Scalable Architecture for Providing Per-flow Bandwidth Guarantees

Vasil Hnatyshin[1] and Adarshpal S. Sethi[2]

[1]Department of Computer Science
Rowan University
201 Mullica Hill Rd.
Glassboro, NJ 08028
hnatyshin@rowan.edu

[2]Department of Computer and Information Sciences,
University of Delaware,
Newark, DE 19716
sethi@cis.udel.edu

ABSTRACT

Despite numerous efforts, the problem of providing per-flow Quality of Service in a scalable manner still remains an active area of research. This paper introduces a scalable architecture for support of per-flow bandwidth guarantees, called the Bandwidth Distribution Scheme (BDS). The BDS maintains aggregate flow information in the network core and distributes this information among boundary nodes as needed. Based on the feedback from the network core the boundary nodes dynamically adjust resource allocations of individual flows. The BDS architecture consists of three main components: the admission control, the resource distribution mechanism, and the protocol for distributing the aggregate flow requirements in the network. This paper describes components of the BDS architecture and illustrates how they operate together to achieve scalable per-flow QoS.

Keywords: Quality of service, bandwidth distribution, network feedback, resource allocation

1. INTRODUCTION

To solve the problem of providing scalable per-flow Quality of Service, a number of service differentiation models have been proposed. The Integrated and Differentiated Service (DiffServ) models are among the most prominent approaches to providing Quality of Service in the Internet. The Integrated Services model [2] requires each router in the network to reserve and manage resources for the flows that travel through it. In large networks, millions of flows may simultaneously travel through the same core routers. In such cases, managing resource reservations on a per-flow basis may cause enormous processing and storage overheads in the core routers. As a result, the Integrated Services model is considered to be not scalable to large networks and thus is not widely deployed in the Internet. The DiffServ model [1] attempts to solve the scalability problem of the Integrated Services approach by combining flows that have similar quality of service requirements into traffic aggregates or classes. The DS core routers process incoming traffic based on the class the packets belong to and thus maintain and manage resource reservations only on a per-class/per-aggregate basis. The DiffServ approach provides a scalable solution to the QoS problem but it supports only coarse per-aggregate guarantees which in certain cases may not be adequate.

This paper examines the architecture of an alternative approach, called the Bandwidth Distribution Scheme (BDS). The BDS core routers do not maintain per-flow information (e.g. bandwidth requirements of individual flows); instead core routers keep aggregate flow requirements. The amount of information kept in the network core is proportional not to the number of flows but to the number of edge routers, which we believe does not raise scalability concerns. The edge nodes maintain per-flow information and fairly allocate network resources (e.g. bandwidth) among individual flows according to the flow requirements and resource availability. The dynamic resource allocation at the edge routers is enabled by the network feedback which consists of periodic path probing and explicit congestion notifications. Overall, the BDS architecture consists of: the admission control mechanism, which determines if a new flow can be admitted into the network, the resource allocation mechanism, which fairly distributes available bandwidth among individual flows, and the protocol for distribution of the aggregate flow requirements, which provides feedback to the network routers about the changes of network characteristics.

The BDS approach relies on the basic idea of performing per-flow management at the network edges and processing traffic aggregates in the network core. This idea is not new and has been examined before, for instance in [1,12,15,16]. However, the primary contribution of this work is a novel approach to aggregating flow information in the network core, dynamically distributing it among edge nodes, and then using the aggregate flow requirements for fair distribution of available bandwidth among individual flows.

The rest of the paper is organized as follows. Section 2 presents an overview of the BDS architecture. Section 3 introduces specification of flow requirements and the admission control mechanism. Definitions of fairness and the resource management mechanism are presented in Section 4, while Section 5 discusses the BDS network architecture and the protocol for dynamic distribution of aggregate flow requirements. Section 6 discusses implementation issues of the Bandwidth Distribution Scheme, while Section 7 provides an example of the BDS

operation. Finally, discussion and conclusions are presented in Sections 8 and 9 respectively.

2. THE OVERVIEW OF THE BDS ARCHITECTURE

The BDS architecture provides a scalable solution to the problems of fair per-flow bandwidth distribution and congestion control. This architecture consists of three components and a set of specifications and definitions. The BDS components are: per-flow admission control which denies access into the network for those flows that violate existing per-flow guarantees, per-flow resource allocation which dynamically distributes available bandwidth among individual flows, and the Requested Bandwidth Range Distribution (RBR) and Feedback protocol which distributes aggregate flow requirements and generates congestion notifications. The BDS specifications and definitions consist of the network architecture which defines the working environment of the BDS and makes the proposed solutions scalable to large networks, definition of the flow requirements which outlines the format for user expectations for traffic treatment, and definitions of fairness which specify what it means for the resource distribution to be fair. The BDS components along with the BDS specifications and definitions form the architecture of the Bandwidth Distribution Scheme as shown in Figure 1.

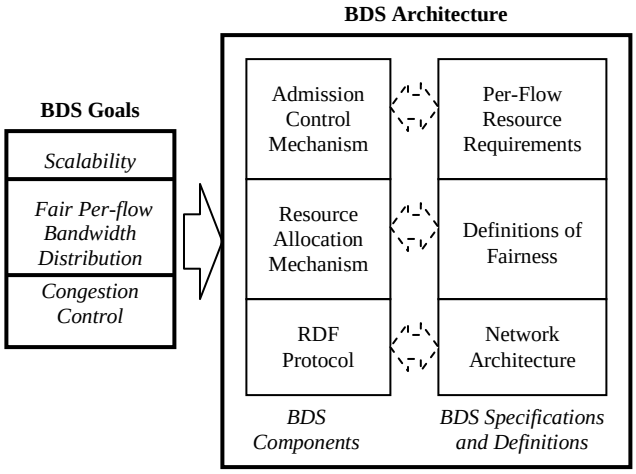

Figure 1. The BDS architecture

In the BDS architecture each BDS component is closely connected to a particular specification or definition as shown in Figure 1. For example, the BDS admission control determines if a new flow can enter the network based on the provided specification of flow requirements, while the BDS resource allocation mechanism distributes bandwidth among individual flows based on provided definitions of fairness. That is why, in subsequent sections, we introduce the BDS components together with their corresponding BDS specifications and definitions.

