

A Premium Traffic Management in DiffServ Through End-to-End delay Budget Partitioning

Hamada Alshaer¹ and Eric Horlait²

¹Department of Electronic and Computer Engineering, Brunel University, London, UK

² QoS MoS Company, 75019 Paris, France

Email: hamada.alshaer@brunel.ac.uk, eric.horlait@qosmos.com

Abstract—Our novel Call Admission Control (CAC) scheme [1], [2] determines the number of Expedited Forwarding (EF) (Premium) flows accepted, based on e2e available bandwidth measurement. However, it does not control the number of flows that could be accepted at a node or over a link along a selected routing path for an EF flow. Hence we complemented the functionality of CAC with original traffic management algorithms [2], [3] to balance EF traffic along the selected routing path for new EF flows and consequently across Differentiated Services (DiffServ) network. The novelty of these algorithms emerges from the fact that they employ an optimization dimension of EF traffic, namely the partitioning of End-to-End (e2e) delay required for a new flow into per-link or per-DiffServ node along the selected routing path. Furthermore, some network resource policies are incorporated with these algorithms to apportion the e2e resource required for EF traffic flows into per-DiffServ node requirement along the selected routing path. We demonstrate through analysis and simulation that our proposed optimal e2e delay budget management algorithms are superior to Equi-Partitioning (EP) algorithm for DiffServ in guaranteeing e2e delay bounds required for EF flows. Furthermore, highly loaded DiffServ nodes are assigned low delay budget and the opposite for low loaded DiffServ nodes, which result in increasing DiffServ network resource utilization.

Index Terms—DiffServ, E2E delay budget partitioning, Resource partitioning policies, EF load balancing

I. INTRODUCTION

The DiffServ architecture [4], [5] is currently considered as a promising IP QoS solution due to its scalability and ease of implementation in Internet switches, routers, and traffic routing mechanisms. End-to-end (e2e) differentiated services are achieved by concatenation of per-domain services and Service Level Agreements (SLAs) between adjoining domains along the different routes that the traffic traverses from the source to destination. The per-domain services are realized through providing qualitative Per-Hop Behavior (PHB), by employing appropriate multiservice scheduling and efficient resource management algorithms in the core routers. Two PHBs are standardized by IETF: Expedited Forwarding (Premium) PHB (EFPHB) [5], [6] to implement services requiring

low delay/jitter and low losses; and Assured Forwarding PHB (AFPHBs) [7], [8] to implement services requiring assured bandwidth. The premium service can offer a user a performance level that is similar to that of a leased line, as long as the user's traffic is limited to a given bandwidth [4].

However, providing strict network resources, precisely bandwidth to packet flows in the Internet is inherently challenging task. It requires signaling mechanisms, policing mechanisms, as well as, accurate and rapid accounting of network resources, and call admission control. Many real-time applications; such as, VoIP, video or industrial control systems demand efficient and effective communication services. By real time traffic, we mean that a packet belonging to this traffic is delivered from its sources to its destination within a predefined e2e deadline. Packets delivered beyond these e2e deadlines are considered useless. To support the QoS requirements of real time traffic, the Internet Service Providers (ISPs) need to dedicate part of network resources for each customer. This creates a challenging problem for every service provider, which is how to maximize the utilization efficiency of his physical network infrastructure and still support heterogeneous QoS requirements of different customers. Telephone service providers can ask almost the same question: How can their IP telephony operator working on underlying DiffServ network distributes EF traffic flows over the different network routing paths and nodes, so that all these flows are offered similar QoS guarantees, while satisfying other traffic classes' QoS requirements.

The direct answer on such challenging problem is to employ traffic engineering methods imbedded in MPLS techniques. These select explicit routes for LSPs between a given source and destination. Each LSP could act as a traffic trunk carrying an aggregate traffic flow that requires QoS guarantees; such as, delay, jitter and loss bounds. The key approach underlying traffic engineering algorithms, such as [9], [10], is to select routing paths so as to balance the traffic loads on the network links and nodes. Meanwhile real time traffic load could be optimally balanced across a DiffServ domain, by employing optimal schemes for partitioning their required e2e delay along the links or nodes constituting a selected routing path for EF flows accepted.

This paper is based on "Expedited Forwarding End-to-End Delay Budget Partitioning," by H. Alshaer and E. Horlait, which appeared in the Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA, January 2006. © 2006 IEEE.

The DiffServ server (i.e., edge or core router) must determine appropriate delay bounds for the different traffic classes, which necessitates a strong coupling relationship between the scheduling mechanisms and management algorithms employed in network. Furthermore, network resources must be managed on e2e basis to guarantee network performance, particularly e2e QoS offered to EF traffic. Scheduling Service Disciplines (*SSDs*) and resource management algorithms operate together in order to provide qualitative per-hop behavior that corresponds to the packets' DSCP. These algorithms also require robust mechanisms for performing call admission control and resource reservation, which react according to determined control policies like those related to the EF QoS guarantees and EF traffic balancing across the network. The link sharing policies imbedded in the higher level resource management mechanisms employed in DiffServ network allow multiple organizations to have access to a guaranteed share of the link bandwidth during their activities over the network. Furthermore, these policies represent the basic rules according to which penalty boxing or call admission algorithms employed at the ingress routers can detect or control, respectively unresponsive and high bandwidth flows in terms of congestion, QoS violation or degradation. Thus creating incentives for the use of traffic imbalance, e2e congestion control procedure and e2e QoS guarantees.

The ability to efficiently manage DiffServ network resources is critical in order to effectively address e2e QoS guarantees required for EF traffic. Networks must determine appropriate delay bounds, for each core router within the networks, in order to guarantee the e2e delay required for EF flows. Furthermore, ingress routers must be able to re-negotiate delays, when, due to node failure along the flow's routing path, the network is no longer able to continue supporting the EF service. The direct implication of DiffServ network design choice is the ability to implement a wide variety of service behaviors. Different *SSDs* can be used to enforce a desired per-hop behaviors for different classes of services, particularly to optimally manage the output link capacity or to partition efficiently the bandwidth among the different supported traffic classes.

In this paper, we investigate this by proposing first some policies for partitioning network resources on e2e basis, to guarantee the e2e QoS required for EF traffic. These are incorporated with optimal e2e delay budget partitioning algorithms to quantify the advantage for DiffServ network dimensioning of having a non-uniform allocation of e2e delay requirement over the links and nodes constituting a routing path. Whenever a new EF flow is accepted by CAC scheme [1], [2], our proposed management algorithms [3], [11]: Load Based Delay Slack Partitioning (*LB_DSB*), and Load balancing Based e2e Delay Partitioning (*LB_EDP*) algorithms balance EF traffic along the selected routing path for this new flow. The *LB_DSB* algorithm apportions the slack in e2e delay of this flow into per-link delay along the selected

routing path. It is supposed that rate-based scheduling service discipline serves EF traffic packets at DiffServ nodes along their routing path. In contrast, the *LB_EDP* algorithm apportions the entire e2e delay requirement of this flow into per-DiffServ node along the selected routing path. It is supposed that delay-based scheduling service discipline serves EF traffic packets along their routing path.

The remainder of this paper is organized as follows. In section II, we introduce the different *SSDs*, and analyze their impact on delay budget. In section III, we discuss and analyze the previous work. In section IV, we define EF traffic model and network topology. We formalize also the problem that will be treated throughout the paper. In section V, we derive different network resource partitioning policies for EF aggregate traffic. These map e2e resource required for an EF connection into per-DiffServ hop service rate along its routing path. In section VI, we present a novel e2e delay budget partitioning algorithm, which is built on rate-based scheduling. In section VII, we define an EF workload model. Based on this model, we derive in section VIII the low and maximum delay values that can be supported by a DiffServ node. In section IX, we present another novel e2e delay budget partitioning algorithm, which is built on delay-based scheduling. In section X, we present the characteristics of the simulation environment, and show some results. Finally in section XI, we will draw a conclusion to this paper.

II. MULTISERVICE SCHEDULING DISCIPLINES

Due to the strong coupling relationship between our proposed management algorithms and *SSDs* employed in DiffServ network: in this section we introduce a brief and oriented survey about the different scheduling service disciplines, which might be employed to guarantee the e2e QoS, namely e2e delay required for EF traffic.

