Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computer Networks 48 (2005) 525–547
Computer Networks
www.elsevier.com/locate/comnet

Estimating aggregate resource reservation for dynamic, scalable, and fair distribution of bandwidth

Vasil Hnatyshin *, Adarshpal S. Sethi

Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA

Received 18 August 2003; received in revised form 7 June 2004; accepted 27 October 2004
Available online 8 December 2004

Responsible Editor: J. Roberts

Abstract

The current best effort approach to Quality of Service in the Internet can no longer satisfy a diverse variety of customer service requirements, and that is why there is a need for alternative strategies. In order to solve this problem a number of service differentiation approaches have been proposed. Unfortunately, these schemes are often inadequate for providing proper service differentiation during periods of congestion. In this paper we introduce a new Bandwidth Distribution mechanism for supporting per-flow Quality of Service guarantees and for dealing with congestion. The Bandwidth Distribution scheme dynamically adjusts resource allocations at the network boundaries based on the network feedback. Within the Bandwidth Distribution framework we introduce a set of techniques for computing per-flow fair share. We evaluate these methods through simulation.

Keywords: Quality of Service; Bandwidth distribution; Network feedback; Resource allocation; Congestion control

1. Introduction

The current approach to provide QoS in the Internet is no longer adequate because of the increasing emergence of applications with diverse customer service requirements. As people become willing to pay more for services that satisfy their application needs, the one-service-for-all approach of today's Internet will become obsolete, creating a need for alternative strategies.

In order to solve this problem, a number of service differentiation models have been proposed. The Integrated and Differentiated Service models are among the most prominent approaches to

* Corresponding author. Present address: Department of Computer Science, Rowan University, 201, Mullica Hill Road, Glassboro, NJ 08028, USA.

E-mail addresses: hnatyshin@rowan.edu, vasil@cis.udel.edu (V. Hnatyshin), sethi@cis.udel.edu (A.S. Sethi).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2004.10.014

providing Quality of Service in the Internet. The Integrated Services model [2,3] 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 Differentiated Services 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 Differentiated Services 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. Although the Differentiated Services approach provides a scalable solution to the QoS problem, it supports only coarse per-aggregate guarantees that in certain cases may not be adequate.

This paper examines an alternative approach, called the Bandwidth Distribution Scheme (BDS). The primary objective of the Bandwidth Distribution Scheme is to combine advantages of the Integrated and Differentiated Services models and to provide support for building scalable per-flow QoS services in computer networks. The Bandwidth Distribution Scheme supports per-flow QoS through bandwidth allocation at the network edges on a per-flow basis. At the same time, the BDS achieves scalability by employing an architectural model in which the complexity of per-flow processing is moved out of the network core into the network edges. In this architecture, only the edge routers maintain per-flow information, while the core routers deal with traffic aggregates only. For this reason, the BDS approach has similar scalability advantages as the Differentiated Services model that uses the same architecture [1].

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. However, the primary contribution of this work is a novel approach to estimating aggregate flow requirements in the network core and then using the obtained information for dynamic fair pre-flow resource allocation at edge routers. Overall the BDS works as follows. The edge routers adjust the resource allocation of individual flows based on knowledge of flow bandwidth requirements and on feedback from the core routers. Network feedback allows edge routers to estimate the aggregate flow requirements and then compute fair shares of available resources for individual flows. The BDS employs the following two types of network feedback: (1) the edge routers periodically probe the network to update the characteristics of the active paths and (2) the core routers explicitly notify the edge routers about congestion. Overall, the BDS edge routers dynamically adjust bandwidth allocations of the flows in response to network changes such as presence of excess resources or congestion.

Besides supporting scalable per-flow QoS, the BDS approach attempts to maximize allocated bandwidth by distributing excess available bandwidth to the flows that can use it. If congestion arises, the excess bandwidth allocation is adjusted so that congestion is eliminated. An important goal of the scheme is to ensure that the available bandwidth is allocated fairly to the active flows. One of the major advantages of the BDS approach over the Differentiated Services model is its support for deployment of fine-grained per-flow QoS services similar to those of Integrated Services model. However, the Integrated Services architecture does not scale well to large networks and employs "hard" per-flow resource reservations which could yield network underutilization when the flows fail to consume all of the resources allocated to them. The BDS approach addresses these problems by employing an architectural model that supports scalability and by dynamically adjusting resource allocations of individual flows according to the changes in network conditions. For these reasons, the BDS approach could prove to be preferable over the existing Differentiated and Integrated Services models.

This paper does not explicitly examine the types of services that can be built using the BDS approach. Instead, it operates under the assumption that with the help of a mechanism for dynamic resource allocation, the network can support efficient deployment of scalable per-flow QoS, where each flow that enters the network is guaranteed to receive the amount of resources (e.g., bandwidth) within its requested bandwidth range. The BDS approach is designed to build services that can support bandwidth guarantees. This paper describes the components of the BDS framework that allocate available resources to individual flows in a scalable and fair manner while maximizing network throughput. Since the BDS approach deals with explicit bandwidth allocation, this paper also examines the problem of congestion control.

Overall, this paper describes the dynamic resource allocation mechanism used in the BDS model and investigates its feasibility via simulation studies; in particular, we examine the ability of the BDS edge routers to distribute available resources fairly, to keep link utilization high, and to eliminate congestion based on the network feedback. The rest of the paper is organized as follows. Section 2 introduces the framework of the BDS approach and argues its scalability. Section 3 introduces a set of estimation techniques for approximating the aggregate flow requirements required for dynamic resource distribution and examines two approaches to distributing leftover excess bandwidth on a path. Evaluation of the BDS approach and simulation results are presented in Section 4. Section 5 provides discussion and related work overview while Section 6 presents conclusions.

2. The BDS architecture

2.1. General idea

The main idea behind the feedback-based BDS is to dynamically adjust the per-flow allocated rates at the network boundaries. In a congestion-free network, the users transmit data at their desired rates. However, during congestion, the boundary nodes limit the amount of traffic admitted into the network. When a link becomes congested, the corresponding core router provides an indication to the boundary nodes to slow down.

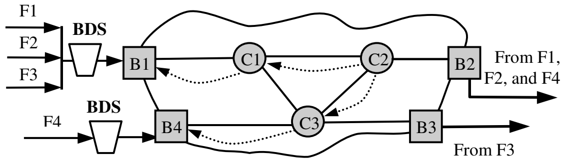

Fig. 1. Example of BDS approach.

The BDS uses an explicit message passing mechanism for providing congestion notifications to the network edges. These congestion notification messages contain the identity of the congested interface and its level of congestion. This information enables the boundary nodes to eliminate congestion and to preserve the minimum per-flow guarantees by adjusting allocated rates of the flows.

Consider the network topology shown in Fig. 1, where flows F1, F2 and F4 travel to boundary node B2 causing congestion on link C2–B2. In this case, core router C2 provides feedback to boundary nodes B1 and B4 in the form of congestion notifications. Upon congestion notification arrival, boundary nodes B1 and B4 adjust allocated rates of flows F1 and F2, and F4, respectively. Flow F3 continues transmitting traffic at the same rate since it does not contribute to congestion. After boundary nodes B1 and B4 adjust resource allocation of their flows, congestion at link C2–B2 is eliminated.

The BDS framework consists of three major components that allow the edge nodes to dynamically adjust allocated rates of the flows in response to network changes. These components are: the network architecture, the resource management unit, and the Requested Bandwidth Range (RBR) Distribution and Feedback (RDF) protocol. In subsequent subsections, we specify the architectural model of the BDS framework and describe the components of the scheme.

2.2. BDS architectural model

The Internet consists of a large number of routers that are traditionally grouped into independent network domains as shown in Fig. 2. A cluster of interconnected routers that are governed by the same administrator is called a *network domain*. Each network domain contains two types of nodes: the *edge* or *boundary* routers and the *core*

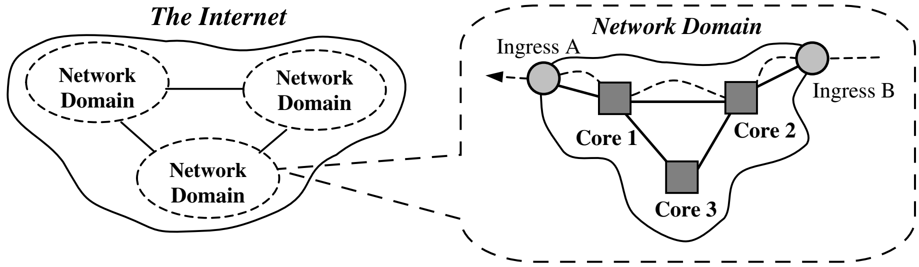

Fig. 2. The BDS architectural model.