This paper defines a "flow" to be a sequence of packets that travel from a given source host to a given destination host. We only consider the flows that receive the BDS treatment and which are therefore subject to the BDS resource allocation. Similarly, the terms “resources”, “capacity”, “load,” or “bandwidth” mean the resources, bandwidth, etc. explicitly allocated by the network administrator for the BDS traffic. This definition of a flow, while different from the more conventional definition as a sequence of packets between individual source-destination applications (e.g., TCP or UDP streams), was chosen to simplify the presentation of the BDS scheme. The BDS architecture, as presented here, can be easily extended to apply to the conventional definition of a flow.

3. DEFINITION OF FLOW REQUIREMENTS AND ADMISSION CONTROL

In this paper we assume that both the minimum and the maximum transmission rates of a flow are known ahead of time. Thus, the flow requirements are defined in the form of a range which is called the Requested Bandwidth Range (RBR). The RBR of flow f, RBR^f, consists of two values: a minimum rate, b^f, below which the flow cannot operate normally, and the maximum rate, B^f, that the flow can utilize.

$$RBR^f = [b^f, B^f] \quad (1)$$

Based on this definition, the BDS network guarantees that each flow would receive at least its minimum requested rate, b^f, while the leftover resources in the network are fairly distributed among participating flows. To achieve these guarantees, the network allocates to each flow an amount of bandwidth not smaller than the flow’s minimum requested rate, and denies network access to those flows whose minimum rate guarantees cannot be met.

The purpose of admission control is to determine if a new flow can be admitted into the network at its minimum rate without violating existing QoS guarantees of other flows. The problem of admission control was extensively examined in the literature [3, 4, 8]. Traditionally, there are two types of admission control: *parameter-based* and *measurement-based*. In parameter-based admission control, the decision to admit a new flow is derived from the parameters of the flow specification. Usually, this type of admission control relies on worst-case bounds and results in low network utilization, although it does guarantee supported quality of service. Measurement-based admission control relies on measurements of the existing traffic characteristics to make the control decision. Measurement-based admission control supports higher network utilization. However, measurement-based admission control may occasionally cause the quality of service levels to drop below user expectations because of its inability to accurately predict future traffic behavior.

Since the network guarantees that each flow will receive at least its minimum requested rate, the edge nodes should check the current resource allocation on a path before granting a new flow request. Thus, to admit a new flow into

the network, the edge routers verify that the sum of the minimum requested rates of all the flows that follow a particular path, including a new flow, is smaller than the capacity of the bottleneck link on that path. Link k is a *bottleneck link* for flow f traveling on path P if k limits the transmission rate of f on P.

We formally define the BDS admission control as follows. Consider a network consisting of a set of L unidirectional links, where link[1] $k \in L$ has capacity C^k. The network is shared by the set of flows, F, where flow $f \in F$ has the RBR of $[b^f, B^f]$. At any time, the flow transmits packets at a rate R^f, called the *allocated rate*, which lies between b^f and B^f. Let $L_f \subseteq L$ denote the set of links traversed by flow f on its way to the destination. Also let $F^k \subseteq F$ denote the set of flows that traverse link k. Then a new flow ϕ with the RBR of $[b^\phi, B^\phi]$ is accepted in the network if and only if:

$$b^\phi + \sum_{f \in F^k} b^f \le C^k \quad \forall k \in L_\phi \qquad (2)$$

Thus, a new flow, ϕ, is accepted into the network only if the sum of the minimum requested rates of the active flows, including the new flow, is not larger than the capacity of each link on the path of flow ϕ to the destination. Equation (2) is often called the *admission control test.*

4. DEFINITIONS OF FAIRNESS AND THE RESOURCE ALLOCATION MECHANISM

In this section we introduce two definitions of fairness, examine and compare the ability of these definitions to maximize network throughput, and introduce the BDS resource allocation mechanism that fairly distributes available bandwidth among individual flows based on introduced definitions of fairness.

4.1. Definitions of Fairness

Consider a core router's interface k and a set of flows, F^k, that travel through it. The set F^k can be divided into two disjoint subsets: the subset, F_B^k, of flows that have link k as their bottleneck and the subset, F_{NB}^k, that contain all the other flows. These subsets are called *bottleneck flows* and *non-bottleneck flows*, respectively.

$$F^k = F_B^k \bigcup F_{NB}^k \qquad (3)$$

The *aggregate bottleneck RBR* and the *aggregate RBR* on interface k are defined as follows:

$$b_B^k = \sum_{f \in F_B^k} b^f \qquad B_B^k = \sum_{f \in F_B^k} B^f \qquad (4)$$

$$b^k = \sum_{f \in F^k} b^f \qquad B^k = \sum_{f \in F^k} B^f \qquad (5)$$

The aggregate bottleneck RBR is the sum of the RBRs of the bottleneck flows on link k, while the aggregate RBR is the sum of the RBRs of all the flows that travel through link k. The total allocated rate of the non-bottleneck flows is called the *non-bottleneck rate* and is denoted as R_{NB}^k. The amount of bandwidth left for distribution among the bottleneck flows is the difference between the capacity of link k and the non-bottleneck rate. This value is called the *bottleneck capacity,* C_B^k.

$$C_B^k = C^k - \sum_{f \in F_{NB}^k} R^f = C^k - R_{NB}^k \qquad (6)$$

When a link is not fully utilized, its bottleneck capacity could be larger then the sum of the allocated rates of the bottleneck flows. Table 1 provides a summary of the presented definitions.