MultiService Scheduling Disciplines (*MSSDs*) are responsible for QoS differentiation, control and guarantee required by the different traffic classes enabled in DiffServ network. Since an *SSD* determines a service order of packets belonging to the different flows, it has a direct control on how the core routers in DiffServ network offer service to the different traffic classes. Due to the diversity of traffic characteristics and its QoS requirements which must be realized in this network, more sophisticated scheduling service disciplines than the simple FIFO or FCFS *SSD* must be employed. In other words, advanced and intelligent scheduling algorithms are required to prioritize users' traffic to meet various QoS requirements while fully utilizing network resources. Scheduling service disciplines are typically evaluated along several dimensions, such as tightness of delay bounds, achievable network utilization, fairness and protection, protocol overhead and robustness.

We classify *MSSDs* into two categories: proportional differentiation based scheduling, and link sharing based scheduling which we further classify into three classes of

scheduling policies, namely fair queuing-based, deadline-based and delay-based, and rate based scheduling. Fair Queuing (*FQ*) allocates bandwidth proportionally to active or backlogged flows based on their weights [12]. The deadline-based scheduling policy dynamically assigns priorities to packets based on their deadlines. The packet with the closest deadline is assigned the highest priority. This policy has been shown to have a maximum schedulability region for a set of delay vectors [13]–[15]. The delay based-scheduling assigns priorities to flows so that flows with shorter delays have higher priorities than those with longer delays. It has been shown that this policy is optimal among fixed-priority scheduling algorithms [16]. Rate based scheduling [17] is a general framework, which consists of two main components: a regulator and a scheduling discipline. The regulator determines the eligibility time for each packet, and once a packet becomes eligible, the scheduler may select this packet for transmission. It is shown in [17], that this policy handles effectively the delay jitter of real time flows. The MSSDs function as work conserving and non-work conserving scheduling mechanisms. In work conserving mechanisms, an SSD becomes idle only when there are no packets waiting for transmission. However, in non-work conserving mechanisms, the SSD transmits a packet only when the packet is eligible for transmission, so it may become idle even when there are packets waiting for service.

III. RELATED WORK

Our aim in this paper is not to propose a new SSD. However, we would rather address the problem of how to efficiently partition the e2e delay required for EF traffic in DiffServ network, over the different links and nodes constituting a routing path. Computing the e2e delay bounds for all affected EF flows on the new flow’s route at each connection setup time is tedious work and time consuming if the network has a large number of Ingress-Egress ($I - E$) flows. This problem can be solved by partitioning the e2e delay requirement into individual link delay budget assignments. A new connection can then be admitted only when the link delay budget guarantees on a routing path are satisfied and this can be checked locally. The e2e delay budget partitioning problem is to find the delay budget link or node allocation that is best according to some relevant objective function or optimal capacity assignment.

The conclusion from what have been discussed so far and aforementioned QoS mechanisms is that a DiffServ hop can support a feasible range of delay values. These values fall in a delay interval starts from the minimum or low delay bound value (d^{lf}), and terminates at the maximum or upper delay bound value (d^{uf}) that the hop can support and real-time applications tolerate (we will see later how we can drive these values). When the e2e delay requirement D_{new}^{req} of a new flow is below the e2e delay of the routing paths connecting its ingress to a closest egress router to its destination, then this

flow is simply rejected. If the D_{new}^{req} of a new flow is greater than the e2e delay of the routing paths connecting its ingress to the egress connecting to its destination, then the new flow may be accepted with probability of e2e delay bound violation equivalent to $(\frac{\sum_{j=1}^M d_j^{uf}}{D_{new}^{req}})^+$. However, when D_{new}^{req} of a new flow satisfies the following condition: $\sum_{j=1}^M d_j^{lf} \leq D_{new}^{req} \leq \sum_{j=1}^M d_j^{uf}$, then there must be a mechanism to distribute properly the total remained delay (i.e., $D_{new}^{req} - \sum_{j=1}^M d_j^{lf}$) over the different M-links or nodes constituting the selected routing path.

The motivation for local allocation approach was discussed extensively in [18]. There has also been a large amount of work on optimal partitioning of e2e QoS budget into local constraints along a path [19]–[21]. In these references, they mainly focus on finding an efficient QoS budget allocation, where the performance metric is considered efficient if the optimal algorithms bring any substantial gain in the network performance. A comparison study done in [18] to evaluate the optimal and near optimal allocation schemes. In [22], the authors examined how to optimally allocate e2e average delay requirements to the links traversed by (σ, ρ, π) -regulated flows. Furthermore, they compared the equal and optimal partitioning with envelope-regulated traffic for a tandem network. However, their proposed optimal partitioning scheme is based on FIFO scheduling discipline, so they considered only one measure of delay for all the flows traversing a routing path.

In [23], the authors provided an elaborate analysis about the e2e statistical performance guarantees given to the traffic generated by sources modeled with a family of bounding interval-dependent random variables, and served by rate controlled service discipline inside the network. However, their work did not address the issue of how to partition the e2e delay requirements into link-delay budgets. In [24], the authors introduced a concept of real-time channel to guarantee a real-time service performance. It is defined as a simplex connection between a source and a destination characterized by parameters representing the performance requirements of a user generating a connection request. They studied the feasibility of providing real-time services on a packet-switched store-and-forward and a modified Earliest Deadline First (EDF) wide-area network with general topology. They describe a scheme for the establishment of channels with deterministic or statistical delay bounds [23], [25], [26], channels with a variety of offered load and burstiness characteristics. A source-destination path goes through a number of nodes where the transmitted information can be stored and then forwarded to the next node. Some of the links between adjacent store-and-forward nodes may actually be non-store-and-forward networks, provided their total delay can be bounded. Thus, the only assumption they made about the links between store-and-forward nodes is that there be a known and finite bound for the link delay of each packet.

In [27], the authors presented an allocation scheme using different traffic models and pointing out precisely where changes have to be made to the e2e QoS allocation and mapping design problem. They also presented a decomposition property of the dimensioning problem when the packet queues are M/M/1. In that case, they found out that the optimal delay budget partitioning can be done independently of traffic matrices and routing. They solved a nonlinear program and showed the advantage of the optimal partitioning over the equal partitioning with various effective bandwidth models in network dimensioning. In [28], the authors proposed Proportional Partitioning (PP) approach. They considered resource reservation for multicast flows and partition of e2e QoS over links of a multicast tree in proportion to the utilization of each link. They were developed within an admission control framework. The performance of PP was shown to be better than Equal Partitioning (EP) since it accounts for different link loads.

If an equal/proportional partition result in tighter delay requirements than the minimum possible at some link, then the proposal in [28] assigns the minimum delay at that link and performs equal/proportional allocation over remaining links. With such an approach, the minimum delay assignment converts the corresponding link into a bottleneck, disabling all the paths that contain it. In contrast, in [13], the authors partition the slack in e2e delay instead of the entire e2e delay, which helps to prevent the formation of bottleneck links as long as non-zero slack is available. Well-thought of algorithms to partition e2e QoS requirements of a unicast or multicast flow into per-link QoS requirements have been proposed in [10], [19], [29]. The optimization criteria is to minimize a global cost function which is the sum of local link costs. The cost functions are assumed to be weekly convex in [19] and increase with the severity of QoS requirements at the link costs in which each link offers only a discrete number of QoS guarantees and costs. For the algorithms in [10], [19], [29], [30] to be effective, one needs to carefully devise a per-link cost function that accurately captures the global optimization objective.

In conclusion, we have two mechanisms of e2e QoS budget partitioning: equal and optimal or proportional partitioning mechanism. Our objective is to maximize the number of real-time flows accepted in a network. Here, we aim at admitting more EF flows in DiffServ network, provided that their QoS requirements are satisfied. Furthermore, EF traffic load is balanced across the network, so other traffic aggregates can also traverse the network without any QoS degradation due to EF traffic. In the following section we define our system model, including EF traffic model and network topology which will be used later as a simulation environment, for evaluating our proposed optimal delay budget partitioning algorithms.

IV. SYSTEM MODEL

We consider a discrete time system serving flows within a class using a Weight Fair Queuing(WFQ) service

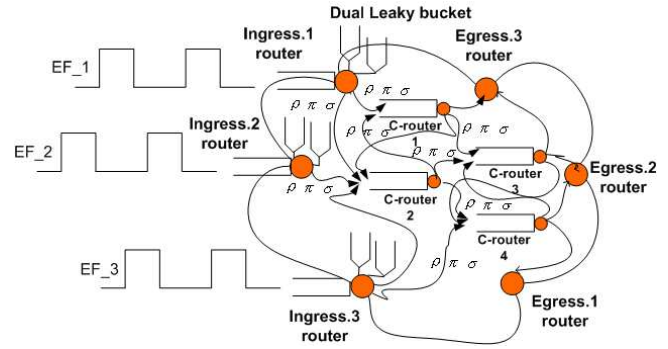


Figure 1. EF traffic model in DiffServ network.

discipline. A path is defined as a set of links and nodes connecting the source to the destination.