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.

The core routers are not concerned with per-flow management and perform functions similar to those of the Differentiated Services nodes [1]. In order to support the BDS model, the core routers provide feedback to the boundary nodes about the network conditions. The edge nodes maintain per-flow information and manage activation and termination of the flows. They determine the fair share of each flow based on the provided feedback 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 will not raise scalability concerns [1]. In addition, this architectural model allows incremental deployment of the Bandwidth Distribution Scheme in the Internet. The BDS does not require being set-up everywhere in the Internet at once. Instead, each network domain can choose to support the Bandwidth Distribution Scheme at its own discretion. If the network decides to support the BDS, then a certain amount of resources should be allocated for the BDS traffic. These resources will be fairly distributed among the BDS flows only, thus isolating the BDS traffic from the rest of the flows traveling through this domain. This paper examines the performance of the Bandwidth Distribution Scheme within a single network domain and assumes that by allocating resources to the BDS traffic we perfectly isolate it from the rest of the non-BDS flows. We plan to address the issue of inter-domain traffic and deployment of the BDS approach in the Internet in future work.

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, 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. Since the BDS processing is done at the IP layer, differentiating among individual TCP and UDP streams would require the edge routers to access the corresponding transport layer headers. The BDS architecture, as presented here, can be easily extended to apply to conventional definition of a flow. Specifically, some of the BDS processing should be added to the transport layer of the source nodes. This addition will not cause any changes to the BDS processing in the network layer. As before, the core routers would provide network feedback and the edge routers would compute the fair shares on a per-source–destination basis and adjust the resource allocation accordingly. However, in addition, the edge routers would forward the computed per-source–destination fair shares to the source nodes that would then distribute these resources among individual flows.

2.3. Resource management mechanism and definitions of fairness

Resource management is a mechanism for sharing available resources (e.g., bandwidth) among active flows, while definitions of fairness are the rules that determine how the available resources are being distributed. The edge nodes distribute available bandwidth among individual flows based on their resource requirements. Flow resource requirements are defined in the form of a range, called the *Requested Bandwidth Range* (RBR), which is assumed to be known ahead of time. The RBR of a flow 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. The allocated rate R^f of flow f is limited by the flow's RBR and lies within this requested range.

$$RLR^f = [b^f, B^f]. \tag{1}$$

Such resource requirements are applicable for elastic traffic that can tolerate frequent rate change and can utilize excess bandwidth that becomes available due to changes in the network conditions. Overall, the BDS approach is most suitable for long-lived elastic applications that can tolerate and benefit from frequent changes in available resources such as video, large data transfers, and FTP.

Let us denote by F^k the set of flows that travel through link k; the total amount of resources $[b^k, B^k]$ requested on link k, which we call the *aggregate RBR* on the link is defined as follows:

$$[b^k, B^k] = \left[\sum_{f \in F^k} b^f, \sum_{f \in F^k} B^f\right]. \tag{2}$$

A link is a bottleneck for a flow if this link limits the allocated rate of that flow. Each edge node computes the fair share FS_f^k of flow f that travels through bottleneck link k with capacity C^k as follows:

$$\mathrm{FS}_f^k = b^f + (C^k - b^k)\frac{b^f}{b^k} = C^k \frac{b^f}{b^k}. \tag{3}$$

Using definition (3) each flow is allocated its minimum requested rate plus a share of the leftover bandwidth proportional to its minimum requested rate. Since flow f cannot transmit data at a rate higher than its maximum requested rate, the allocated rate R^f of flow f is limited by the flow's maximum requested rate B^f.

$$R^f = \min(\mathrm{FS}_f^k, B^f). \tag{4}$$

Flows that originate from the same ingress node and travel on the common parts of a path might have different bottleneck links in the network. As a result, if flow $f1$ travels through link k but has a bottleneck on link l, it may not need to adjust its allocated rate according to link k's information even though its resource requirements are added to the aggregate RBR of link k. As a result, link k may become underutilized and flows that have k as their bottleneck can benefit from the available excess bandwidth by increasing their allocated rates. In such cases, the fair share of these flows will include the share of leftover bandwidth.

2.4. The RBR distribution and feedback protocol

The third part of the BDS framework is the RBR Distribution and Feedback (RDF) protocol that notifies the edge nodes about the network changes. The feedback provided by the RDF protocol allows the edge nodes to estimate the aggregate RBR on the congested links. The RDF protocol is fairly simple and consists only of two phases: the path probing phase and the notification phase. The path probing phase discovers new paths, alerts the edge nodes about the presence of excess bandwidth on the path, and helps the core nodes to identify and keep track of the edge routers that should be notified during congestion. The notification phase alerts the boundary nodes about congestion in the network.

2.4.1. The path probing phase

The edge routers initiate the path probing phase for a particular path only if a new flow activates and the route characteristics are unknown to the edge router. The edge nodes probe the network only on a per-active-path basis. The path probing phase consists of periodic messages that travel to the egress node of a particular path and back. While traversing the network domain the probe messages collect the IP addresses, the estimated arrival rates, and the capacities of the router interfaces they pass through. The probes terminate at the egress nodes, which are the last routers within the domain on the path to the destination. The egress nodes forward the probes back to the ingress nodes that generated them in messages called probe replies. The first probe message generated on a path serves as the tool for discovery of the route to the destination. The edge nodes store collected information about the probed path in the Path Table. This information is used to update allocated rates of the flows that travel on that path.

The core routers use the path probing phase to identify and keep track of the edge nodes that should be notified during congestion. The edge nodes probe only active paths, e.g., the paths that have flows traveling on them. Thus, the core routers can use the probe message arrival as an indication that a particular edge router is still active and should be notified in the event of congestion. The core routers maintain a soft state for each active edge router. Such soft state contains the identity (e.g., IP address) of the edge router, a countdown timer, and identity of the core router's outgoing interface on which the edge router's traffic departs. The core routers update the soft-state information as follows.

Upon the probe message arrival, the core router retrieves the identity of the edge node that generated this probe from the source field in the packet's IP header. If the soft state for this edge router already exists, then the core node resets the corresponding countdown timer. Otherwise, the core router creates a new soft state entry for this edge router. The core router discards the edge node's soft state information whenever the countdown timer expires.

2.4.2. Notification phase

A core router initiates the congestion notification phase when one of its outgoing interfaces becomes congested. Interface i of a core router is congested if the arrival rate R^i of the traffic transmitted on this interface i is larger than the capacity, C^i, of the link attached to this interface [1]. The core router starts the notification phase by retrieving identities of the edge routers that should be notified and generating congestion notification messages to each of them. A congestion notification (CN) message contains the identity of the congested interface, its capacity, and estimated arrival rate on this interface. The edge routers use the congestion notification messages to estimate the aggregate RBR and, based on the results, distribute available resources among the flows. The notification phase terminates after all the edge routers receive their corresponding notification message and adjust resource allocation.

2.5. The BDS processing at the edge and core routers

The main responsibilities of the edge nodes in the BDS architecture are maintaining resource requirements of those flows that enter the network through them and keeping track of the active paths that these flows traverse. The edge nodes maintain flow requirements in the SLA Table. To avoid keeping redundant path information, the edge nodes maintain two additional tables. The first table, called the Path Table, keeps the list of active paths and corresponding flows that travel through those paths. Characteristics of individual links are stored in a separate table, called the Links Table. Fig. 3 illustrates the data structures maintained in each edge node.

The BDS processing in the core routers consists of three primary tasks: estimation of the arrival rate on each of the outgoing links, maintaining the list of edge routers to be notified in the event of congestion, and responding to the RDF protocol messages. To approximate the arrival rate on each of its interfaces and to keep the processing in the network core simple, the BDS employs an

[1] We use the term capacity C^i to denote the total bandwidth on link i that can be allocated to active flows. It is often the case that a link cannot be operated at full capacity because traffic variability can cause excessive delays. In such a situation, C^i reflects the fraction of the actual capacity that may be allocated to the flows.