Table 1. Summary of the traffic type definitions for fairness specification

Flows	RBR	Capacity
All flows: $F^k = F_B^k \bigcup F_{NB}^k$	***Aggregate RBR:*** $b^k = \sum_{f \in F^k} b^f$ $B^k = \sum_{f \in F^k} B^f$	***Link Capacity:*** $C^k = R_B^k + R_{NB}^k$ $C^k \ge \sum_{f \in F^k} R^f$
Bottleneck flows: $F_B^k = F^k - F_{NB}^k$	***Aggregate Bottleneck RBR:*** $b_B^k = \sum_{f \in F_B^k} b^f$ $B_B^k = \sum_{f \in F_B^k} B^f$	***Bottleneck Capacity:*** $C_B^k = C^k - R_{NB}^k$ $C_B^K \ge \sum_{f \in F_b^k} R^f$
Non-bottleneck Flows: $F_{NB}^k = F^k - F_B^k$	***Aggregate Non-bottleneck RBR:*** *Not used, not defined.*	***Non-bottleneck Rate:*** $R_{NB}^k = \sum_{f \in F_{NB}^k} R^f$ $R_{NB}^k \le C^k - C_B^k$

Based on the notation specified in Table 1, we introduce two definitions of fairness. First, the proportional fair share, FS_f^k, of the flow f on link k is defined as follows:

$$FS_f^k = b^f + (C_B^k - b_B^k)\frac{b^f}{b_B^k} = C_B^k \frac{b^f}{b_b^k} \qquad (7)$$

Using definition (7), each flow is allocated its minimum requested rate plus a share of leftover bandwidth. We call this definition of fairness *proportional fairness* because each flow receives an amount of bandwidth proportional to its

[1] In this paper the terms link, interface, and interface to a link are often used interchangeably.

minimum requested rate. This definition of fairness should not be confused with Kelly's proportional fairness [8, 9], which deals with different issues. Throughout the rest of this paper, the expression "proportional fairness" refers to the definition of fairness specified by equation (7).

A second definition of fairness uses a similar idea, except that the excess bandwidth is distributed proportionally to the difference between the flow's maximum and minimum requested rates. The idea is to allocate resources proportionally to the amount of bandwidth a flow needs to be completely utilized. We assume that a flow is completely utilized when it sends traffic at its maximum requested rate, B^f. That is why this definition of fairness is called *maximizing utility fairness*. The maximizing utility fair share, FS_f^k, of flow f on link k is computed as follows:

$$FS_f^k = b^f + \left(C_B^k - b_B^k\right)\frac{B^f - b^f}{B_B^k - b_B^k} \qquad (8)$$

4.2. Maximizing Allocated Rate in the Network via Definitions of Fairness

This section examines if resource distribution using the proposed definitions of fairness achieves our primary objective of maximizing allocated rate in the network. The network has allocated rate maximized if the allocated rate on every bottleneck link is maximized. Allocated rate on a link is the sum of allocated rates of those flows that travel through that link. Thus, the allocated rate on the bottleneck link is maximized if the bottleneck capacity on that link is completely distributed among the bottleneck flows. Also, the bottleneck link has its allocated rate maximized if all the bottleneck flows that travel through that link are allocated and transmit at their maximum requested rates.

Let us consider an example of Figure 2, where each link is provisioned with 48 Kbps of bandwidth. The RBR and path for each active flow are shown in Table 2, while the aggregate RBR is recorded next to the link as shown in Figure 2.

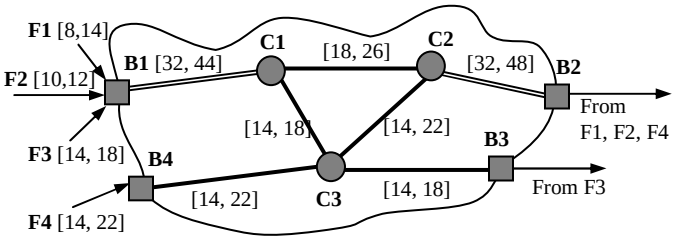

Figure 2. Example of the resource distribution

The network of Figure 2 contains two bottleneck links: B1-C1 and C2-B2. Link C2-B2 is the bottleneck for flows F1, F2, and F4. The bottleneck capacity of C2-B2 equals the link's capacity because all the flows that travel through link C2-B2 are bottleneck flows. Table 2 presents the resource distribution using both definitions of fairness. As Table 2 shows, the proportional definition of fairness does not utilize all available bandwidth on link C2-B2. In particular, using the proportional fairness only 45 Kbps out of 48 Kbps of the bottleneck link's capacity is distributed among the flows, leaving 3 Kbps of bandwidth unused. At the same time, flows F1 and F4 are sending traffic below their maximum requested rates and can utilize leftover bandwidth. Link B1-C1 is also underutilized; however, since the bottleneck flow F3 is allocated its maximum requested rate, the allocated rate on the link B1-C1 is maximized.

On the other hand, the maximizing utility fairness completely distributes all available resources among the flows and keeps bottleneck link C2-B2 fully utilized. When using maximizing utility definition of fairness, link B1-C1 also remains underutilized. However, as before, the allocated rate on B1-C1 is maximized.

Table 2. Example of the resource distribution

Flow	**Flow RBR**	**Path**	**Proportional Fair Share**
F1	[8, 14]	B1-C1-C2-B2	MIN (48*(8/32), 14) = **12**
F2	[10,12]	B1-C1-C2-B2	MIN (48*(10/32), 12) = **12**
F3	[14, 18]	B1-C1-C3-B3	MIN (24*(14/14), 18) = **18**
F4	[14, 22]	B4-C3-C2-B2	MIN (48*(14/32), 22) = **21**
Flow	**Flow RBR**	**Path**	**Maximizing Utility Fair Share**
F1	[8, 14]	B1-C1-C2-B2	MIN (8+16*6/16, 14) = **14**
F2	[10,12]	B1-C1-C2-B2	MIN (10+16*2/16, 12) = **12**
F3	[14, 18]	B1-C1-C3-B3	MIN (14+(48-26)*4/4, 18) = **18**
F4	[14, 22]	B4-C3-C2-B2	MIN (14+16*10/16, 22) = **22**

Now let us examine the conditions when the proportional and the maximizing utility definitions of fairness are unable to maximize allocated rate on the bottleneck link. The link's allocated rate is maximized in two cases:

1. The bottleneck flows are transmitted at their maximum requested rates. In this case the link's capacity may not be fully utilized.
2. The bottleneck capacity is completely distributed among the bottleneck flows. In this case the link's capacity is fully utilized.

To identify the conditions when the allocated rate on the link is not maximized, we need to examine when the sum of allocated rates of the bottleneck flows is smaller than the corresponding bottleneck capacity.

$$\sum_{f \in F_B^k} R_f^k = \sum_{f \in F_B^f} \min\left(FS_f^k, B^f\right) < C_B^k \qquad (9)$$

To determine when inequality (9) holds, we consider the following three cases:

Case 1: The fair shares of all the bottleneck flows are larger than their corresponding maximum requested rates and thus, all the bottleneck flows are allocated their maximum requested rate. Although the link is underutilized, its allocated rate is maximized because the bottleneck flows are allocated their maximum requested rates. This case corresponds to the situation on link B1-C1 of Figure 2.