A. definitions

We provide the following notations that will be used throughout the text. The corresponding symbols can be found in Fig. 1 and Fig 2.

- 1) ρ := data rate
- 2) r_k^{avg} := average bandwidth required for k^{th} flow
- 3) σ := burst size
- 4) π := data Peak rate
- 5) L_k := EF packet size, belonging to k^{th} flow
- 6) L_{max} := maximum best effort (BE) packet size
- 7) N := number of active flows
- 8) M := number of DiffServ nodes
- 9) B_{EF} := reserved buffer for EF aggregate
- 10) C_{EF} :=reserved bandwidth for EF aggregate

B. Expedited Forwarding Traffic Model

As shown in Fig. 1, we suppose that EF traffic is generated by a number of deterministic ON-OFF sources. These generate traffic with peak rate π , which is described by a succession of cycles, each such cycle comprising an ON-period followed by an OFF-period. During the ON-periods the sources are active and produce traffic at the constant rate of π (packet/slot)(for simplicity we set this rate to the unity, say one packet/slot, i.e., $\pi = 1$); the sources are silent during the OFF-periods: For each $n = 0, 1, \dots$, let O_n and F_n denote the durations in time slots of the ON-period and OFF-period in the $(n + 1)^{st}$ cycle, respectively. An independent ON-OFF source is one for which, first the $\{1, 2, \dots\}$ -valued rvs $\{F_n, n = 1, \dots\}$ and $\{O_n, n = 1, \dots\}$ are mutually independent rvs which are independent of the pair of rvs F_0 and O_0 associated with the initial cycle. Second the rvs $\{F_n, n = 1, \dots\}$ and $\{O_n, n = 1, \dots\}$ are i.i.d rvs with generic OFF-period duration rv F and ON-period duration rv O . Throughout the generic rvs, O and F are assumed to be independent $\{1, 2, \dots\}$ -valued rvs such that $0 < E[O], E[F] < \infty$, and we simply refer to the independent ON-OFF process just defined as the ON-OFF source(O, F).

As long as the periods of the sources relate to rational numbers, the resulting queue is periodic. The only random

effect is the phase of the source; the durations of the activity periods and silence themselves are not random. EF traffic is regulated through a Dual Laky Bucket (DLB) installed at the different ingress routers. Thus (ρ, π, σ) -regulated EF flows arrive at the core routers. The DiffServ server is modeled with an input buffer B_{EF} and constant depletion rate C_{EF} reserved for EF aggregate. We assume that all the traffic flows are homogeneous and regulated with the same parameters (ρ, π, σ) , where π is the traffic peak rate. Notice that the only random effect is the phase of the sources: The start of the ON time is uniformly distributed on the period of the source.

C. Network Model And Problem Identification

As shown in Fig. 2, we use WFQ SSD to support the different traffic aggregates in DiffServ network. Let F_k^{EF} represents the packets belonging to k^{th} EF flow among other active flows in the EF traffic aggregate in the network. The delay bound of these flows at each link is inversely proportional to their bandwidth reservation. Thus, the amount of delay budget allocated to F_k^{EF} at a link determines the amount of bandwidth reservation that F_k^{EF} requires at that link. In Fig. 2, F_k^{EF} is defined as a traffic aggregate that carries traffic with an average bandwidth of r_k^{avg} and burst size σ_k . Given two requests from EF sou_{1,2} for setting up new EF flows at edges_{1,2}, respectively. Suppose that both of edges_{1,2} selected the same routing path between them and edge₄ to route accepted new flows until their destinations EF Dest_{1,12}, respectively. It is very possible that resource reservation over some links constituting this routing path runs out much earlier than others, thus rendering the entire network routing path unusable. Consequently, our problem can be stated as follows: How do the edge routers_{1,2} as other edges partition the e2e resource requirements into per-link resource requirement. Furthermore, how do these

routers partition the e2e delay D_k^{EF} requirement into per-link delay budgets such that (1) the e2e delay D_k^{EF} requirement is satisfied, (2) the amount of traffic admitted along the routing path can be maximized in the long-term? and (3) The deviation in the EF traffic loads on the different network links is as small as possible. In the following section, we solve the first part of the problem, by proposing some network policies for determining the required bandwidth at a DiffServ server, so that the local delay bound is met and hence e2e delay bound is satisfied.

V. RESOURCE PARTITIONING POLICES

EF PHB is defined as a forwarding treatment for a DiffServ aggregate such as the rate at which EF traffic is served at a given output interface should be at least the configured rate R , over a suitably defined interval, independent of the offered load of non-EF traffic to that interface [5], [6]. However, this condition is almost difficult to satisfy in the Internet network. As a result, queues are spontaneously formed which certainly cause high delay and non negligible jitter to traversing traffic in this network. Based on this configuration definition [5], [6] of EF service class, F_k^{EF} is assigned a service rate r_k as follows: $r_k > r_k^{avg}$ at each DiffServ server along its routing path. Suppose that F_k^{EF} traverses a routing path of M DiffServ nodes before it arrives at its destination. Let $r^{min} = \min_{i=1...M} r_i$, where $\sum_{k=1}^N r_{k,m} \leq C_{EF}^m$ at m^{th} DiffServ node. In [31], A. K. Parekh and R. G. Gallager derived the e2e delay of any packet belonging to F_k^{EF} , which is described as follows:

$$D_k^{EF} \leq \frac{\sigma_k}{r^{min}} + \sum_{j=1}^M \frac{L_k}{r_{j,k}} + \frac{L_{max}}{C_j - C_{EF}^j} \tag{1}$$

F_k^{EF} is served by m^{th} DiffServ router, if EF traffic aggregate at this router meets two conditions: First, the stability condition which requires $\sum_{k=1}^N \rho_k \leq C_m^{EF}$. Second,

the schedulability condition which requires $\sum_{k=1}^N r_k \leq C_m^{EF}$. If it is accepted, then it is served at service rate at each DiffServ node along a selected routing path by an Edge router, according to one of the following three Resource Division Policies (RDPs).

A. Symmetric Distribution Policy

In the Symmetric Distribution Policy (SDP), all the DiffServ nodes offer to F_k^{EF} packets the same service rate, i.e., $r_{j,k} = r^{min}, \forall j \in [1, \dots, M]$. By substituting $r_{j,k}$ by r^{min} in the formula described in 1, and solving for the minimum service rate r^{min} assigned for flow k , then we can describe r^{min} as follows:

$$r_{j,k} = r^{min} = \frac{\sigma_k + (M - 1) * L_k}{D_k^{EF} - \sum_{j=1}^M \frac{L_{max}}{C_j - C_{EF}^j}} \tag{2}$$

Note that $r_{j,k}$ satisfies the local delay bound of the packets belonging to F_k^{EF} at the DiffServ nodes along their routing path.

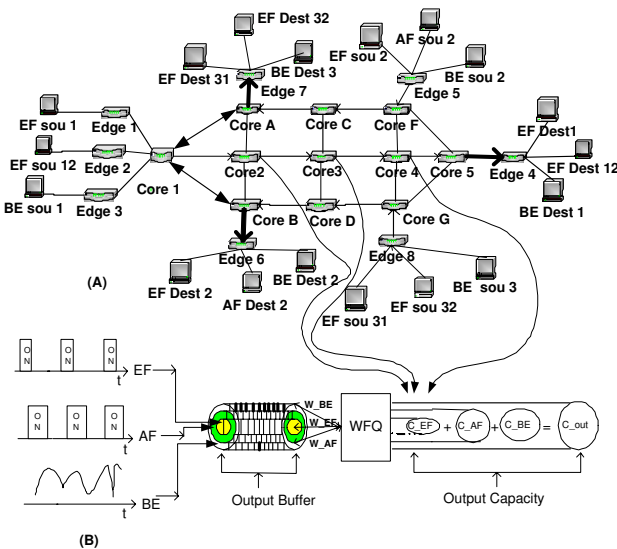


Figure 2. (A) The employed DiffServ network topology in the simulation environment (B) Output buffer and link sharing properties of DiffServ router.