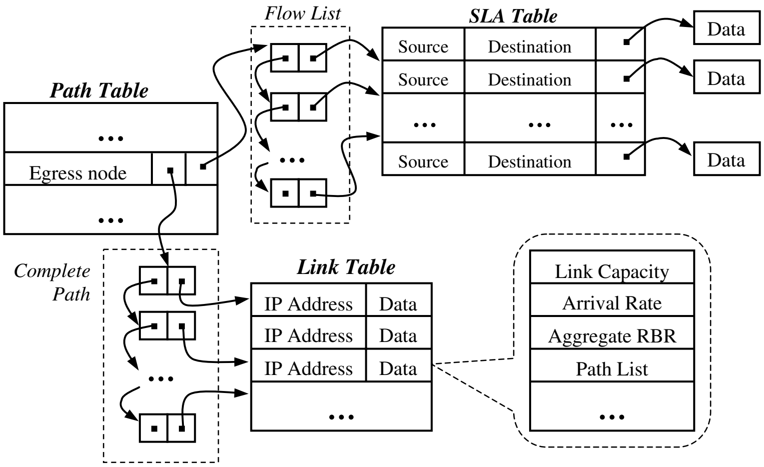

Fig. 3. Data structures maintained in the edge nodes.

exponential weighted moving average (EWMA) algorithm [4,5].

$$R^k = \alpha R^k_{\mathrm{old}} + (1-\alpha)R^k(\tau), \quad 0 \leqslant \alpha \leqslant 1, \tag{5}$$

$$R^k(\tau) = \frac{\sum_\tau \mathrm{size(packet)}}{\tau}, \tag{6}$$

where the R^k_{old} is the previous estimation of the arrival rate on link k, $R^k(\tau)$ is the arrival rate on link k during the time interval τ, and α is a constant. The EWMA algorithm was chosen for estimating the arrival rate because of its simplicity and small overhead. In simulations, the time interval τ was set to 1 s because with such configuration, the estimation mechanism appeared to provide the most accurate values of the arrival rate. Clearly, the choice of the time interval and the rate estimation mechanism influences the performance of the BDS approach. Although this paper does not address this issue, we hope to examine it further in the future.

2.6. Scalability of the BDS architecture

The presented BDS approach is scalable because of the network architecture employed by it. In BDS networks, the edge nodes maintain resource requirements on a per-flow basis and network information on a per-path basis. We assume that the number of flows that enter the network domain through a particular edge node is fairly small and does not raise the scalability problem. The Differentiated Services model makes a similar assumption. Furthermore, since the number of active paths originating from a particular edge router is not larger than the number of flows that enter the network at that edge router (e.g., a single flow cannot travel through two different paths simultaneously), the edge nodes can maintain network information on a per-path basis without raising scalability concerns.

The amount of information kept in the core routers is proportional to the number of edge nodes that send traffic through that core router. However, the core routers maintain information only about edge nodes that belong to the same network domain and not the whole Internet and that is why it is reasonable to assume that maintaining per-edge node information in the network core does not cause a scalability problem. Thus, the architectural model of the BDS approach is scalable to large networks.

3. The resource management mechanism

The BDS resource management mechanism leads to a more accurate and fair resource allocation among individual flows as compared to the rate control mechanisms that use TCP's "additive increase–multiplicative decrease" approach. In the absence of congestion, TCP increments transmission rates of individual flows (congestion window) by one full-sized segment per-round-trip time (RTT) (linear increase during congestion avoidance). However, when congestion occurs, TCP cuts the flow transmission rates in half. While this policy has been satisfactory for TCP, it resulted in unfairness under certain conditions [6,13]. The BDS resource management mechanism employs a different approach that considers the bandwidth requirements of individual flows, the aggregated flow requirements, and the amount of excess traffic on congested nodes in the network. Instead of TCP's gradual increment of flow transmission rates to the optimal values, the BDS "instantaneously" computes transmission rates of the flows while preserving bandwidth requirements, guaranteeing fairness, and eliminating congestion in the network if it arises.

The BDS resource management mechanism consists of two independent parts: resource allocation during congestion or the rate reduction mechanism, and resource distribution in the presence of excess bandwidth or the rate increase mechanism. This section examines four variations of the rate reduction mechanism and two approaches to rate increase. The mechanisms for rate reduction and rate increase approximate the resource distribution defined in Section 2.3.

3.1. The rate reduction mechanisms

The rate reduction approaches described in this section rely on the congestion notifications to adjust allocated rates of the flows that contribute to congestion. The first approach, called *proportional rate reduction* or *naïve method*, assumes that the flows that traverse the overloaded link send traffic above their fair shares and thus should slow down proportionally to their RBR. The remaining three approaches, called *simple two-step rate reduction*, *2CN two-step rate reduction*, and *hybrid rate reduction*, rely on consecutive congestion notifications to estimate the aggregate RBR on the path and then adjust allocated rates of the flows accordingly. The rate reduction mechanism relies on the congestion notification messages that deliver the identity of the congested interface k, its capacity, C^k, and the estimated arrival rate value, R^k, to the edge nodes. As before, let symbol F^k denote the set of flows that contribute to congestion on interface k, symbol R^f denote allocated rate of the flow f, and symbols b^f and B^f denote minimum and maximum requested rates of the flow f, respectively.

3.1.1. Naïve rate reduction method

The naïve method, presented in Fig. 4, assumes that each flow that contributes to congestion transmits data at a rate higher than its fair share and thus should slow down. The flows decrease their allocated rates by an amount that is proportional to their respective minimum requested rates (MRR) and the total excess traffic arriving on the congested interface. Then, assuming that the allocated rate of the flow is always smaller than its maximum requested rate, B^f, proportional rate reduction is defined as follows:

$$R^f = \max\left(R^f - b^f\frac{R^k - C^k}{C^k}, b^f\right). \tag{7}$$

Edge Router	***Core Router with congested link k***

Initialization:
NONE
Processing:
IF (CN arrives) {
Identify flows that travel via k;
Adjust flow rates according to (7);
}
	Initialization:
	Reset timer;
	Processing:
	IF (($R^k > C^k$) **AND** (timer >= inter-CN delay)){
	Identify Edge nodes that transmit over k;
	Transmit CN to identified Edge nodes;
	Reset timer;
	}

Fig. 4. Naïve rate reduction method.

The rate reduction method defined by Eq. (7) is able to eliminate congestion in the network; however, it may fail to distribute available bandwidth fairly among individual flows. The naïve method guarantees fair Bandwidth Distribution only if the allocated rates of all participating flows are proportional to their corresponding MRRs.

$$\forall f_1, f_2 \in F^k \quad \frac{R^{f_1}}{b^{f_1}} = \frac{R^{f_2}}{b^{f_2}}. \tag{8}$$

3.1.2. Simple two-step rate reduction

The two-step rate reduction techniques estimate the interface MRR and, based on obtained values, fairly distribute available bandwidth among individual flows. The two-step rate reduction techniques work as follows. Upon the first congestion notification arrival, the edge nodes adjust allocated rate of their flows using proportional rate reduction as defined by Eq. (7). If congestion is not completely eliminated, then a second congestion notification is received after some time. Based on the arrival rate reported by the second congestion notification, the edge nodes compute the interface MRR and corresponding allocated rates of the flows as follows. The superscripts are used to distinguish between the rate reduction upon the first and second congestion notification arrivals. When the first congestion notification arrives from overloaded link k, the allocated rate of flow f is computed as follows:

$${}^1R^f = R^f - b^f \frac{{}^1R^k - C^k}{C^k}. \tag{9}$$

After the first congestion notification, the total rate decrease observed at the congested interface k equals the following value:

$$\sum_{f \in F^k} b^f \frac{{}^1R^k - C^k}{C^k} = b^k \frac{{}^1R^k - C^k}{C^k}. \tag{10}$$

Upon the second congestion notification arrival, the edge node computes the total rate decrease and the interface MRR, b^k, on congested link k as follows:

$${}^1R^k - {}^2R^k = b^k \frac{{}^1R^k - C^k}{C^k}, \tag{11}$$

$$b^k = ({}^1R^k - {}^2R^k) \frac{C^k}{{}^1R^k - C^k}. \tag{12}$$

Once the edge router obtains the interface MRR, it computes the fair share of each flow that travels through the congested link.

$$\forall f \in F^k \quad R^f = C^k \frac{b^f}{b^k}. \tag{13}$$

The core routers delay generation of the second congestion notification to ensure that the rate estimation mechanism in the network core detects the rate reduction caused by the first congestion notification and converges to an accurate estimation value of the arrival rate on the interface, reported in the second congestion notification. The delay value should be at least the sum of the largest round trip time between the congested interface and the edge routers, and the time required by the core router to obtain an accurate estimation of the arrival rate on the congested interface. The rate reduction method described by Eqs. (9)–(13) is called *simple two-step rate reduction* and is shown in Fig. 5.