Case 2: All the bottleneck flows are allocated the rates that correspond to their fair shares. The sum of the allocated rates of the bottleneck flows on a link equals the bottleneck capacity of that link. In this case the link capacity is completely utilized and the allocated rate on the link is maximized.

Case 3: Among the bottleneck flows that travel through the link there are flows that are allocated their maximum requested rates because their fair shares are larger than their corresponding maximum requested rates and there are flows that are allocated only their fair shares. The flows that transmit data at their maximum rates but below their fair shares cause the link to become underutilized. This case corresponds to the situation on link C2-B2 described in the example of Figure 2.

Let us examine the last case for both definitions of fairness in more detail. Using proportional fairness, Case 3 yields the following inequality:

$$R_B^k \frac{b^f}{b^k} > B^f \Rightarrow \frac{R_B^k}{b^k} > \frac{B^f}{b^f} \qquad (10)$$

Thus, the proportional fairness does not maximize the allocated rate on the link whenever the ratio between the bottleneck capacity and the minimum requested rate of the aggregate RBR is smaller than the ratio between the flow's maximum and minimum requested rates. The main reason for this phenomenon is the fact that the proportional fairness does not consider the maximum requested rates in the computation of the fair shares. The maximizing utility fairness, on the other hand, does include the maximum requested rates in computation of the fair shares and thus does not suffer from the above deficiency.

$$b^f + (R_B^k - b^k)\frac{B^f - b^f}{B^k - b^k} > B^f \Rightarrow$$

$$\frac{R_B^k - b^k}{B^k - b^k}\left(B^f - b^f\right) > B^f - b^f \Rightarrow$$

$$R_B^k - b^k > B^k - b^k \Rightarrow R_B^k > B^k \qquad (11)$$

According to inequality (11) the maximizing utility fairness causes the link to become underutilized only when the bottleneck capacity is larger than the maximum requested rate of the aggregate RBR. However, this means that all the bottleneck flows transmit traffic at their maximum requested rates and thus the allocated rate on the link is maximized.

In summary, the maximizing utility fairness is able to maximize allocated rate in the network, while the proportional fairness fails to do that whenever the inequality (10) holds, and thus, may require additional mechanisms to improve the overall performance in the network.

4.3. The Resource Management Mechanism

To distribute bandwidth according to equations (7) – (8), the resource management mechanism requires the knowledge of such link characteristics as the aggregate bottleneck RBR and the bottleneck capacity. However, these characteristics are not readily available in the network. Instead the core routers keep track of the capacity, the arrival rate, and the aggregate RBR for each outgoing link and distribute this information among the edge nodes. The edge nodes use the aggregate RBR and link capacity instead of the aggregate bottleneck RBR and the bottleneck capacity to compute fair shares of individual flows. The edge nodes compute the fair share of flow f on its bottleneck link k using the proportional and maximizing utility definitions of fairness as shown below. However, the flows that do not have link k as their bottleneck would not adjust their allocated rates.

$$FS_f^k = b^f + (C^k - b^k)\frac{b^f}{b^k} = C^k \frac{b^f}{b^k} \qquad (12)$$

$$FS_f^k = b^f + \left(C^k - b^k\right)\frac{B^f - b^f}{B^k - b^k} \qquad (13)$$

Clearly, such resource distribution may leave link k underutilized because the non-bottleneck flows will transmit data at rates below their fair shares on link k. Thus, the edge nodes require additional means for utilizing the leftover resources. A "water-filling" technique employed for implementation of the max-min fairness [13, 17] allows the edge nodes to completely distribute leftover capacity. The idea of the "water-filling" is to increase allocated rates of individual flows as long as the bottleneck link is not fully utilized.

Periodic path probing, that delivers information about availability of resources on the path, enables the edge routers to implement the "water-filling" technique. Thus, in the presence of excess bandwidth the edge routers increase allocated rates of individual flows until available bandwidth on the path is consumed completely. It was shown in [5] that by distributing leftover bandwidth proportionally to the individual flow requirements the resource management mechanism achieves an optimal resource distribution defined by equations (7) – (8).

The resource management mechanism enforces resource allocation through the token bucket. Thus, if a flow transmits data above its allocated rate then the token bucket discards all excess traffic of that flow. As a result, the flow injects the amount of data into the network that corresponds to its share of the allocated resources.

5. THE BDS NETWORK ARCHITECTURE AND THE RBR DISTRIBUTION AND FEEDBACK PROTOCOL

5.1. The BDS Network Architecture

The Internet consists of a large number of routers that are traditionally grouped into independent network domains as shown in Figure 3. A cluster of interconnected routers that are governed by the same administrator are called a *network domain*. Each network domain contains two types of nodes:

the *edge* or *boundary* routers and the *core* routers. Traffic enters a network domain through the edge nodes called *ingress* routers. It further travels through the core routers to reach the network boundary and exits the domain through the edge nodes called *egress* routers.

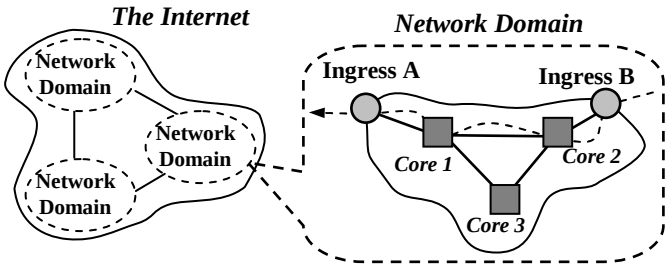

Figure 3. The BDS Network Architecture

The BDS core routers do not perform per-flow management and treat arriving traffic on per-aggregate basis, in a way similar to that of the Differentiated Services nodes [1]. The BDS core routers provide feedback to the boundary nodes about the changes of network conditions. The edge nodes maintain per-flow information and manage activation and termination of the flows. Based on the provided feedback the edge nodes compute the fair shares of bandwidth for their flows and then allocate available resources accordingly.