B. Capacity Proportional Policy

In the Capacity Proportional Policy (CPP), the DiffServ nodes offer to F_k^{EF} packets service rate proportional to the reserved capacity rate for EF traffic aggregate C_{EF}^k . Thus, $r_{j,k} = \alpha_j * C_{EF}^j$, $\forall j \in [1, \dots, M]$. By substituting $r_{j,k}$ by $\alpha_j * C_{EF}^j$ in the formula described in 1, and solving this for the constant α_j . Then, we solve it for the minimum service rate $r_{j,k}$ which is described as follows:

$$r_{j,k} = \frac{\frac{\sigma_k - L_k}{C_{EF}^{min}} + \sum_{j=1}^M \frac{L_k}{C_{EF}^j}}{D_k^{EF} - \sum_{j=1}^M \frac{L_{max}}{C_j - C_{EF}^j}} * C_{EF}^j, \quad \forall j \in [1, \dots, M] \quad (3)$$

where α is a constant variable takes values between 0 and 1. Note that, $r_{j,k}$ is required for a new EF flow to meet its local delay bound at each DiffServ node along its routing path.

C. Remaining Capacity Proportional Policy

In the Remaining Capacity Proportional Policy (RCPP), the DiffServ nodes offer to F_k^{EF} packets service rate proportional to the remaining capacity reserved for EF aggregate at a DiffServ node: $R_{j,k} = C_{EF}^j - \sum_{K=1}^N r_{j,k}$, $\forall j \in [1, \dots, M]$. Thus, $r_{j,k} = \alpha_j * R_{j,k}$, $\forall j \in [1, \dots, M]$. By substituting $r_{j,k}$ by $\alpha_j * R_{j,k}$ in the formula described in 1, and solving the equation for the constant α_j . Then, we solve this for the minimum service rate required for EF flow to meet its delay bound at DiffServ nodes, we get the following:

$$r_{j,k} = \frac{\frac{\sigma_k - L_k}{R_{j,k}^{min}} + \sum_{j=1}^M \frac{L_k}{R_{j,k}}}{D_k^{EF} - \sum_{j=1}^M \frac{L_{max}}{C_j - C_{EF}^j}} * R_{j,k}, \quad \forall j \in [1, \dots, M] \quad (4)$$

Note however that using SDP and CPP to compute the service rate at a DiffServ server may result in a service rate greater than the remaining capacity $R_{j,k}$, at one or more DiffServ servers. These hops are called Resource-Restricted Hops (RRHs). To handle this problem, the DiffServ edge router either simply rejects the new flow, or it applies the RCPP, where it distributes the slack in e2e delay over the links constituting F_k^{EF} 's routing path. In the following section, we introduce an optimal delay budget partitioning algorithm for EF e2e delay, which is built on rate-based SSD.

VI. EF E2E DELAY PARTITIONING

When a new EF flow is requested at Edge₁ shown in Fig. 2, and requires D_{new}^{req} , i.e., none of the packets carried by this flow can exceed the delay bound of D_{new}^{req} . Assume that the selected routing path between Edge₁ and Edge₄ for the new flow consists of M links, and that $N-1$ flows have already been admitted on this path. The total capacity and current bandwidth reservation of EF traffic

on the output link j^{th} of each DiffServ node are denoted by $C_{EF_{tot}}$ and $C_{EF_{res}}$ respectively. The goal of delay budget partitioning is to distribute the new flow's D_{new}^{req} into a set of delay budgets $d_{j,new}$ on the M links or nodes and dual leaky bucket delay d_{DLB} operating at the Edge₁, such that the following constraint is satisfied

$$d_{DLB} + \sum_{j=1}^M d_{j,new} \leq D_{new}^{req} \quad (5)$$

And the number of flows that can be admitted over the selected routing path for the new flow in the long-term is maximized. Packets at each DiffServ hop are served by the WFQ, as shown in Fig. 2. In [12], the authors derived the worst case delay $d_{j,k}$ experienced at hop j by any packet belonging to flow F_k^{EF} under WFQ, which is described as follows:

$$d_{j,k} = \frac{\gamma_{j,k}}{r_{j,k}} + \frac{L_k}{r_{j,k}} + \frac{L_{max}}{C_j - C_{EF}^j} \quad (6)$$

where $\gamma_{j,k}$ is F_k^{EF} 's input burst size at hop j . The first component of the queuing delay is fluid fair queuing delay. The second component delay is the packetization delay. And, the third component is SSD non-preemptive delay. For any rate-based SSD, we assume that a function of the form $d_j(\cdot)$ exists that correlates the buffer $B_{EF_{res},k}$ and bandwidth reservation $r_{j,k}$ on a hop j to its packet delay bound $d_{j,k}$, i.e., $d_{j,k} = d_j(r_{j,k}, B_{EF_{res},k})$.

Algorithm 1 Load-Based e2e Delay Slack Partitioning(LB_{DS}P) Algorithm.

Algorithm VI.1: EFDELAYPARTITIONING(D_{req})

comment: Compute $d_{j,new}[j] \quad \forall j \in [1, \dots, M]$

comment: Apply the admission control decision

$\gamma = 0.45; \eta = \gamma;$

while (1)

for $j = 1$ **to** M

do $\left\{ \begin{array}{l} r_{j,new}[j] = \max(C_{EF_{tot}} - C_{EF_{res}}, \\ -C_{EF_{tot}} \frac{\rho_{EF_j}}{\sum_{j=1}^M \rho_{EF_j}} \cdot \eta, r_{new}^{avg}); \\ d_{j,new}[j] = d_j(r_{j,new}[j]); \end{array} \right.$

$d_{DLB} = \frac{\sigma_{new}}{\min_{j \in M} \{r_{j,new}[j]\}};$

$\Delta D_{new} = D_{req} - (d_{DLB} +$

$\sum_{j=1}^M d_{j,new}[j]);$

if $(\Delta D_{new} > 0 \text{ and } \Delta D_{new} \leq \zeta)$

return $(D_{j,new}[j]);$

$\gamma = \frac{\gamma}{2};$

if $(\Delta D_{new} > 0)$

$\eta = \eta + \gamma;$

else

$\eta = \eta - \gamma;$

A. Naive Admission Control Algorithm

Before computing the delay of a new flow $d_{j,New}$ at each link j , the Edge₁ needs to determine whether this new flow F_N^{EF} can be admitted in the network. To this end, first we need to calculate the minimum delay budget that can be guaranteed to F_N^{EF} . Thus the minimum delay budget at link j is given by $d_j(C_{EF_{tot,j}} - C_{EF_{res,j}})$. From the equation described in 6, the flow F_N^{EF} can be admitted if the following is satisfied:

$$\frac{\sigma_{new}}{\min(C_{EF_{tot,j}} - C_{EF_{res,j}})} + \sum_{j=1}^M d_j(C_{EF_{tot,j}} - C_{EF_{res,j}}) \leq D_{new}^{req} \quad (7)$$

B. Load-Based e2e Delay Slack Partitioning

After having admitted the new flow F_N^{EF} , Edge₁ needs to determine the delay assignment at each link along the selected routing path. From the equation described in 7, the edge can calculate the slack in a delay as follows:

$$\Delta D_{new} = D_{new}^{req} - (d_{DLB} + \sum_{j=1}^M d_{j,new}) \quad (8)$$

If the flow F_N^{EF} could be admitted, this means that after assigning minimum delay budgets to the flow at each link, the slack ΔD_{new} is positive. The goal of Load-Based e2e Delay Slack Partitioning (LB_{DSP}) is to reduce the bandwidth requirement of a new flow at each of the M links so that the number of admitted flows can be maximized. By minimizing the load variation over these links, the number of flow requests supported on the routing path can be maximized.

Based on the analysis so far, the procedure of Load-Based e2e Delay Slack Partitioning (LB_{DSP}) is described in the algorithm 1. Let the remaining bandwidth on the j^{th} link after delay budget assignment be the form $C_{EF_{tot}} \cdot \frac{\rho_{EF_j}}{\sum_{j=1}^M \rho_{EF_j}} \cdot \eta$. The algorithm changes the values of η in iterative manner until the resulting ΔD_{new} falls below a predetermined threshold ζ . The more ζ approaches toward the zero value, the close LB_{DSP} can ensure that a link's remaining capacity is proportional to the current load on the links. In the following section, we introduce an EF traffic model based on Linear Bounded Arrival Process [32].