3.1.3. 2CN two-step rate reduction

If the rate reduction carried out in the first step is sufficient to eliminate the congestion, then the second congestion notification is not generated. In such cases, simple two-step rate reduction may fail to correctly estimate the interface MRR resulting in an unfair distribution of available bandwidth. A third method called *2CN two-step rate reduction* enforces generation of congestion notifications in sets of two, and thus eliminates this potential problem. During congestion, the core routers that implement a 2CN two-step rate reduction always generate two congestion notification messages, even if congestion is eliminated after the first step. This method is described in Fig. 6.

Since the two-step rate reduction techniques rely on the approximate value of the arrival rate at the congested interface, the value of the estimated interface MRR could be different from the exact value of the interface MRR (the sum of MRRs of the participating flows). The estimated interface MRR computation is the same at all the ingress nodes and thus each flow is transmitted

Edge Router	*Core Router with congested link k*

Initialization:
first_CN = **TRUE;**
Processing:
IF (CN arrives) {
Identify flows that travel via k;
 	
IF (first_CN){
Adjust flow rates according to (9);
first_CN = **FALSE**;
}
ELSE{
Compute interface MRR according to (12);
Adjust flow rates according to (13);
}
}
	Initialization:
	Reset timer;
	Processing:
	IF (($R^k > C^k$) **AND**
	(timer >= inter-CN delay)){
	
	Identify Edge nodes that transmit over k;
	Transmit CN to identified Edge nodes;
	Reset timer;
	}

Fig. 5. Simple two-step rate reduction method.

Edge Router	*Core Router with congested link k*

Initialization:
first_CN = **TRUE;**
Processing:
IF (CN arrives) {
Identify flows that travel via k;
 	
IF (first_CN){
Adjust flow rates according to (9);
first_CN = **FALSE**;
}
ELSE{
Compute interface MRR according to (12);
Adjust flow rates according to (13);
}
}
	Initialization:
	Reset timer;
	first_CN_gone = **FALSE**;
	Processing:
	IF ((($R^k > C^k$) **AND** (timer >= inter-CN delay))
	OR
	((timer >= inter-CN delay) **AND** (first_CN_gone)) {
	
	Identify Edge nodes that transmit over k;
	Transmit CN to identified Edge nodes;
	Reset timer;
	first_CN_gone = **NOT** (first_CN_gone);
	}

Fig. 6. 2CN two-step rate reduction method.

at a rate proportional to its MRR. However, since the estimated interface MRR is not equal to the exact interface MRR value, the resource distribution mechanism may cause the congested link that initiated the rate reduction process either to become underutilized or to remain overloaded.

3.1.4. Hybrid two-step rate reduction

To eliminate the above problem, a fourth method called *hybrid two-step rate reduction* is introduced. For each congestion episode the hybrid two-step rate reduction method counts the number of CN messages arrived at the edge router. Initially when the number of CN messages is less than or equal to two, the flows adjust their allocated rates the same way as in the 2CN two-step rate reduction method. This initial rate reduction ensures that all the ingress nodes compute the same value of the interface MRR, even though this value may be inaccurate. Furthermore, it guarantees that all the flows transmit data at rates proportional to their MRRs, which makes the proportional rate reduction method applicable. Subsequently, if after the first two CN messages congestion remains, then the flows use proportional rate reduction to adjust their allocated rates. Fig. 7 illustrates the idea of hybrid rate reduction method.

3.2. The rate increase methods

The edge nodes increase allocated rates of the flows when the path probing reports the presence of excess bandwidth on the path. The edge nodes compute the excess bandwidth, EB^P, on the path P as follows:

Edge Router	***Core Router with congested link k***

Initialization:
CN_count = 0;
first_CN = **TRUE**;
Processing:
IF (CN arrives) {
Identify flows that travel via k;
CN_count = CN_count +1;
 	
IF (CN_count > 2){
Adjust flow rates according to (7);
}
ELSE IF (first_CN){
Adjust flow rates according to (8);
first_CN = **FALSE**;
}
ELSE {
Compute MRR according to (12);
Adjust flow rates according to (13);
}
}
	Initialization:
	Reset timer;
	first_CN_gone = **FALSE**;
	Processing:
	IF ((($R^k > C^k$) **AND** (timer >= inter-CN delay))
	OR
	((timer >= inter-CN delay) **AND** (first_CN_gone)) {
	
	Identify Edge nodes that transmit over k;
	Transmit CN to identified Edge nodes;
	Reset timer;
	first_CN_gone = **NOT** (first_CN_gone);
	}

Fig. 7. Hybrid rate reduction method.

$$\mathrm{EB}^P = \min_{k \in P}(C^k - R^k). \tag{14}$$

This section describes two approaches for utilizing excess bandwidth.

3.2.1. Available bandwidth rate increase

In the first method, the edge node increases allocated rates of the flows proportionally to the amount of the excess bandwidth available on the path. This method is called *Available Bandwidth (AB) rate increase*. Assuming that link k is the link that contains the smallest amount of excess bandwidth, the edge nodes adjust allocated rate of the flow f using AB rate increase as follows:

$$R^f = \min\left(R^f + \mathrm{EB}^P \frac{b^f}{b^k}, B^f\right). \tag{15}$$

The AB rate increase distributes available bandwidth fairly quickly; however, when multiple flows compete for the same resource and receive results of the path probing at different times, the AB rate increase method may lead to unfair Bandwidth Distribution. Only the flow that probes the path first is guaranteed its fair share of the excess bandwidth. A flow that probes the path after another flow has already consumed its share of excess bandwidth may not be able to achieve its fair share. The probe reply messages generated after one of the competing flows, say flow F, has already consumed its share of the excess bandwidth, will report back an amount of excess bandwidth smaller than that observed by F. Thus, using the AB rate increase method, the flows that discover and consume excess bandwidth later may unfairly get a smaller share of resources.

3.2.2. MRR rate increase

To distribute excess resources fairly, each flow should increase its allocated rate proportionally to its MRR, and not to the amount of the excess bandwidth. We propose an alternative method for distribution of excess bandwidth called *MRR rate increase*. We define the MRR rate increase method as follows:

$$R^f = \min(R^f + \beta \times b^f, B^f) \tag{16}$$

Symbol β denotes a constant that determines how swiftly each flow increases its allocated rate. The MRR rate increase method guarantees a proper share of excess bandwidth for each flow, but its performance depends on the value of the constant β. Selecting constant β to be too small may lead to a slow convergence to the fair rates, while choosing constant β to be too large may lead to an unfair Bandwidth Distribution or even to congestion in the network. In general, the value of the constant should be inversely proportional to the number of flows competing for the excess bandwidth. However, this information is not available to the flows and may vary in each instance of congestion. We further examine performance of the rate increase methods through simulation.

3.2.3. Gradual rate increase optimization

As it was argued above, the AB rate increase method may occasionally lead to unfair resource distribution, while the MRR rate increase method may not be usable due to difficulty in finding the optimal value of β. To remedy these problems, the edge nodes, instead of distributing all of the excess bandwidth right after the probe message arrival, should gradually increase allocated rates of individual flows throughout the duration of the path probing period. For example, let us assume that an ingress node discovers excess bandwidth on the path and computes the fair share of excess bandwidth for its flows to be 60 Kbps. However, instead of allocating the entire 60 Kbps at once, the ingress node increases allocated rate of its flows multiple times by small amounts. For example, if the probe period is 6 s and the ingress node allows the rate increase every 100 ms, then its flows will receive $60\text{Kbps}\frac{100\text{ms}}{6000\text{ms}} = 1.0\text{Kbps}$ of excess bandwidth every 100 ms. This optimization is called *gradual rate increase*.

The gradual rate increase boosts the probability that the probing during the same probe period by different edge nodes will return similar results. When the edge nodes employ gradual rate increase optimization, the core routers observe a smaller arrival rate change over the same period of time as compared to when the edge nodes allocate all excess bandwidth at once. We examine the performance of the AB and MRR rate increase methods with and without gradual rate increase optimization in Section 4.3.

4. Evaluation of the BDS approach

The performance of the BDS approach was studied using the OPNET Network Simulator [12]. To simplify the notation, a flow that originates from source i is denoted as flow Fi. For example, the flow that originates from Source 1 is denoted as F1 and the flow of Source 2 as F2. Additionally, links Core 2–Core 5 and Core 5–Core 3 are denoted as c2–c5 and c5–c3, respectively.