It is reasonable to assume that the number of active flows that enter and exit the network domain through a particular edge router is fairly small. Thus, managing per-flow information at the network boundaries allows this network architecture to scale well to large networks [1]. Furthermore, this architecture does not require the BDS approach to be set-up everywhere in the Internet at once. Instead, each network domain can choose to support the Bandwidth Distribution Scheme at its own discretion, and thus facilitates incremental deployment of the BDS architecture in the Internet. If a network domain decides to support the BDS, then a certain amount of resources are allocated for the BDS traffic. These resources are fairly distributed among the BDS flows only, thus isolating the BDS traffic from the rest of the non-BDS flows traveling through this domain. This paper examines the architecture of the Bandwidth Distribution Scheme within the confines of a single network domain. We plan to address the issue of inter-domain traffic and deployment of the BDS approach in the Internet in future work.

5.2. The RBR Distribution and Feedback (RDF) Protocol

The feedback protocol that governs the information sharing between the nodes in the BDS network is called the RBR Distribution and Feedback (RDF) protocol. The RDF protocol is one of the most important components of the BDS. The RDF protocol operates as the “glue” that holds the BDS architecture together by supplying information to the admission control and the resource management mechanism. The RBR Distribution and Feedback protocol consists of two major components that determine its name: distribution of the aggregate RBR and periodic feedback from the network.

The RBR Distribution and Feedback protocol consists of three distinct phases: the path probing phase, the RBR update phase, and the notification phase. Each phase is classified based on the information flow as either edge-to-core or core-to-edge. During the edge-to-core phases, the information travels from the edge nodes into the network core to update the aggregate flow requirements stored in the *Interfaces* Tables. At the same time, during the core-to-edge phases, information about the status of the network core is being distributed among the edge routers to refresh network information stored in the *Path* and the *Link* Tables.

The path probing phase discovers characteristics of a particular path and works as follows. The ingress node periodically generates a probe message on a path while the egress node sends the probe message with collected path characteristics back to the ingress node. The ingress node uses received information to update its Path and Link Tables. In addition, the core routers can discover the edge nodes that send traffic through their interfaces using the path probing phase. For example, upon the probe message arrival, the core routers retrieve the identity of the edge node that generated this probe and update corresponding edge node entry in the Interfaces Table, which contains the identity of the edge router and a countdown timer. If the entry in the Interfaces Table for this edge router already exists, then the core node resets the corresponding countdown timer. Otherwise, the core router creates a new entry for this edge router. The core router discards the edge node's information whenever the countdown timer expires.

The purpose of the RBR update phase is to notify the core routers about the changes to the aggregate RBR information upon flow activation or termination. The edge routers initiate the RBR update phase by generating the RBR update message on a particular path. Each core router renews its Interfaces Table based on the information received from the RBR update message. The egress node terminates progress of the RBR update message.

Only in the event of congestion do the core routers initiate the notification phase. In this case, the core routers generate congestion notification messages to the edge routers asking them to adjust allocated rates of their flows. The edge routers update their Path and Link Tables and re-compute the fair shares of the corresponding flows based on the information received from the congestion notification messages.

Thus, the edge routers update the Path and Link Tables based on the feedback received during the path probing and the notification phases, while the core routers update their Interfaces Tables during the RBR change and the path probing phases. Table 4 provides a summary and classification of the RDF protocol phases.

Table 4. Classification of the phases of the RDF protocol

Phase Name	Direction of info flow	Initiated by	Cause of Initiation
Path Probing	Edge-to-Core, Core-to-Edge	Edge Routers	Periodic
RBR Update	Edge-to-Core	Edge Routers	Flow Activation/ Termination
Notification	Core-to-Edge	Core Routers	Congestion

Phase Name	Updates Data Structures	Carries Information
Path Probing	Path Table, Link Table, Interfaces Table	Edge node ID, Path Characteristics
RBR Update	Interfaces Table	RBR Change values
Notification	Path Table, Link Table	Congested Link Characteristics

6. IMPLEMENTATION OF THE BDS ARCHITECTURE

This section discusses implementation details of the BDS architecture. In particular, it describes a set of data structures maintained in the routers of the BDS domain that allow answering the following questions: How do the core routers identify the edge nodes that should be notified about the network changes? How do the edge routers identify the flows that should adjust their allocated rates in the event of the network change?

6.1. The BDS Traffic Processing at the Edge Routers

The edge routers maintain the Service Level Agreement (SLA) Table that keeps track of the per-flow requirements negotiated between the end-user and the network domain. An entry in the SLA Table usually contains the following information: the source and destination addresses, the requested bandwidth range, the current allocated rate of a flow, and possibly flow status (e.g., active, idle, etc.). Since the flow SLA is usually negotiated ahead of time and is not frequently updated, the SLA Table is sorted and indexed based on the source-destination pair. This sorting allows efficient retrieval and update of flow information.

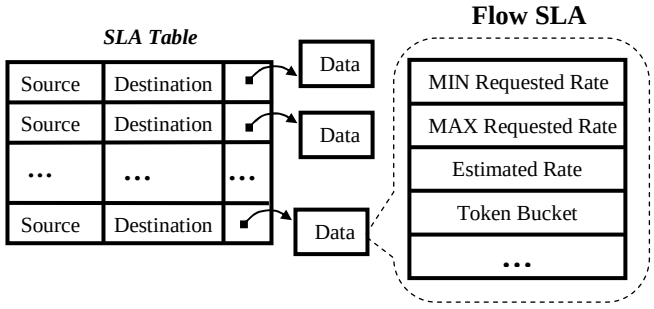

Figure 4. The SLA Table

The source-destination pair was chosen as the primary index into the SLA Table because we define a flow as a sequence of packets that travel from the source node to the destination. This definition of a flow simplifies the overall traffic classification at the edge routers. Identifying flows based only on the source-destination pair may limit flexibility of the flow differentiation (e.g., such flow identification does not allow distinguishing different flows that originate from the same source and travel to the same destination). However, such flow identification has little or no effect on the overall BDS architecture, since the flow definition could be easily extended to support a more complex differentiation without any modification to the main BDS components (e.g., admission control, resource management, or the feedback protocol). For example, the end user and the network domain can use the Type of Service (ToS) field of the IP header as an additional parameter for flow discrimination, which would require a slight modification of the classification element and the SLA Table only.

