge, J., Piedad, P., and Baines, M., "Diff-Serv implementation for ns," <http://www7.nortel.com:8080/CTL/software>, 2000.
- [7] H. T. Phan, Hoang, D. B., "Extension of BGP to support multi-domain FICC-Diffserv architecture," presented at IEEE Conference on Advanced Information Networking and Applications (AINA06), Vienna, April 2006.