4.1. Simulation set-up

Fig. 8 presents the network topology and the flow activation schedule used in our study. There are four heavyweight video flows in the network: F1, F2, F3, and F4 that activate at times 110 s, 210 s, 160 s, and 60 s, respectively. All the flows remain active until the end of the simulation,

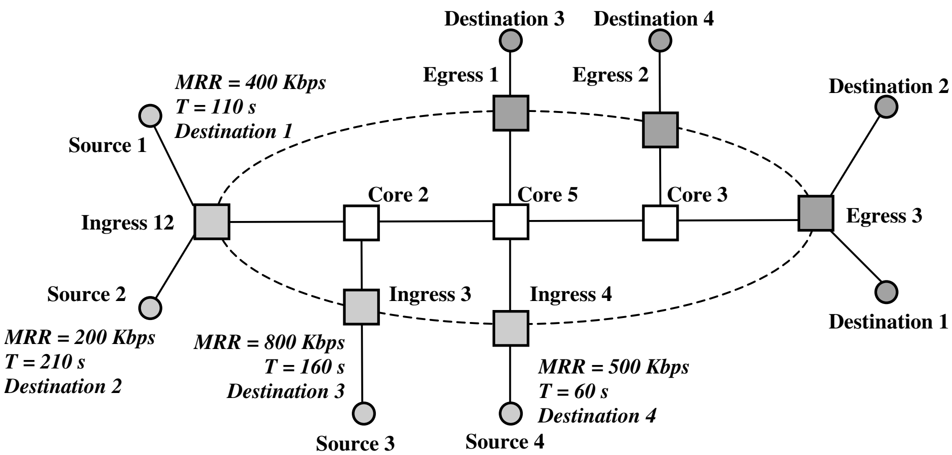

Fig. 8. Simulation topology.

Table 1
Optimal resource allocation for example of Fig. 4

Flow name	Flow allocated rates during the time periods			
	[60s, 110s]	[110s, 160s]	[160s, 210s]	[210s, 250s]
F1	0	672	504	432
F2	0	0	0	216
F3	0	0	1008	864
F4	1400	840	1008	864

which lasts 250 s. Flows F1, F2, F3, and F4 travel to destinations 1, 2, 3, and 4 and have minimum requested rates of 400 Kbps, 200 Kbps, 800 Kbps, and 500 Kbps, respectively. Each flow can transmit at maximum rate of 1400 Kbps and each link in the network is provisioned with 1513 Kbps of bandwidth (e.g., link capacity is 1544 Kbps, 98% of capacity is allocated for the BDS traffic).

Table 1 shows an optimal resource distribution for the scenario of Fig. 8. When flow F4 activates at time 60 s, it is the only flow in the network and it transmits data at the maximum rate of 1400 Kbps. At time 110 s flow F1 activates and causes congestion on link c5–c3. As a result, flows F1 and F4 adjust their allocated rates to 672 Kbps and 840 Kbps, respectively. At time 160 s flow F3 activates and causes congestion on link c2–c5, which becomes a new bottleneck for flow F1. After flows F1 and F3 adjust their allocated rates to 504 Kbps and 1008 Kbps, respectively, flow F4 observes excess bandwidth on the link c5–c3 and increases its allocated rate to 1008 Kbps. Finally, flow F2 activates at time 210 s causing congestion on links c2–c5 and c5–c3. At first the flows adjust allocated rates to their fair shares on the corresponding bottleneck links as follows: F1 gets 432 Kbps, F2 is allocated 216 Kbps, F3 acquires 688 Kbps, and F4 obtains 864 Kbps of available bandwidth. However, flow F3 observes that its bottleneck link is underutilized and increases its allocated rate to 864 Kbps.

4.2. Evaluation of resource management in the BDS

The simulation scenario of Fig. 8 was divided into four time periods based on the flow activation times. To evaluate performance of the BDS resource management mechanism, we examined resource allocation during each of these time periods. Table 1 shows optimal allocated rates of the flows during each time period. An optimal allocated rate of a flow is the amount of bandwidth the flow should receive according to the resource management mechanism defined in Section 2.4. To compare the performance of the proposed rate reduction and rate increase mechanisms, we define a new metric called *degree of fairness*. The degree of fairness at time t for flow f is computed as the ratio between the allocated rate $R^f(t)$ and the optimal allocated rate $R^f_{\mathrm{OPT}}(t)$, and is adjusted so that the degree of fairness values always fall between 0 and 1. The values of the allocated and the optimal allocated rates are always non-negative.

$$DF^f(t) = \begin{cases} 1 - \left|1 - \frac{R^f(t)}{R^f_{\mathrm{OPT}}(t)}\right|, & \text{if } \frac{R^f(t)}{R^f_{\mathrm{OPT}}(t)} < 2 \\ 0, & \text{otherwise} \end{cases}. \quad (17)$$

The degree of fairness is a statistic that shows how fairly the resources have been distributed at a particular point in time. High degree of fairness values (e.g., 0.95–1.0) correspond to a situation where resources are distributed fairly and the allocated rates of the flows converge to corresponding optimal rates. Small degree of fairness values correspond to a situation when resources are not distributed fairly and the allocated rates of the flows diverge from the corresponding optimal rates. To compare the performance of the rate reduction methods directly, the degrees of fairness of all the active flows during a particular time period are averaged. The value thus obtained is called the *average degree of fairness*.

4.3. Evaluation of the resource management mechanism

This section examines the performance of the rate reduction and rate increase mechanisms. More specifically, we compared the performance of the individual rate reduction methods, examined the influence of the inter-CN delay on the performance of the best rate reduction method, evaluated and compared the performance of the rate increase methods, and finally examined link utilization in the network. The simulations were configured with the MRR increase method and a probe period (e.g., delay between consecutive periodic probes) of 2 s. When using the proportional rate reduction method the inter-CN delay was set to 0.5 s, while the other methods were configured with the inter-CN delay of 2.0 s.

4.3.1. Evaluation of the rate reduction methods

Fig. 9 presents a comparison between the average degrees of fairness for each of the rate reduction methods. The "dips" in the degree of fairness curves correspond to the events of flow activation and subsequent initiation of the rate reduction process. Initially, all the rate reduction methods have an average degree of fairness close to 1, because only flow F4 is active in the network and as a result, the edge node Ingress 4 computes the initial allocated rate without the help of the resource management mechanism.

As Fig. 9 shows, the naïve rate reduction method performs on average the worst, because it distributes resources without estimating the aggregate MRR on the congested link. When using the simple two-step rate reduction method, the edge routers may adjust transmission rates so that congestion is eliminated after the first notification and as a result no second congestion notification is generated. Subsequently the edge nodes do not estimate the aggregate MRR on the congested link causing the simple two-step rate reduction method to perform as poorly as the naïve method. This event is illustrated during the time period [160, 210] when the average degree of fairness of the simple two-step rate reduction method is smaller than that of the naïve method.

Both the hybrid and the 2CN rate reduction methods were the best out of the four examined methods. However, the 2CN rate reduction method often requires slightly more time to converge than the other rate reduction methods. This happens because the 2CN rate reduction method goes through multiple rate reductions before it can compute an accurate enough value of the interface MRR. The hybrid method does not suffer from this deficiency because after the initial rate reduction it uses the proportional rate reduction method to adjust allocated rates of the flows. This phenomenon is clearly illustrated during time periods [110, 160] and [210, 250] seconds, where the 2CN method converges to optimal values much slower than the hybrid method. However, after the initial round of reduction, the average degree of fairness of the 2CN method was already above 0.9 even

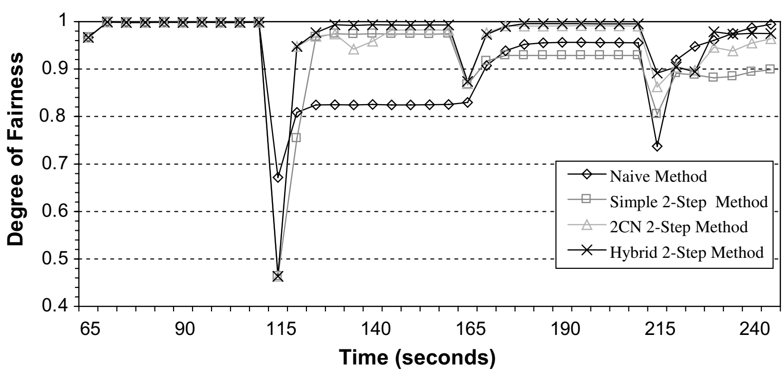

Fig. 9. Comparison of the rate reduction methods during [0, 250] time period.

though the method was still converging to optimal values.

4.3.2. Degree of fairness vs. inter-CN delay

Overall, the simulation results showed that the hybrid 2-step rate reduction method is the most stable and the most reliable of all rate reduction mechanisms. To further study the hybrid rate reduction method the inter-CN delay was varied. Figs. 10 and 11 show collected results of this study.