Let us examine how an arriving packet is processed at the edge router. Upon a new packet arrival, an edge node uses the header of the arriving packet to determine the identity of the flow the packet belongs to. Then, the edge node retrieves the status of the flow, which specifies if a flows is active or not, along with the flow's characteristics from the SLA Table.

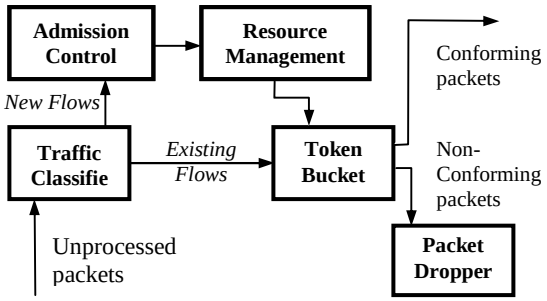

Figure 5. Packet processing at the edge nodes

If a packet belongs to a new flow, then the edge node initiates the admission control procedures. If the flow can be accepted into the network, then the edge node allocates a share of available resources to the new flow, which may entail adjusting allocated rates of the other flows, and forwards the packet further. Otherwise, the packet is dropped, and the source is notified that its request cannot be granted.

If the packet belongs to an already active flow, then the edge node verifies if the packet conforms to the rate requirements by passing the packet through that active flow's token bucket. The token bucket of each flow has its token generation rate set to the flow's allocated rate as determined by equations (10) and (11). If the packet conforms to the token bucket specification, then the packet is forwarded further. Otherwise, the packet is discarded. Figure 5 shows the packet processing at the edge router.

As Figure 5 shows, a packet from a new flow passes through the admission control and the resource management units before being forwarded. However, to compute the allocated rate of a flow, or to determine if the flow can be admitted into the network, an edge node needs to know the path that the flow will traverse and identity and characteristics of the bottleneck link. To provide the necessary information for the admission control and the resource management units, edge nodes maintain additional data structures that keep track of the active paths. We define a *path* or a *route* as a sequence of interfaces or links from the ingress router to the corresponding egress node. An *active path* is a route that is currently being used by at least one active flow.

The table that keeps track of all active paths and their characteristics is called a *Path Table* and is indexed by the identity (e.g., IP address) of the egress node. An entry in the Path Table contains a complete path to the corresponding egress node and a list of flows that traverse it. To minimize the amount of redundant information kept in the Path Table, the edge routers also maintain a *Link Table*. The Link Table keeps track of the individual link characteristics and is indexed by the IP address of the link (or more precisely, the IP address of the outgoing interface for the link). The list of flows and the complete path maintained in each Path Table entry are implemented via linked lists of pointers as shown in Figure 6.

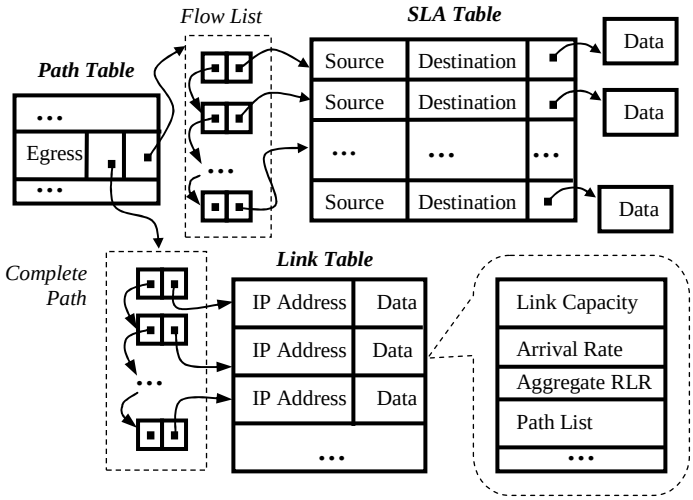

Figure 6. Data structures maintained in the edge node

The SLA Table is primarily used for packet classification, while the Path and the Link Tables are primarily used for admission control and resource management. For example, to determine if a new flow could be admitted into the network, the edge router identifies the path that the flow will follow, retrieves information about the bottleneck link on that path, and then performs the admission control test. In addition, the Path and the Link Tables allow the edge nodes to identify the flows influenced by the network changes.

Let us assume that characteristics of a particular link k in the network were changed causing the flows that travel through k to adjust their allocated rates. Once the edge router learns about this event it identifies the flows that should adjust their allocated rates as follows. First, the edge router identifies the entry in the Link Table of link k. Then, the edge router retrieves the list of paths that k belongs to. Next, for each entry in the path list, the edge node retrieves a set of flows that follow the corresponding path. The union of the resulting flow sets is a list of all the flows that travel through link k and thus should adjust their allocated rates.

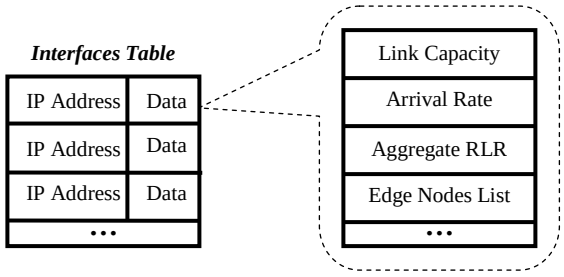

Figure 7. The Interfaces Table

6.2. The BDS Traffic Processing at the Core Routers

Now let us examine the data structure that helps the core routers to identify a set of edge nodes that should be notified in the event of network changes (e.g., the change of the characteristics of the outgoing link). Each core router maintains a single Interfaces Table (Figure 7) that keeps track of the core router's interface characteristics. Each entry in the Interfaces Table maintains an identity (e.g., IP address) of the outgoing link and its characteristics, including the list of edge nodes that send traffic though this interface.

Table 3. Summary of the data structures maintained by the BDS nodes

Name	Located At	Required by BDS Unit	Information Maintained
SLA Table	Edge Nodes	Traffic Classifier, Admission Control	Maintains flow identities and their service specification.
Path Table	Edge Nodes	Admission Control, Resource Allocation	Maintains active paths and the flows that follow these paths.
Link Table	Edge Nodes	Admission Control, Resource Allocation	Maintains characteristics of the traversed links.
Interfaces Table	Core Nodes	Feedback protocol	Maintains information about each outgoing link.

