Providing Quality of Service Guarantees Using Only Edge Routers

Sudeept Bhatnagar and Brett J. Vickers

{sbhatnag,bvickers}@cs.rutgers.edu

Dept. of Computer Science

Rutgers University

110 Frelinghuysen Road

Piscataway, NJ 08854-8019

Abstract

Providing strict bandwidth guarantees to packet flows in the Internet is an inherently challenging task. It requires signaling mechanisms, policing mechanisms, accurate and rapid accounting of network resources, and call admission control. We present a novel protocol that provides bandwidth guarantees to Internet packet flows. This protocol, called the Edge-assisted Quality of Service (EQOS) protocol, requires modifications to only a subset of an administrative domain's routers, namely the edge routers on the domain's periphery. Legacy routers within the core of the domain require no modifications. We present the results of several simulation experiments showing that EQOS is able to provide bandwidth guarantees to competing flows.

Keywords

quality of service, network architecture, core-stateless networking, route pinning, bandwidth guarantees

I. Introduction

It is an inherently challenging task to provide quality of service guarantees to packet flows generated by network applications. Providing QoS guarantees requires that applications be able to signal their resource requirements to the network, that the network be able to account for its available resources, that the network be capable of making distributed call admission decisions, and that the network be capable of policing the behavior of individual flows. The challenge of providing quality of service guarantees is particularly daunting in the Internet, because the Internet was designed to be connectionless. No per-flow state is maintained by Internet routers, and thus per-flow accounting and policing mechanisms are not easily integrated into the Internet.

Nevertheless, over the years many researchers have taken up the challenge by proposing modifications or additions to the Internet architecture to give it support for quality of service. While the proposed approaches differ significantly in detail, one characteristic they universally share is their requirement that every router within a network be modified in order to support QoS. In the stream protocol [1] and integrated services [2], [3] approaches, all routers must maintain per-flow state and participate in a signaling protocol. In the core-stateless approaches, the most notable of which are differentiated services [4] and dynamic packet state [5], per-flow state is maintained only at routers on the edge of the network, but routers at the core of the network are also modified to perform special operations on individual packets.

We propose an alternative approach, in which only a subset of the network's routers requires modification in order to support quality of service guarantees. More specifically, within a single administrative domain, only routers on the edge of the network require modification; routers within the core of the network require no changes. This approach, which we call *Edge-assisted Quality of Service* (EQOS), is potentially more scalable, less costly to deploy, and simpler to migrate toward than other proposed QoS solutions for the Internet. Moreover, it is interoperable with other QoS solutions such as differentiated and integrated services.

The EQOS protocol consists of a resource reservation signaling mechanism, a distributed call admission control mechanism, and a local route pinning mechanism. We detail these mechanisms in section 2. In section 3, we present the results of several simulation experiments intended to demonstrate the ability of EQOS to provide bandwidth guarantees to flows that require them. In section 4, we discuss several features and implementation issues of the EQOS protocol. And in section 5, we provide some concluding remarks.

II. The EQOS Protocol

The EQOS protocol makes several assumptions about the network in which it is implemented. For the purposes of the protocol, a network is generally defined as a group of interconnected routers. Within a network, the protocol distinguishes between two types of routers: edge routers, which reside at the boundaries of the network, and core routers, which reside within the network's perimeter. Edge routers may behave as either *ingress* or *egress* routers, depending on whether they act on packets entering or leaving the network. The edge router through which a packet enters the network is considered that packet's ingress router, whereas the edge router through which a packet leaves the network is considered that packet's egress router. The EQOS protocol is implemented entirely within the edge routers; no modifications are made to core routers.

EQOS allows for two types of packet flows: reserved flows and best effort flows. A reserved flow is any flow that requires bandwidth be allocated to it on the path between the ingress and egress router, whereas a best effort flow is a flow that requires no bandwidth reservation. Reserved flows are allocated bandwidth through the use of (1) an end-to-end signaling mechanism and (2) a token passing mechanism that maintains reservation consistency across edge routers. Best effort flows require no signaling and may enter the network without any special participation by the end systems.