VII. EF FLOW WORKLOAD MODEL

The capability of a DiffServ hop to provide a range of feasible delay values for a given EF flow requires a mechanism by which a hop can determine the amount of service time required by an EF flow over a given time interval. Consider a routing DiffServ hop, m , and an EF flow, k , characterized by its maximum e2e delay value, D_{max}^{EF} , and its Linear Bounded Arrival Process (LBAP) [32] traffic rate specification vector (b_k, r_k, n_k) , where n_k is the number of packets generated over r_k

and b_k is the maximum packet burst size over any time interval, so that the application's long-term packet rate is $(\frac{n_k}{r_k})$. The maximum number of packets, $\sigma_k(\tau)$, generated by k over a time interval of size τ can be computed as follows:

$$\sigma(\tau) = b_k + \lceil \frac{\tau}{r_k} \rceil * n_k \quad (9)$$

Let $\mu_{m,k} = (\frac{L_k}{g_m})$ represents the maximum amount of service time required to process a packet from flow k at DiffServ hop m , where L_k is the packet size of flow k , and g_m is the service rate at hop m . Given the traffic rate specified by flow k , the maximum amount of service time required by k over an interval of size τ can be computed as follows: $S_{k,m}(\tau) = \sigma_k(\tau) * \mu_{k,m}$. Note that, flow k 's workload is specified by computing the maximum number of intervals of size r_k that can fit in τ and multiplying this value by the number of packets that can be generated per r_k interval. This load characterization provides an upper bound on the amount of traffic generated by flow k over a time interval τ . In the following section, we derive the delay range feasibility values at DiffServ hop m .

VIII. DELAY RANGE FEASIBILITY OF DIFFSERV-HOP

Small per-DiffServ hop delay values reduce the effective buffer required for EF traffic flows, but increase its effective capacity requirements. In contrast, large per-DiffServ hop delay values increase the effective buffer required for EF traffic flows. Hence, the characterization of the low feasible delay bound $d_{k,m}^{lfEF}$, of EF flow k at hop m for a fixed amount of buffer, become a factor of hop m 's capacity excess. Consequently, given a sufficient buffer size to hold flow k 's traffic, $d_{k,m}^{lfEF}$ can be exclusively determined based on the amount of capacity that can be allocated to k without violating the traffic requirements of the remaining flows currently supported by hop m .

On the other hand, the characterization of the upper feasible delay bound $d_{k,m}^{ufEF}$ of flow k at DiffServ hop m is directly correlated to the buffer size of this hop. Assuming that a sufficient amount of capacity is available, the upper delay bound $d_{k,m}^{ufEF}$, that can be supported by hop m is exclusively dependent on the amount of buffer capacity that can be allocated to flow k without violating the traffic requirements of the currently supported flows. Consequently, the characterization of $d_{k,m}^{ufEF}$ can be achieved based exclusively on DiffServ hop i 's buffer excess. In what follows, we present models to characterize the capacity and buffer size reserved for EF traffic at DiffServ hop. We then show how these models can be used to derive the low and upper feasible delay bound values, $d_{k,m}^{lfEF}$ and $d_{k,m}^{ufEF}$, respectively.

A. Expedited Forwarding Reserved Capacity Model

For a given class of delay-based DiffServ hop, we can derive some quantitative mechanisms to be used by the hop to assess the feasibility of supporting the requirements of a new EF flow while continuing to guarantee

the requirements of currently supported flows. This can be achieved by computing the utilization of the scheduled activities and comparing it to a schedulable bound which depends on the scheduling policy used by the employing scheduling service discipline at DiffServ hop.

Let m be a DiffServ hop which uses a non preemptive deadline-based scheduling policy to serve EF traffic packets, and consider a set of EF flows, $k = 1, 2, \dots, N$, traversing the hop m . Each flow k is characterized by its traffic specification vector (b_k, r_k, n_k) and its e2e delay $D_{max,k}^{EF}$. Furthermore, packets from flow k encounter a delay $d_{k,m}$ at DiffServ hop m . The support of flows $k = 1, 2, \dots, N$ are feasible at DiffServ hop m if

$$\sum_{k=1}^N \frac{S_{k,m}(d_{k,m})}{d_{k,m}} \leq 1 - \frac{\sup_{1 \leq k \leq N} \{\mu_{k,m}\}}{\inf_{1 \leq k \leq N} \{d_{k,m}\}} \quad (10)$$

where $S_{k,m}(d_{k,m})$ is the maximum amount of service time required by flow k over its assigned delay $d_{k,m}$, at node i . The term $(\frac{\sup_{1 \leq k \leq N} \{\mu_{k,m}\}}{\inf_{1 \leq k \leq N} \{d_{k,m}\}})$ accounts for nonpreemptive aspect of the packet service at hop k . It represents the maximum waiting time before a higher priority packet is served, when it arrives just at the instant a lower priority packet gained access to the DiffServ hop.

For a given a DiffServ hop m and a set of N flows belonging to EF traffic aggregate, with delays, $\{d_{1,m}, \dots, d_{k,m}, \dots, d_{N,m}\}$, and service requirements, $(S_{1,m}(d_{1,m}), \dots, S_{k,m}(d_{k,m}), \dots, S_{N,m}(d_{N,m}))$, respectively, the feasibility test for the above nonpreemptive, delay based DiffServ hop can be described as follows:

$$\sum_{k=1}^N \frac{S_{k,m}(d_{k,m})}{d_{k,m}} \leq C_{EF}^m - \frac{\bar{\mu}}{\underline{d}} \quad (11)$$

where $\bar{\mu} = \sup_{1 \leq k \leq N} \{\mu_{k,m}\}$ and $\underline{d} = \inf_{1 \leq k \leq N} \{d_{k,m}\}$. The term C_{EF}^m denotes the total percentage of service capacity which must be allocated to provide guaranteed service to traffic flows constitute EF traffic aggregate at hop m .

So far we mentioned that amount $C_{EF_{tot}}$ of the whole capacity of DiffServ hop m , C^m is reserved to support the EF aggregate traffic requirements. When a new EF flow is admitted by the network, there is a amount $C_{EF_{res}}$ of $C_{EF_{tot}}$ reserved for already accepted EF flows. Let C_{BE}^m represents the capacity needed for potential nonpreemptiveness at the hop m . Hence, we can define the excess capacity $C_{EF_{exc}}^m$ of $C_{EF_{tot}}$ as the percentage of hop m 's total reserved capacity for EF aggregate, which can be allocated to support new EF flows. Consequently, at any instant, $C_{EF_{exc}}^m$ satisfies $C_{EF_{exc}}^m = C_{EF_{tot}} - C_{EF_{res}} - C_{BE}^m$.

B. Expedited Forwarding allocated Buffer Model

The buffer size B_m determines the total number of packets which can be queued either in the waiting rooms or service queues at DiffServ hop m . The waiting room is used to absorb the jitter [33], [34] introduced upon a flow's packets at the previous upstream hop. A packet is

considered early at a given hop if the actual time spent at the upstream hop is less than the original delay assigned to the packet. The earliness of a packet determines the amount of time a packet is held in the waiting room before it is considered eligible for service and moved to its appropriate service queue. By absorbing the jitter, the traffic pattern of a flow can be reconstructed at each DiffServ hop to the form it had when it entered the network.

At any time, amount B_m^{EF} of hop m 's B_m is allocated to accept EF flows such that no packets are dropped due to lack of buffer space. Let B_m^{exc} represents the buffer excess which can be allocated to handle packets belonging to new EF flows. Thus, $B_i^{exc} = B_m^{EF} - B_m^{res}$. In what follows, we see how the EF capacity and buffer models described so far can be used by a DiffServ hop m to derive the low and upper feasible delay bound values, $d_{k,m}^{lf_{EF}}$ and $d_{k,m}^{uf_{EF}}$, respectively for a new EF flow k , characterized by its rate specification vector (b_k, r_k, n_k) . These delay bound values provide a range of feasible delays that can be supported by the DiffServ hop m .