When characteristics of the link change (e.g., the link became congested), the core router consults the Interfaces Table to retrieve identities of the edge routers send traffic through that link. The core router generates and sends

explicit notification messages to each router in the retrieved list. Upon such message arrival, the edge routers adjust allocated rates of the corresponding flows as described in Section 6.1. Table 3 provides a summary of the BDS data structures, while the next section describes how the BDS data structures are maintained.

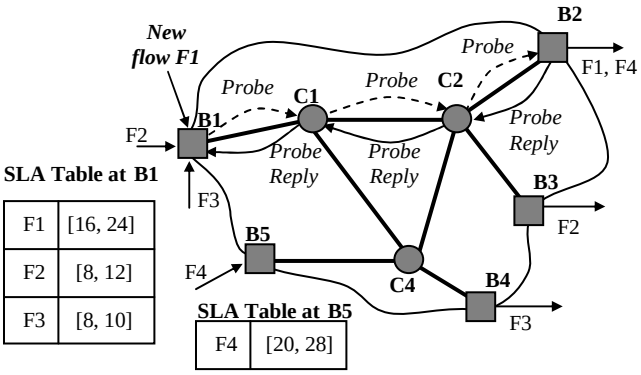

Figure 8. Path probing phase

7. EXAMPLE OF THE BDS OPERATION

Let us consider the example of Figure 8 which presents the network topology and the SLA Tables of the edge nodes B1 and B4. Each link in the network of Figure 8 is provisioned with 48 Kbps. Flows F2 and F3 travel from ingress node B1 to egress nodes B3 and B4, respectively, while flow F4 travels from ingress B5 to egress B2. New flow F1 requests permission to enter the network at edge node B1 to travel to egress node B2. However, edge node B1 does not have up-to-date information about the path to B2 and needs to initiate periodic path probing. The probe message collects information about each link on the path, and egress node B2 generates the probe reply message that carries collected information back to ingress node B1. Figures 9 and 10 show how the Path and the Link Tables of edge node B1 were updated after the path probing phase.

Path Table

Dest.	Path	Flows
B3	B1-C1-C2-B3	F2
B4	B1-C1-C4-B4	F3

Link Table

Link	RLR	Path List
B1-C1	[16, 22]	B3, B4
C1-C2	[8,12]	B3
C2-B3	[8,12]	B3
C1-C4	[8,10]	B4
C4-B4	[8,10]	B4

Figure 9. The Path and the Link Tables
BEFORE the path probing phase

Based on the results of the path probing, edge node B1 discovers that link C2-B2 is the bottleneck for flow F1. Since the minimum requested rate of the aggregate RBR (including flow F1) is smaller than capacity of link C2-B2 (e.g., 36 and 48 Kbps, respectively), new flow F1 is admitted into the network. As a result, edge node B1 initiates the RBR update phase and notifies all core routers on the path to egress B2 about the RBR change. Addition of another flow on the path to egress B2 causes congestion on link C2-B2 and triggers the notification phase. Core router C2 generates congestion notifications to edge nodes B1 and B5. Figure 11 shows the control message exchange during the RBR update and the notification phases, while Figure 12 shows how the Interfaces Table of router C2 is changed after the RBR update phase.

Path Table

Dest.	Path	Flows
B3	B1-C1-C2-B3	F2
B4	B1-C1-C4-B4	F3
B2	B1-C1-C2-B2	F1

Link Table

Link	RLR	Path List
B1-C1	[16, 22]	B3, B4, B2
C1-C2	[8,12]	B3, B2
C2-B3	[8,12]	B3, B2
C1-C4	[8,10]	B4
C4-B4	[8,10]	B4
C2-B2	[36,52]	B2

Figure 10. The Path and the Link Tables
AFTER the path probing phase

After receiving a congestion notification from C2, edge node B1 discovers that congested link C2-B2 belongs only to the path to edge node B2. From the Path Table, edge node B1 identifies F1 as the flow that contributes to congestion and adjusts its allocated rate accordingly. Similarly, upon receiving congestion notification, edge node B5 consults its Path and Link Tables to discover that flow F4 should slow down. Then, edge router B5 updates its Link Table with fresh information about link C2-B2 and computes a new allocated rate for F4.

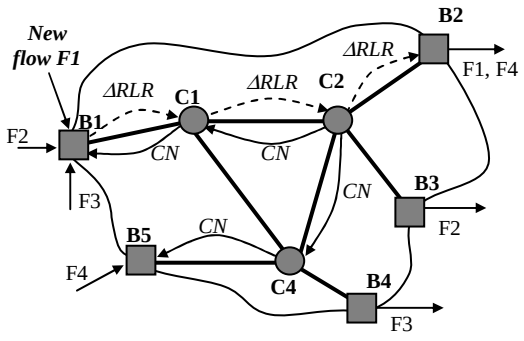

Figure 11. The RBR update and the notification phases

8. DISCUSSION AND RELATED WORK OVERVIEW

Most of the current architectures that support QoS in the Internet have emerged from various proposals by the Internet Engineering Task Force (IETF). In 1994, the IETF introduced Integrated Services [2] architecture, followed by the Differentiated Services [1] model in 1998. Although both

approaches address the same problem of supporting quality of service in the Internet, they are different in terms of implementation and provided services. Integrated Services supports end-to-end guarantees on a per-flow basis, while DiffServ attempts to provide end-to-end guarantees based on per-hop assurances for a small set of predefined traffic classes. At the implementation level, Integrated Services requires per-flow management in the network core, while the Differentiated Services model employs a network architecture that pushes per-flow management to the network edges.

Before the RLR Update Phase

Link	RLR	Edge Nodes
C2-B2	[20, 28]	B5
…	…	…

After the RLR Update Phase

Link	RLR	Edge Nodes
C2-B2	[36, 52]	B5, B2
…	…	…

Figure 12. The Interfaces Table after the RBR update phase

The architecture presented in this paper attempts to combine advantages of the Differentiated and Integrated Services models and provides support for building per-flow QoS services in a scalable manner. The network architecture enables the Bandwidth Distribution Scheme to become scalable, while per-flow bandwidth distribution at the network edges provides framework for deployment of the QoS services on a per-flow basis.