As Figs. 10 and 11 show, when configured with small inter-CN delay (e.g., 1.0–1.4 s), the hybrid rate reduction method often fails to converge to optimal resource distribution and estimates the interface MRR inaccurately. On the other hand, the hybrid rate reduction method performs much better when configured with larger inter-CN delay (e.g., 1.6–2.0 s). The inter-CN delay influences the accuracy of the reported arrival rate on a congested interface and thus impacts the accuracy of the estimated interface MRR.

Small value of the inter-CN delay causes the core routers to generate the second congestion notification message shortly after the first one. As a result, the results of the first rate reduction may not have propagated into the network core yet and the rate estimation mechanism may not have enough time to converge to the accurate values. This in turn causes the second congestion notification to report inaccurate values of the arrival rate on the congested link, resulting in the edge nodes computing the interface MRR incorrectly. On the other hand, large values of the inter-CN delay may unnecessarily suspend the rate reduction process, degrading the response time of the congestion control.

The inter-CN delay directly depends on the propagation delay between the edge nodes and a corresponding congested link and on the convergence time of the rate estimation mechanism, which is usually an order of magnitude larger than the propagation delay. Thus, the inter-CN delay

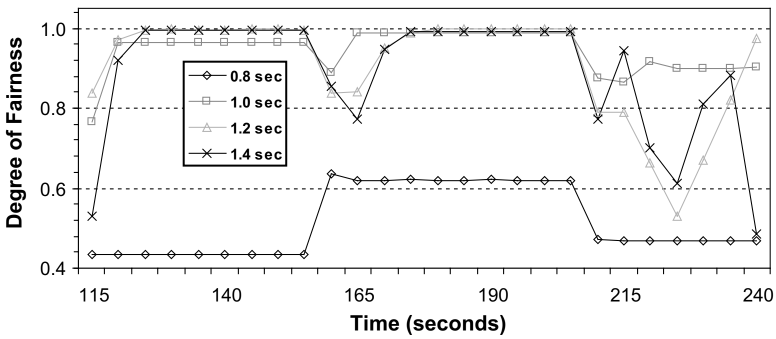

Fig. 10. Degree of fairness vs. inter-CN delay (0.8–1.4 s).

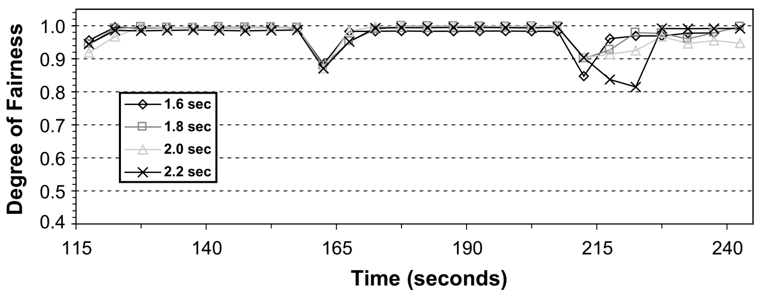

Fig. 11. Degree of fairness vs. inter-CN delay (1.6–2.2 s).

should be primarily computed based on the convergence time of the rate estimation mechanism. Overall, collected results suggest that the inter-CN delay should be large enough to allow the core nodes to accurately estimate arrival rate and at the same time it should be small enough so as not to slow down congestion control.

Evaluation of the rate reduction mechanism indicates that the performance of the BDS approach greatly depends on the precision of the rate estimation mechanism. Thus, a more accurate rate estimation mechanism in the network core may allow the boundary nodes to achieve an optimal resource distribution much faster.

4.3.3. Evaluation of the rate increase methods

To evaluate performance of the rate increase mechanism the scenario of Fig. 8 was slightly modified. The simulation was executed for 310 s instead of 250, and at time 250 s flows F1 and F4 were simultaneously terminated. As a result, flows F2 and F3 discover excess bandwidth on the paths to their respective destination and started increasing their allocated rates.

As expected, simulation results showed that the AB rate increase method distributes available bandwidth fast, but it causes the bandwidth allocation to be unfair. Since the path probing phases are not synchronized among individual edge nodes, they can report the characteristics (e.g., amount of excess bandwidth) of the same route to be different. As a result, the available bandwidth rate increase method is unable to distribute excess bandwidth fairly among individual flows because the path probing phase of each edge router reports a different amount of resources available on the path. On the other hand, the MRR rate increase always distributes available bandwidth fairly but it could be hard to deploy because of the difficulty selecting an optimal value of the MRR rate increase constant β.

Fig. 12 compares the average degree of fairness of the MRR and AB rate increase methods with and without the gradual rate increase optimization. The results presented in Fig. 12 were collected for the following network configuration: the MRR rate increase method constant β is set to 0.08, the edge nodes probe the network every 6 s and increase allocated rates of the flows every 100 ms. Since one of the main disadvantages of the MRR increase method is difficulty of finding the value of β we conducted a set of simulations that compared performance of the MRR rate increase method for different values of β. Simulation results suggested that the MRR rate increase method performs the best when $\beta = 0.08$.

As Fig. 12 shows, the AB gradual rate increase method performs better than all the other methods. It stabilizes at time 273 s and its average degree of fairness reaches 0.97. Although the average degree of fairness of both MRR rate increase methods also reaches 0.97, their convergence time is much longer. The MRR rate increase methods with and without the optimization complete their excess Bandwidth Distributions at times 283 and 288 s, respectively. Finally, the average degree of fairness of the AB rate increase method reaches only 0.91 which signifies

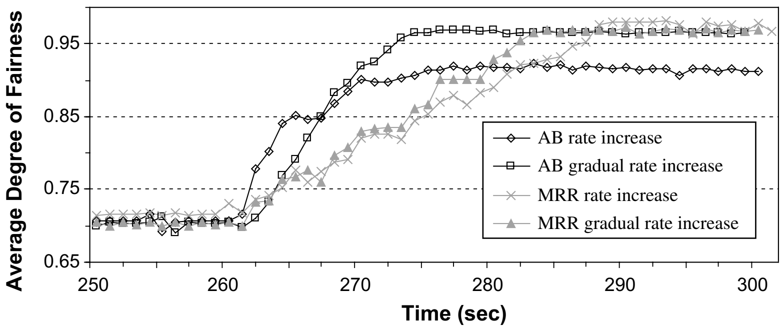

Fig. 12. Comparison of the rate increase methods.

that the AB method may fail to distribute excess bandwidth fairly without the gradual rate increase optimization. Thus, the AB gradual rate increase method combines the advantages of the AB and MRR rate increase methods and is able to fairly distribute excess bandwidth without the need to guess an optimal value for constant β.

4.3.4. Link utilization

Finally, let us examine performance of the resource management mechanism of the BDS approach in terms of the link utilization. Fig. 13 displays utilization of the bottleneck links c2–c5 and c5–c3 for the scenario of Fig. 8. As expected, during the time period [60, 110] seconds the bottleneck link c2–c5 is not fully utilized because only flow F4 is active in the network. Since flow F4 transmits at its maximum rate of 1.4 Mbps, which is smaller than the link capacity, link c2–c5 is not completely utilized. However, throughout the rest of the simulation the bottleneck links c2–c5 and c5–c3 are utilized close to 100%. Thus, these simulation results support our belief that the resource management mechanism, which consists of the rate reduction and rate increase mechanisms, tends to maximize throughput in the network.

4.4. Evaluation of congestion control

To evaluate the BDS congestion control mechanism, we examine the timetable of congestion occurrences using hybrid rate reduction method for the simulation scenario presented in Fig. 8 and Table 1. As it was mentioned in Section 4.1,

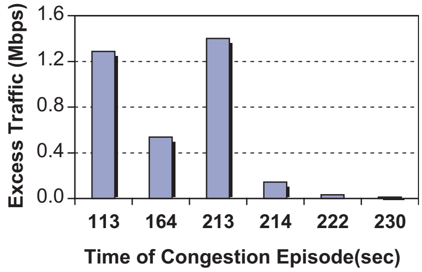

Fig. 14. Excess traffic during each congestion occurrence.

this simulation scenario has four distinct time periods determined by the schedule of flow activations, which are the primary cause of congestion. Fig. 14 identifies each congestion occurrence and shows the amount of excess traffic arriving at the congested link.