C. EF Low Feasible delay Value

The characterization of the EF excess capacity is the basis for the computation of the low per-hop delay bound value that can be supported by a DiffServ hop. Let $S_{N+1,m}(\tau)$ represents the maximum workload required by a new EF flow $N + 1$ at DiffServ hop m over a potential delay bound τ . As described above the exact criterion for the new EF flow to be accepted by a delay-based DiffServ hop m , without violating the requirement of the flows currently supported by m , can be described as follows:

$$\sum_{k=1}^N \frac{S_{k,m}(d_{k,m})}{d_{k,m}} + \frac{S_{N+1,m}(\tau)}{\tau} \leq C_{EF}^m - \frac{\bar{\mu}}{\min(\underline{d}, \tau)} \quad (12)$$

Substituting $\frac{S_{N+1,m}(\tau)}{\tau}$ by its workload-based values, results in

$$\frac{b_{N+1} + \lceil \frac{\tau}{r_{N+1}} \rceil \cdot n_{N+1}}{\tau} \leq C_{EF}^m - \sum_{j=1}^N \frac{S_{j,m}(d_{j,m})}{d_{j,m}} - \frac{\bar{\mu}}{\min(\underline{d}, \tau)} \quad (13)$$

The value τ represents a lowest feasible delay bound value $d_{N+1,m}^{lf_{EF}}$ that hop m can offer to new flow $N + 1$. If $b_{N+1} + (\frac{\tau}{r_{N+1}} + 1) \cdot n_{N+1}$ packets can be served by the hop m without any delay, then $b_{N+1} + \lceil \frac{\tau}{r_{N+1}} \rceil \cdot n_{N+1}$ can also be served without any delay by the hop m . The lowest feasible delay values $d_{N+1,m}^{lf_{EF}}$ can thus be described as follows:

$$d_{N+1,m}^{lf_{EF}} = \begin{cases} \frac{b_{N+1} + n_{N+1} + \bar{\mu}}{C_{EF_{exc}}^m - \frac{n_{N+1}}{r_{N+1}}} & \text{if } \tau \leq \underline{d} \\ \frac{b_{N+1} + n_{N+1} + \bar{\mu}}{C_{EF_{exc}}^m - \frac{1}{\underline{d}} - \frac{n_{N+1}}{r_{N+1}}} & \text{O.W} \end{cases} \quad (14)$$

D. EF Upper Delay Bound Value

The upper delay bound value $d_{N+1,m}^{uf_{EF}}$ represents the maximum waiting time a packet from a flow $N + 1$ can

be delayed at DiffServ hop m without causing flow $N + 1$ to loose any of its packets and without violating the buffer requirements of the remaining flows which are currently backlogged at DiffServ hop m . This delay encompasses the total time a packet from flow $N+1$ may wait in i 's waiting rooms and service queues, and can be computed based on the current excess buffer of hop m . Taking in consideration the requirements of currently supported flows, the buffer capacity constraint at hop m can be expressed as follows:

$$\sum_{k=1}^N B_{k,m}^{EF} + B_{N+1,m}^{EF} \leq B_m^{EF} \quad (15)$$

Since packets from $N + 1$ may be queued in their associated waiting room for up to $d_{N+1,m-1}$ units of time, the maximum amount of time a packet from $N + 1$ can be queued at m is $(d_{N+1,m-1} + d_{N+1,m})$. Consequently, the number of buffers, $B_{N+1,m}$, required to ensure a loss-free service to $N + 1$ must be sufficient to hold the maximum number of packets generated by $N + 1$ over $(d_{N+1,m-1} + d_{N+1,m})$. This number can be described as follows:

$$B_{N+1,m} = b_{N+1} + \lceil \frac{d_{N+1,m-1} + d_{N+1,m}}{r_{N+1}} \rceil . n_{N+1} \quad (16)$$

$$b_{N+1} + (\frac{d_{N+1,m-1} + d_{N+1,m}}{r_{N+1}} + 1) . n_{N+1} \leq B_m^{exc} \quad (17)$$

Solving for $d_{N+1,m-1} + d_{N+1,m}$, this results in

$$d_{N+1,m-1} + d_{N+1,m} \leq \frac{r_{N+1}}{n_{N+1}} (B_m^{exc} - b_{N+1} - n_{N+1}) \quad (18)$$

Therefore, if $(B_m^{exc} - b_{N+1} - n_{N+1}) > 0$, a feasible delay bound value $d_{N+1,m}^{u, f_{EF}} = d_{N+1,m-1} + d_{N+1,m}$ can be expressed as follows:

$$d_{N+1,m}^{u, f_{EF}} = \frac{r_{N+1}}{n_{N+1}} (B_m^{exc} - b_{N+1} + n_{N+1}) \quad (19)$$

Based on the analysis so far, we introduce in the following section another optimal EF e2e delay budget partitioning algorithm.

IX. NETWORK RESOURCE BASED E2E DELAY PARTITIONING

By considering the network resource reserved for EF traffic aggregate at the DiffServ nodes, we could compute the feasible delay range values that support new EF traffic flows. Suppose that each DiffServ node has a relatively large output buffer and small link capacity. The initial feasible delay values are then adjusted to meet the capacity requirements of each DiffServ node along the routing path without violating the e2e delay requirements of the flow. This policy is likely to achieve a more efficient use of the network resources, which in turn increases the capability of a DiffServ node to support new flow requests. However, this policy entails some overhead, and it may cost a lot when the EF traffic intensity is small since the benefit of the network resource reduction and resizing process is small. To this end, we propose another

Algorithm 2 Load Balancing Based e2e Delay Partitioning(LB_{EDP}) Algorithm.

Algorithm VIII.1: EFBALANCINGTRAFFIC($Dreq$)

```

for  $j \leftarrow 1$  to  $M$ 
   $d_j^1 = \max(Dreq \cdot \frac{\rho_{EFj}}{\sum_{j=1}^M \rho_{EFj}}, d_j^{lf});$ 
  do  $\begin{cases} d_j^2 = d_j^{lf} + |(d_j^{uf} - (d_j^{lf} + d_{j-1}^{lf}))|; \\ d_{j,new}[j] = \min(d_j^1, d_j^2); \end{cases}$ 
for  $j \leftarrow 1$  to  $M$ 
  do  $\begin{cases} \text{while } (d_{j,new}[j] > d_j^{uf}) \\ \text{do } \{ d_{j,new}[j] = d_j^{lf} + (\frac{d_{j,new}[j] - d_j^{lf}}{2}); \end{cases}$ 
while  $(\sum_{j=1}^M d_{j,new}[j] > Dreq)$ 
  do  $\begin{cases} \text{for } j = 1 \text{ to } M \\ \text{do } \{ D_{j,new}[j] = d_j^{lf} + (\frac{d_{j,new}[j] - d_j^{lf}}{2}); \end{cases}$ 
return  $(D_{j,new}[j]);$ 

```

algorithm 2 for partitioning the entire e2e delay of EF traffic to balance the load across the routing path so the likelihood bottlenecks is minimized. To achieve this objective, we propose the algorithm 2, to assign larger per-DiffServ node delay budgets to highly loaded DiffServ nodes than to lightly loaded nodes. In comparison to the *LB_{DSP}* algorithm, here the algorithm partitions entirely the e2e delay of EF traffic.

A. Load Balancing Based e2e Delay Partitioning Algorithm

The mechanism underlying Load Balancing Based e2e Delay Partitioning (*LB_{EDP}*) is described in the algorithm 2. It uses the entire e2e delay of EF traffic rather than the slack in e2e delay employed in *LB_{DSP}* algorithm, to balance the EF traffic load along the routing path when accepting a new EF traffic flow request. First, it computes the initial d^{lf} for this new flow in term of the EF traffic intensity at each DiffServ node along its routing path. Second, it tunes d^{lf} values using a numerical search method until they lie within each DiffServ node's smallest and largest feasible delay values, without violating the e2e delay requirements of the new and already accepted flows. Consequently, a lightly loaded node is assigned a smaller delay value, hence it accepts higher load. Meanwhile a highly loaded DiffServ node is assigned a higher delay value, thus it accepts a smaller increase in its load.

X. SIMULATION AND RESULTS

We have conducted four experiments to evaluate the performance of *LB_{DSP}* and *LB_{EDP}* algorithms and to compare it with the performance of Equi-Partitioning (EP) algorithm. All the simulations were performed on the network shown in Fig. 2 using a modified version of UC Berkeley ns-2.26 [35]. The primary objective of these

experiments was to manage optimally EF traffic across DiffServ network through EF flows balancing. In the first experiment, we compared the EF e2e delay sensitivity of both algorithms on the number of EF flows accepted along the network simulation run time. In the second experiment, we computed the number of EF flows successfully accepted by our proposed algorithms and EP algorithm from many connection requests generated. In this experiment, the flows accepted require e2e delay bound of 90ms and burst size (σ) of 280bytes. During these experiments, we followed our interest in monitoring and computing the number of EF flows accepted along the routing path which is formed of Edge_{1,2}–Core_{1,2,3,4,5}–Edge₄. To simulate the LB_{EDP} algorithm along this routing path, we applied what we have already explained in section VIII, so we increased the capacity and buffering size at Core_{1,2,3,4,5} to compute d^{lf} and d^{uf} values, respectively at these DiffServ nodes.