The RDF protocol relies on periodic path probing and explicit network feedback for dynamic bandwidth allocation and congestion control. This idea is not new and it has been examined before. In particular, the Explicit Congestion Notification (ECN) extension to IP [14] uses binary feedback to notify ECN-capable transports about congestion occurrences. Unlike the ECN extension, the network feedback in the BDS model not only notifies the edge routers about congestion but also carries additional information such as the arrival rate and aggregate flow requirements on the congested link. A similar idea is used in ATM networks for Available Bit Rate (ABR) congestion control [10], where the feedback carried by the resource management cells also includes rate information. However, the ABR congestion control relies on per-flow information stored in the network core and tries to achieve utilization goals first and only then seeks fairness. In contrast, in the BDS model core routers do not store per-flow information; instead they maintain aggregate flow requirements. Furthermore, the BDS approach tries to achieve utilization and fairness goals simultaneously: the edge nodes compute the fair shares of individual nodes so as to consume all bandwidth allocated for BDS traffic, and in the presence of excess bandwidth individual flows increase their transmission rates so as to preserve fairness. The Explicit Control Protocol (XCP) [11] generalized the ECN proposal by sending additional information about congestion. XCP also does not require per-flow information in the network core. However, unlike BDS, XCP is not a rate-based but a window-based protocol that separates utility control from the fairness control.

The BDS approach is designed primarily for support of per-flow bandwidth guarantees. A similar feedback-based idea of providing dynamic per-flow bandwidth allocation for elastic traffic sources called simple rate control algorithm was introduced in [12]. A traffic source is called elastic if it does not require a fixed rate and can adjust its transmission rate as needed. Unlike BDS, the boundary nodes in the simple rate control algorithm employ knowledge of the level of network congestion and the user utility functions to determine a fair resource distribution among elastic sources. The end users obtain the level of congestion through the explicit acknowledgements (ACK) that carry the number of congested links on a particular path.

The Stateless-Core approach [15] provides an interesting solution for supporting per-flow QoS without keeping per-flow information in the network core. The main idea of this scheme relies on the Dynamic Packet State (DPS), where control information is carried in the IP header of the data packets [16]. The routers use the DPS information to provide per-flow guarantees without maintaining per-flow state in the network core. However, because of such features as required route pining without which the SCORE/DPS model will fail, use of existing IP header fields to encode control information, additional per-packet processing in the network core, inability to distribute excess bandwidth, and possible network underutilization due to use of the upper bound of the aggregate reservation for admission control, the Stateless-Core architecture may not be deployed soon in today's Internet [15].

Evaluation of the BDS architecture [5, 6, 7] using Opnet network simulations showed that the Bandwidth Distribution Scheme is capable of fair distribution of available bandwidth among individual flows under all network conditions and thus, can support deployment of per-flow QoS services. Furthermore, the BDS eliminates congestion when it arises, keeps network utilization high by fair distribution of leftover bandwidth among corresponding flows, causes small overhead, and dynamically adjusts resource allocation in respect to network changes. Finally, evaluation of the BDS approach suggests that the Bandwidth Distribution Scheme is scalable to large networks, although further investigation of this feature under more realistic conditions is still needed.

9. SUMMARY AND CONCLUSOINS

In this paper we presented a scalable architecture for deployment of per-flow QoS services in computer networks. The BDS consists of three major components: the admission control, the resource management mechanism, and the RDF protocol. These components together with the network architecture and a set of definitions and specifications form the architecture for the Bandwidth Distribution Scheme. This paper in detail describes the BDS components, the data structures maintained in the BDS routers, and how the

Bandwidth Distribution Scheme operates to achieve fair distribution of bandwidth among individual flows.

REFERENCE:

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss," An Architecture for Differentiated Services," December 1998. IETF RFC 2475.

[2] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", June 1994, IETF RFC 1633.

[3] L. Breslau, S. Jamin, and S. Shenker, "Comments on the Performance of Measurement-Based Admission Control Algorithms," In *Proc. of IEEE INFOCOM'00*, March 2000.

[4] R. Gibbens and E. Kelly, "Measurement-based connection admission control", In *Proc. of 15th International Teletraffic Congress*, Amsterdam, Netherlands, June 1997.

[5] V. Hnatyshin, "Dynamic Bandwidth Distribution Techniques For Scalable Per-Flow QoS," Ph.D. Thesis, University of Delaware, 2003.

[6] V. Hnatyshin and A. S. Sethi, "Reducing load distribution overhead with message aggregation," In *Proc. of the 22nd IEEE International Performance, Computing, and Communications Conference*, pp. 227-234, April 2003.

[7] V. Hnatyshin and A.S. Sethi, "Fair and Scalable Load Distribution in the Internet," In *Proc. of the International Conference on Internet Computing*, pp. 201-209, June 2002.

[8] F. Kelly, P.B. Key, and S. Zachary, "Distributed Admission Control," *IEEE Journal on Selected Areas in Communications*, 18 (2000), pp.2617-2628.

[9] F. Kelly, A. Maulloo, and D. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, Vol. 49, No. 3, pp. 237-252, March 1998.

[10] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, February 2000, pp. 87-98.

[11] D. Katabi, M. Handley, and C. Rohrs, "Internet congestion control for future high bandwidth-delay product environments," In *Proc. ACM SIGCOMM'02*, August 2002.

[12] K. Kar, S. Sarkar, L. Tassiulas, " A Simple Rate Control Algorithm for Maximizing Total User Utility," In *Proc. of INFOCOM 2001*, April 2001.

[13] P. Marbach, "Priority Service and Max-Min Fairness," In *Proc. of IEEE INFOCOM'02*, June 2002.

[14] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", September 2001, IETF RFC 3168.

[15] I. Stoica, "Stateless Core: A Scalable Approach for Quality of Service in the Internet," Ph.D. Thesis, Carnegie Mellon University, 2000.

[16] I. Stoica, H. Zhang, "Providing Guaranteed Services without Per-Flow Management," In *Proc. ACM SIGCOMM'99*, September 1999.

[17] H. Tzeng and K. Sui, "On Max-Min Fair Congestion Control for Multicast ABR Service in ATM," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, pp. 545-556, April 1997.

[18] OPNET Modeler. OPNET Technologies Inc. http://www.mil3.com.