As Fig. 14 shows, congestion occurs only around flow activation time. More specifically, flow F1 enters the network at time 110 s causing congestion on link c5–c3. At around time 113 s, core router c3 identified congestion and notifies edge routers Ingress 12 and Ingress 4 that it has about 1287 Kbps of excess traffic arriving on link c5–c3. Similar situation occurs when flow F3 activates at time 160 s causing congestion on link c2–c5. At around time 164 s, core router c5 identifies congestion and notifies edge routers Ingress 12 and Ingress 3 that it has about 540 Kbps of excess traffic arriving on link c2–c5.

When flow F2 activates at time 210 s it causes congestion on links c2–c5 and c5–c3. However, since flow F2 first travels over link c2–c5 and only then over link c5–c3, router c5 estimates a larger

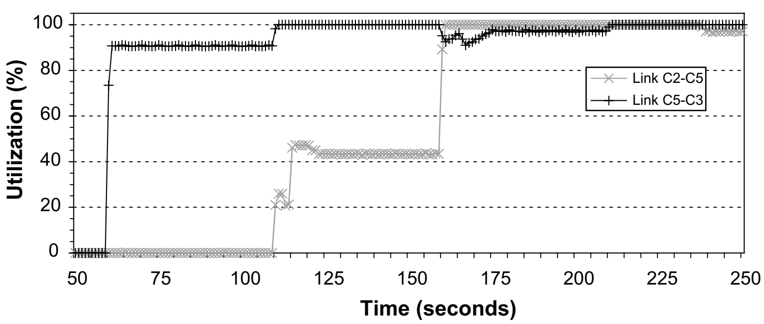

Fig. 13. Utilization of links c2–c5 and c5–c3.

amount of excess traffic on link c2–c5 than router c3 on link c5–c3. As a result, c5 identifies congestion and notifies corresponding edge routers sooner than router c3. Subsequent congestion occurrences that happen at times 222 and 230 s were the results of estimation errors. In both cases the amount of excess traffic was very small and it did not cause any noticeable performance degradation in the network.

We further examined the BDS congestion control mechanism by studying the correlations between the amount of excess traffic on the congested link and the time required by the core nodes to identify congestion. We also examined the time needed by the hybrid rate reduction method to respond to congestion. Fig. 15 presents the collected results.

Fig. 15(a) shows the time needed by the core routers to *identify* congestion while Fig. 15(b) shows the time to eliminate the *congestion*. As Fig. 15(a) shows, the time required by the core routers to identify congestion is inversely proportional to the amount of excess traffic on the congested link. Since congestion is a function of arrival rate, the time needed by the rate estimation mechanism of the core routers to identify congestion is influenced by the amount of excess traffic. Core routers need less time to identify congestion if the amount of excess traffic arriving on the congested link is large. On the other hand if the amount of excess traffic is fairly small, the core router may often require more time to identify congestion due to rate estimation errors. Clearly, the rate estimation mechanism is one of the primary parameters that influences response time of the BDS congestion control. Currently, we are examining alternative methods to estimate the arrival rate faster and more accurately.

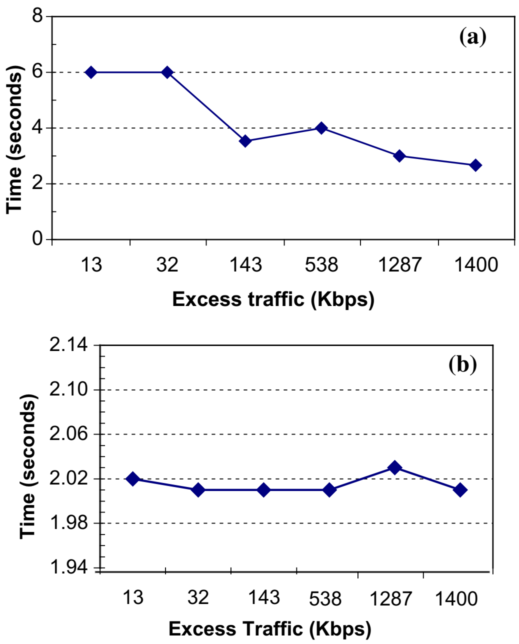

Fig. 15. Influence of excess traffic on BDS congestion control. (a) Excess traffic vs. time to *identify* congestion, (b) excess traffic vs. time to *eliminate* congestion.

As Fig. 15(b) shows, response time of the hybrid rate reduction method remains constant regardless of the severity of congestion. It takes slightly more than 2 s for the hybrid method to respond to congestion (e.g., to estimate the aggregate RBR on the congested link and to adjust transmission rates of individual flows accordingly). The 2-s delay required to eliminate congestion corresponds to the value of the simulation parameter inter-CN delay that was set to 2 s. Thus, based on the results presented in Fig. 7, if the core routers set the inter-CN delay to 1.6 s then they will still achieve fair resource distribution while eliminating congestion much faster.

In summary, our simulation results suggest that the BDS is capable of providing satisfactory congestion control. However, it may require a few seconds (dependent on the value of inter-CN delay) to fairly distribute available resources and to eliminate congestion in the network. Overall, the response time of congestion control and the BDS as a whole, directly depends on the effectiveness of the rate estimation mechanism (e.g., how fast the rate estimation mechanism converges) because it determines the value of the inter-CN delay. Thus, a more sophisticated rate estimation mechanism, that swiftly and accurately estimates the arrival rate, may significantly improve the congestion response time of the BDS approach. In addition, other methods that do not rely on the rate estimation algorithm for discovering congestion and can determine that the link became overloaded faster will significantly reduce the time re-

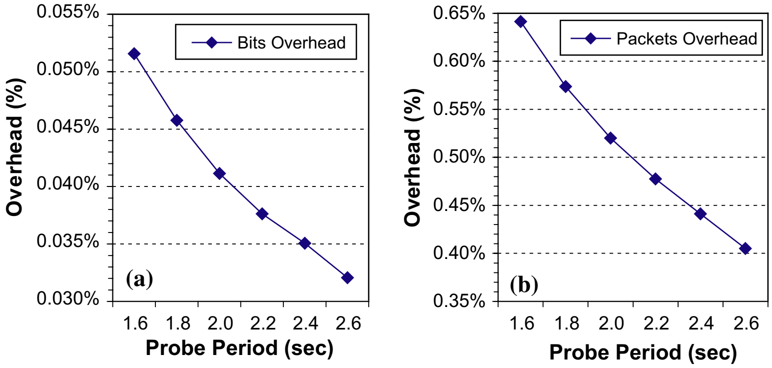

Fig. 16. The overhead of the RDF protocol. (a) Bits overhead, (b) packets overhead.

quired by the BDS approach to eliminate congestion.

4.5. Control load overhead

The control load overhead caused by the RDF protocol is the ratio between the amount of control data and the total amount of data generated in the system. The overhead of the Bandwidth Distribution Scheme was computed in terms of the number of packets and the number of bits generated in the system. Fig. 16 shows how the overhead varies with the change of the path probing period.

Overall, for a simple scenario of Fig. 8 the RDF protocol performs well and does not incur a noticeable amount of overhead. However, to draw conclusions about the RDF protocol's overhead, a further study of the BDS model under more realistic network conditions is needed.

5. 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 of these 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 provides 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.

The Bandwidth Distribution Scheme (BDS) described in this paper attempts to combine the advantages of the Integrated and Differentiated Services models by providing a framework for scalable support of per-flow bandwidth guarantees. The edge nodes in the BDS network manage per-flow information while the core routers deal only with the traffic aggregates. This paper introduces a novel approach for approximating aggregate flow requirements and then using this information for distributing available bandwidth among individual flows. The BDS employs a combination of periodic probing and explicit network feedback to estimate the aggregate information which is used by the edge nodes for per-flow bandwidth management and congestion control.

The idea of using explicit network feedback for dynamic rate and congestion control is not new. In particular, the Explicit Congestion Notification (ECN) extension to IP [16] 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 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, the BDS model does not store per-flow information in the network core and relies on arrival rate feedback to estimate fair shares of individual flows. 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 [9]. 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 [14] 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 [15]. The routers use the DPS information to provide per-flow guarantees without maintaining per-flow state in the network core. The main difference between the BDS and Stateless-Core models is in the mechanisms used to distribute resources among individual flows. In the Stateless-Core approach, the core routers provide per-flow rate allocation via a FIFO queue with probabilistic drop, where the probability of dropping a packet is a function of the estimated rate carried in the packet's header and the fair share at that router which is estimated based on measurements of the aggregate traffic [14]. Such approach requires additional processing of each data packet at the core routers and causes wasteful usage of network resources by the packets that travel through the network but are dropped before reaching their destination. On the other hand, the BDS model adjusts transmission rates at the network edges, which avoids these deficiencies. However, adjustment of per-flow rates in the core has an advantage of not being subject to the propagation delay of the notification messages and the delay required to estimate the aggregate requirements as in the BDS model. Another disadvantage of the Stateless-Core model is its inability to distribute excess bandwidth due to use of the upper bound of the aggregate reservation for admission control, which could possibly lead to network underutilization [14]. On the contrary, the BDS approach fairly distributes excess bandwidth and maximizes network throughput. However, current version of the BDS model does not have the admission control which could lead to violation of per-flow guarantees in under-provisioned networks.