In the third experiment, we compared the traffic burst size sensitivity of both algorithms on the number of EF flows accepted. In the fourth experiment, we evaluated the evolution of available capacity C_{av} remained for EF traffic with the number of EF flows accepted by LB_{DSP} , LB_{EDP} and EP algorithms. In this experiment, the flows accepted require e2e delay bound of 84ms and burst size (σ) of 250bytes. During these experiments, we followed our interest in monitoring and computing the number of EF flows accepted along the routing path which is formed of Edge₈–Core_{G,D,B,1,A}–Edge₇.

TABLE I.
EF TRAFFIC SOURCES CHARACTERISTICS DURING LB_{DSP} AND LB_{EDP} SIMULATION

Source Type	Packet Size	Peak Rate	ON time	OFF time
EF source 1 CBR	70 bytes	64kbps	-	-
EF source 12 CBR	70bytes	64kbps	-	-
EF source 2 Exponential	50bytes	64kbps	400ms	600ms
EF source 31 Exponential	50bytes	64kbps	400ms	600ms
EF source 32 Exponential	50bytes	64kbps	400ms	600ms

To simulate the LB_{EDP} algorithm along this routing path, we increased the capacity and buffering size at Core_{G,D,B,1,A}, to compute the feasible lower and upper delay values at these DiffServ nodes. The links connecting both Core₅- Edge₄ and Core_A- Edge₇ shown in Fig. 2 represent the network bottlenecks. The capacity of these bottleneck links is 4.5Mbps, and the capacity of other links is 6Mbps. The capacity is shared between the supported traffic classes as 35% for EF class, which is equivalent to 1.575Mbps of the bottleneck links' capacity, and 2,1Mbps of other links. And, 45 % of the links' capacities are reserved for AF class. The remaining capacity can be used by BE flows. The propagation delay over all the network links was set to 1ms. The characteristics of

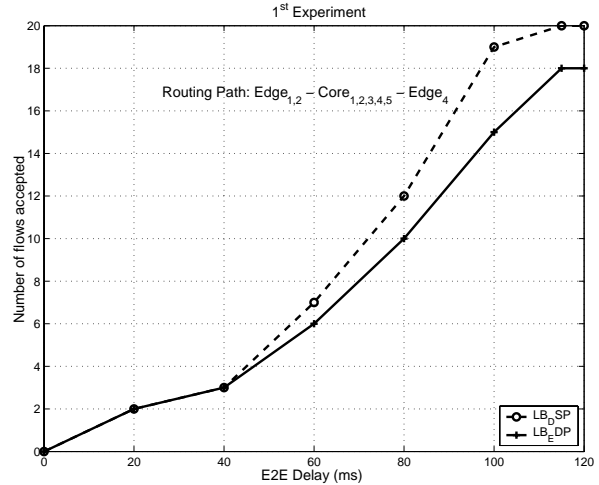


Figure 3. Effect of end-to-end delay on number of EF flows accepted.

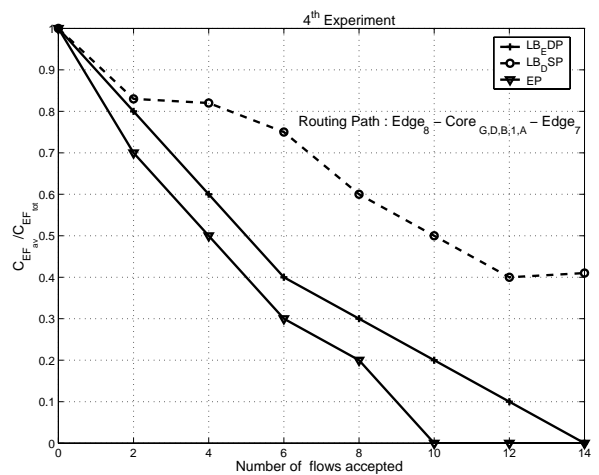


Figure 4. Capacity evolution with the number of EF flows Accepted.

EF traffic sources are described in Table. I.

As a result, Fig. 3 shows us that the number of EF flows accepted increases as their e2e delay requirements increase as well. Obviously, as long as the real time application flows can work normally with higher e2e delay bounds, the network will be capable to accept more EF flows in EF traffic aggregate, provided that all their e2e delay requirements are satisfied. Both algorithms try to accept the maximum of EF traffic flows which is constrained by the bottleneck capacity along their routing path. When the e2e delay is small, both algorithms have almost the same performance. However, when the e2e delay requirements get higher, the LB_{DSP} accepts more flows or outperforms the LB_{EDP} algorithm. This might be referred to that LB_{DSP} performs better EF traffic flows balancing along their routing path, when the slack in their e2e delay is somehow high. When EF flows require a deterministic e2e delay bound, where the number of connection requests is small, then LB_{DSP} algorithm can lead to better network utilization than others as we can conclude from Fig. 4. This results in increasing the

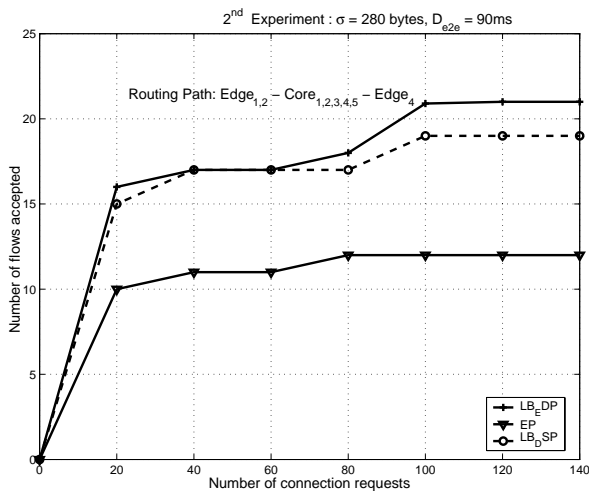


Figure 5. Number of flows accepted Vs. number of connection requests generated.

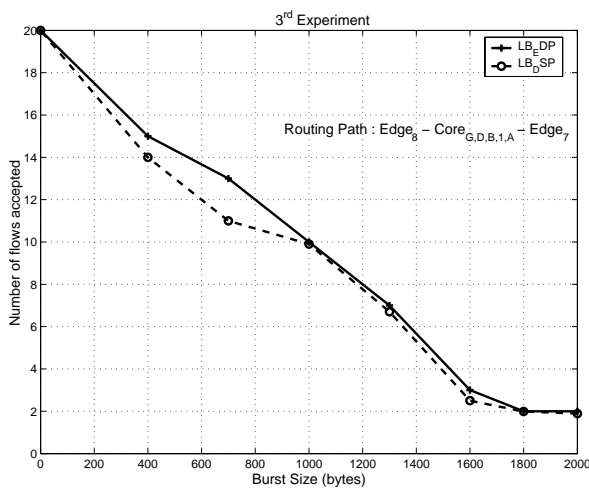


Figure 6. Effect of EF traffic burst size on number of EF flows accepted.

number of eventual EF flows that could be accepted.

However, this good performance of LB_{DSP} algorithm almost disappears as we can conclude from Fig. 5 and Fig. 6. Fig. 5 shows that the LB_{EDP} algorithm accepts more EF flows than LB_{DSP} and EP algorithms as long as the number of connection requests generated increases. Fig. 6 shows also us that the number of EF flows accepted decreases as long as their traffic burst increases. It seems evident to get such results, since the traffic burst size has a direct influence on the e2e delay budget of EF flows. After removing this part from the e2e delay requirement of a flow, the network will have somehow stringent delay requirements, thus it could not support more EF flows since the slack in e2e delay becomes small. However, by employing LB_{EDP} algorithm, DiffServ nodes may support easily the traffic flows' d^{lf} . This is why LB_{EDP} outperforms LB_{DSP} performance when EF traffic is bursty or number of EF connection requests increases leading to a potential traffic burst. Finally, we

can conclude from these results that LB_{DSP} algorithm outperforms LB_{EDP} when the service rate of EF flows is kept constant, and the number of connection requests generated along their routing path is low. However, LB_{EDP} algorithm outperforms LB_{DSP} when EF traffic flows become bursty along their routing path.

XI. CONCLUSION

In this paper, we introduced original management algorithms to be integrated with the call admission control [1] to accommodate optimally EF flows accepted in DiffServ network. In fact, both service management [3] and control mechanism [1] are two main components in the DiffServ architecture to guarantee the e2e QoS required for EF traffic.

We could improve the e2e QoS of EF traffic by managing better its distribution over the different links along its routing path. This management has been done through optimizing network resource reservation for EF traffic in the network, and by controlling the distribution of EF traffic based on the available network resources. We proposed LB_{DSP} and LB_{EDP} algorithms for partitioning the e2e delay requirements of a new EF flow into per-link and per-node delay budget along its routing path, respectively such that the e2e delay of new EF flow and already EF traffic flows accepted in network are guaranteed. Some network resource partitioning policies were incorporated with these algorithms, to apportion the e2e resource requirements into per-link resource along the selected routing path.

We have conducted some experiments using ns 2.26 [35] to compare the performance of e2e delay and traffic burstiness sensitivity of both algorithms on the number of EF flows accepted during the network simulation run time. We have found out that LB_{DSP} algorithm does better traffic balancing than LB_{EDP} , when EF traffic keeps its characteristics imposed at the network ingress routers. In contrast, LB_{EDP} algorithm makes the network capable to accept more EF traffic flows, when this traffic is bursty. By employing optimal e2e delay budget and network resource partitioning algorithms for EF aggregate traffic, we could obtain substantial gains in DiffServ network efficiency and resource utilization. Furthermore, processing load placed by the new flow over the DiffServ node is reduced. EF traffic is also distributed uniformly across the routing path, which results in balancing EF traffic loads on the network links along the different routing paths.

ACKNOWLEDGMENT

We would like to thank Prof. Annie Gravey and Prof. Pascal Lorenz for their valuable comments and rich discussion. We also thank Dr. Wenbing Yao from Brunel university for her support.

REFERENCES

- [1] H. Alshaer and E. Horlait, "Expedited forwarding delay budget through a novel call admission control," in *Proc. of the 3rd European Conference on Universal Multiservice Networks (ECUMN'2004)*, ser. LNCS 3262, Porto, Portugal, October 2004, pp. 50–59.
- [2] H. Alshaer, "Priority traffic control and management in multiservice ip networks," Ph.D. dissertation, Université Pierre et Marie Curie, LIP6, Paris, France, December 2005.
- [3] H. Alshaer and E. Horlait, "Expedited forwarding end-to-end delay budget partitioning," in *Proc. of the IEEE Consumer Communications and Networking Conference, Las Vegas, USA, January, 2006*.
- [4] K. Nichols, V. Jacobson, and L. Zhang, "A two bit differentiated services architecture for the internet," *IETF draft, draft-nichols-diff-svc-arch-00.txt*, November 1997.
- [5] S. Blake, D. Black, E. D. M. Carlson, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475, IETF*, December 1998.
- [6] B. Davie, A. Charny, J. Bennett, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding phb (per-hop behavior)," *RFC 3246, IETF*, March 2002.
- [7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.
- [8] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362 – 373, August 1998.
- [9] M. S. Kodialam and T. V. Lakshman, "Minimum interference routing with applications to mpls traffic engineering," in *Proc. of the Infocom*, pp. 884–893, March 2000.
- [10] M. Kodialam and S. H. Low, "Resource allocation in a multicast tree, new york," in *Proc. of Infocom*, March 1999.
- [11] H. Alshaer and E. Horlait, "Management and control of expedited forwarding traffic," *submitted to computer communications journal on 06 September, 2005*, 2005.
- [12] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," in *IEEE/ACM Transaction Networking*, vol. 1, no. 3, June 1993, pp. 334–357.
- [13] R. Guerin, L. Georgiadis, and A. Parekh, "Optimal multiplexing on a single link," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1518–1535, September 1997.
- [14] M. Andrews and L. Zhang, "Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule," in *Proc of the INFOCOM '99*, 1999.
- [15] V. Sivaraman, F. M. Chiussi, and M. Gerla, "Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping," in *Proc. of the NFOCOM'2000*, 2000, pp. 631–640.
- [16] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. Assoc. Computer Machine*, vol. 20, pp. 46–61, January 1973.
- [17] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *High Speed Networks*, vol. 3, no. 4, pp. 389–412, 1994.
- [18] R. Nagarajan, J. Kurose, and D. Towsley, "Local allocation of end-to-end quality-of-service in high speed networks," in *Proc. of IFIP Workshop on Performance Analysis of ATM Systems, North Holland*, pp. 99–118, January 1993.
- [19] D. H. Lorenz and A. Orda, "Optimal partition of qos requirements on unicast paths and multicast trees," *IEEE/ACM trans. on Networking*, vol. 10, no. 1, pp. 102–114, February 2002.
- [20] R. A. Gurin and A. Orda, "Qos routing in networks with inaccurate information: theory and algorithms," *IEEE/ACM Trans. on Networking*, vol. 7, no. 3, pp. 350–364, 1999.
- [21] A. Orda and A. Sprintson, "A scalable approach to the partition of qos requirements in unicast and multicast," in *Proc. of IEEE Infocom*, vol. 2, June 2002, pp. 685–694.
- [22] A. Girard, C. Rosenberg, and H. Cho, "Optimal performance partitioning for networks with envelope-regulated traffic," *15th ITC Specialist Seminar*, July 2002.
- [23] H. Zhang and E. W. Knightly, "Providing end to end statistical performance guarantees with bounding interval dependent stochastic models," in *Proc. of the SIGMETRICS*, pp. 211–220, 1994.
- [24] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE journal on Selected Areas in communications*, vol. 8, no. 3, pp. 368–379, April 1990.
- [25] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet switched networks," in *Proc. of the IEEE*, vol. 82, pp. 122–139, January 1994.
- [26] J. Kurose, "Open issues and challenges in providing quality of service guarantees in high-speed networks," *Computer Communication*, vol. 23, no. 1, pp. 6–15, January 1993.
- [27] A. Girard, H. Cho, and C. Rosenberg, "On optimal end-to-end qos budget partitioning in network dimensioning," in *Proc. of 19th ITC*, vol. 2, 2005.
- [28] V. Firoiu and D. Towsley, "Call admission and resource reservation for multicasat sessions," in *Proc. of IEEE Infocom*, 1996.
- [29] F. Ergun, R. Sinha, and L. Zhang, "Qos routing with performance dependent costs," in *Proc. of IEEE Infocom*, March 2000.
- [30] D. Raz and Y. Shavitt, "Optimal partitioning of qos requirements with discrete cost functions," in *Proc. of IEEE Infocom*, March 2000.
- [31] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," in *IEEE/ACM Transaction Networking*, vol. 2, no. 2, April 1994, pp. 137–150.
- [32] R. L. Cruz, "A calculus of network delay and a note on topologies of interconnection networks," Ph.D. dissertation, Univ. of IL, July 1987.
- [33] H. Alshaer and E. Horlait, "Expedited forwarding end to end delay variations," in *Proc. of the IFIP/IEEE International Conference on Telecommunications (ICT)*, ser. LNCS 3124, Los Fortaleza, Cear Brazil, August 2004, pp. 1183–1194.
- [34] —, "Expedited forwarding end to end delay jitter in the differentiated services network," in *Proc. of the IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC'04)*, ser. LNCS 3079, Toulouse, France, June 2004, pp. 14–25.
- [35] L. N. R. Group, "Lbnl network research group, ucb/lbnl/vint network simulator- ns(version 2.26)," <http://www-mash.cs.berkeley.edu/ns/>, September, 1997.

Hamada Alshaer was born in the city of Rafah in Gaza Strip, Palestine on 02 November, 1977. He graduated from the Department of Electrical Engineering and Computer Science at Birzeit University, Palestine in 2001. He also received the M.S. (DEA) degree in information technologies and Systems from the Compiègne university of technology, Compiègne, France, in 2002. He defended his thesis, entitled "Priority Traffic Control and Management in Multiservice IP Networks" on 08 December, 2005, at Pierre et Marie Curie university. Prof. Eric Horlait was his PhD supervisor. Currently, he is a postdoctoral research fellow at Brunel University, United Kingdom. His main research interests revolve around traffic control and management in multiservice IP networks, QoS, MIPv6, mobility management, Content Distribution Networks (CDNs), self-organizing networks, and inter-vehicle communications.