Similarly to the simple rate control algorithm [9], the BDS approach can be used to support bandwidth guarantees and is most suitable for elastic sources that can tolerate and benefit from frequent changes of the allocated rates. FTP and video flows are examples of such elastic traffic. The main advantages of the BDS approach are its ability to eliminate congestion while maintaining high link utilization, fairly distributing available resources in the event of congestion, and fair sharing of the leftover excess bandwidth as shown in Section 4.2. Another significant advantage of the BDS approach is its simplicity. The RDF protocol used in the BDS appears not to cause scalability concerns and does not incur significant overhead as shown in Sections 2.6 and 4.2, respectively.

However, simplicity of the BDS comes at the cost of slow convergence to optimal resource distribution as shown in Section 4.2. In addition, the current version of the BDS architecture does not have admission control and all flows that request to enter the network are admitted. The lack of admission control reduces processing complexity at the edge routers but it may lead to violation of bandwidth guarantees, especially in under-provisioned networks. To implement admission control within the BDS architecture, the edge nodes should possess knowledge of the interface MRR values on the new flow's path. However, this information is not maintained anywhere in the network and can be obtained only during congestion and only if the edge router has traffic traveling on the congested path. Thus, the edge nodes that have no traffic traveling on a particular path may have no information about interface MRR values on that path, which makes addition of admission control mechanism to the BDS approach a difficult problem. We examined a modification of the BDS architecture that allows introduction of admission control in [7,8].

Another important issue is the stability of the BDS approach. We consider the BDS network to be stable if it can return to its steady state (e.g., all flows in the network are allocated their fair shares of bandwidth) after such network events as congestion or presence of excess bandwidth. Currently we are investigating the problem of stability of the BDS approach. Table 2 provides a summary of the BDS characteristics.

In this paper we made a number of assumptions about the BDS network environment. In particular, we assumed that the RDF protocol is reliable and that the control packets cannot be lost. However, we are planning to examine this issue (e.g., how the control packet loss influences the BDS performance) in future work. In addition, we assumed that the underlying routing protocol takes care of all the routing issues, does not cause routing loops, and that the routes in the network remain static. Currently, we are studying the BDS performance in the network where these assumptions are eliminated.

6. Conclusions and future work

In this paper we presented a novel approach for estimation of the aggregate resource requirements that enables support of scalable, per-flow, dynamic resource distribution. The Bandwidth Distribution Scheme is capable of eliminating congestion while maximizing network throughput. In this paper, we presented and evaluated four techniques for fair distribution of available resources among individual flows during congestion. Furthermore, we examined two mechanisms that allow competing flows to fairly share leftover excess bandwidth. However, we also discovered that in the absence of congestion, the BDS is unable to fairly distribute resources among individual flows. In addition, the lack of admission control makes it difficult to

Table 2
Summary of the BDS characteristics

Advantages	Disadvantages and improvements
Simplicity: relies on simple RDF protocol and simple processing in the network core	*No admission control*: no minimum rate per-flow guarantees in current version
Congestion control: eliminates congestion in the network	*Inefficient*: slowly converges to optimal resource distribution
Maximizes throughput: keeps the bottlenecks link in the network completely utilized	*Rate estimation*: a more sophisticated rate estimation mechanism may improve overall performance
Resource management: provides fair resource distribution in the event of congestion	*Alternative means for congestion detection*: can reduce the time required to eliminate congestion
Low overhead: the path probing phase of the RDF protocol causes a low overhead	*Stability*: stability of the BDS approach is being investigated

extend the BDS approach to support absolute bandwidth guarantees.

However, despite these deficiencies, our study of the BDS approach provided a valuable insight into the problem of dynamic per-flow resource distribution. Furthermore, based on this study we believe that the BDS architecture with addition of a mechanism that delivers the aggregate RBR under all network conditions can provide support for scalable per-flow QoS through fair, dynamic resource distribution implemented at the network boundaries. Currently, we are investigating a modification of the presented BDS model, where the aggregate resource requirements are explicitly maintained in the network core. Such a modification allows the edge routers to implement admission control and to distribute available resources under all network conditions [7,8]. In addition we are examining the problem of BDS deployment into the Internet, and in particular, the issue of dealing with traffic that travels through multiple network domains. We are also examining the issue of the BDS influence on TCP traffic and planning to extend the BDS framework to a mobile environment.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, 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] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource reservation protocol (RSVP)—version 1 functional specification, September 1997, IETF RFC 2205.

[4] D. Clark, W. Fang, Explicit allocation of best effort packet delivery service, IEEE/ACM Transactions on Networking 6 (4) (1998) 362–373.

[5] W. Fang, N. Seddigh, B. Nandy, A time sliding window three colour marker (TSWTCM), June 2000, IETF RFC 2859.

[6] W. Feng, D. Kandlur, D. Saha, K. Shin, Understanding and improving TCP performance over networks with minimum rate guarantees, IEEE/ACM Transactions on Networking 7 (2) (1999) 173–187.

[7] V. Hnatyshin, A.S. Sethi, Reducing load distribution overhead with message aggregation, in: Proceedings of the 22nd IEEE International Performance, Computing, and Communications Conference, April 2003, pp. 227–234.

[8] V. Hnatyshin, A.S. Sethi, Fair and scalable load distribution in the Internet, in: Proceedings of the International Conference on Internet Computing, June 2002, pp. 201–209.

[9] K. Kar, S. Sarkar, L. Tassiulas, A simple rate control algorithm for maximizing total user utility, in: Proceedings IEEE INFOCOM 2001, April 2001.

[10] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, B. Vandalore, The ERICA switch algorithm for ABR traffic management in ATM networks, IEEE/ACM Transactions on Networking 8 (1) (2000) 87–98.

[11] D. Katabi, M. Handley, C. Rohrs, Internet congestion control for future high bandwidth-delay product environments, in: Proceedings ACM SIGCOMM, Pittsburgh, PA, August 2002.

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

[13] N. Seddigh, B. Nandy, P. Pieda, bandwidth assurance issues for TCP flows in a differentiated services network, in: Proceedings GLOBECOM'99, December 1999.

[14] I. Stoica, Stateless Core: a scalable approach for quality of service in the Internet, Ph.D. Thesis, Carnegie Mellon University, 2000.

[15] I. Stoica, H. Zhang, Providing guaranteed services without per-flow management, in: Proceedings ACM SIGCOMM, September 1999.

[16] K. Ramakrishnan, S. Floyd, D. Black, The addition of explicit congestion notification (ECN) to IP, September 2001, IETF RFC 3168.

Vasil Y. Hnatyshin is an Assistant Professor in the Computer Science Department at Rowan University, Glassboro, New Jersey, USA. He started his education at L'viv State University, L'viv, Ukraine. He received his B.S. (summa cum laude) from Widener University, Pennsylvania, and M.S. and Ph.D. in Computer Science from University of Delaware, Delaware. His research interests include service differentiation, resource management, and quality-of-service in IP networks, as well as network management, BGP and inter-domain routing, mobile ad hoc networks, and wireless technologies.

Adarshpal S. Sethi is a Professor in the Department of Computer and Information Sciences at the University of Delaware, Newark, Delaware, USA. He has an MS in Electrical Engineering and a Ph.D. in Computer Science, both from the Indian Institute of Technology, Kanpur, India. He has served on the faculty at IIT Kanpur, was a visiting faculty at Washington State University, Pullman, WA, and Visiting Scientist at IBM Research Laboratories, Zurich, Switzerland, and at the US Army Research Laboratory, Aberdeen, MD. He is on the Editorial Advisory Board for the *Journal of Network and Systems Management*, and on the editorial boards of *eTNSM (IEEE electronic Transactions on Network and Service Man-*

agement), International Journal of Network Management, and *Electronic Commerce Research Journal*. He was co-Chair of the Program Committee for ISINM '95, and was General and Program Chair for DSOM '98; he is also active on the program committees of numerous conferences. His research interests include architectures and protocols for network management, fault management, quality-of-service and resource management, and management of wireless networks.