The EQOS protocol assumes the existence of a network routing protocol that updates edge routers with information about the topology of the network. This is a reasonable assumption, since widely used link state routing protocols such as OSPF [6] do this. The protocol also assumes the existence of a route pinning mechanism that is able to force a given flow's packets to follow a specific path between two edge routers. Examples of such route pinning mechanisms include multiprotocol label switching (MPLS) [7] and strict IP source routing. Although IP source routing is not feasible on an end-to-end basis due to the limited space available in the IPv4 header, it is feasible within a single administrative domain, where the number of router hops is substantially lower.

In the remainder of this section, we discuss the details of the EQOS protocol. First, we present the signaling mechanism for reserved flows. Then we present the distributed call admission mechanism used by EQOS. Finally, we present a simple route pinning mechanism that works with many existing routers.

A. Signaling for Reserved Flows

The EQOS protocol relies on the RSVP signaling mechanism to let applications inform the network of their bandwidth requirements. The RSVP protocol is an end-to-end, soft-state, receiver-initiated resource reservation algorithm for network flows. In RSVP, sources and receivers periodically exchange signaling messages, which cause the installation or refreshing of state in all routers through which they pass. The source of a flow periodically sends a *path* message toward the receivers. As path messages passes through routers, they install or update temporary “path state” in the routers, indicating the path through which the path message traveled. Receivers of path messages also periodically generate *reservation* messages, which routers propagate toward the source along the reverse of the path used by the path messages. Reservation messages cause the installation or update of “reservation state” in the routers. When a router receives a reservation message, it determines whether there are enough resources to handle the reservation. If so, it installs (or updates) its reservation state for the flow and propagates the reservation message to the next router indicated by the path state. If not enough resources are available, an RSVP reject message is returned to the receiver, and routers on the way remove any state that had been reserved for the flow.[1]

[1] This overview of RSVP is necessarily brief. For a readable and more complete discussion of the RSVP protocol, the reader is encouraged to read the seminal tutorial paper on the topic [2].

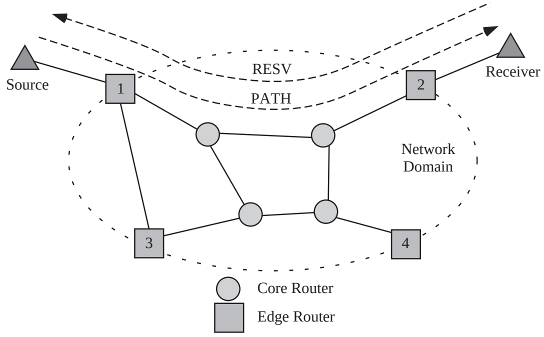


Fig. 1. An example illustrating the exchange of RSVP messages in an EQOS network

Since one of the design goals of EQOS is to relieve core routers from the burden of maintaining per-flow state, an EQOS-capable network must interact with RSVP messages in a way that is different from normal RSVP-enabled networks. The EQOS protocol interacts with RSVP as follows:

1. When an ingress router receives a path message, it stamps its own address into the message and forwards it toward the receiver.
2. When a core router receives a path message, it simply routes and forwards the packet like any other.
3. When an egress router receives a path message, it uses the address stamped into the message by the ingress router to set up or update the flow's path state. The egress router then re-stamps its own address in the path message and forwards it to the next hop.
4. When an egress router receives a reservation message, it examines the flow specification contained in the message and looks up the flow's path state, which indicates the flow's ingress router. The egress router uses a route pinning mechanism (such as source routing or MPLS) to return the message to the flow's ingress router.
5. When a core router receives a reservation message, it simply routes and forwards the packet like any other.
6. When an ingress router receives a reservation message, it examines the message's flow specification as well as its reservation specification. If the flow's reservation request can be met, reservation state for the flow is installed or updated in the ingress router. If the reservation cannot be met, an RSVP reject message is returned to the receiver.
7. When an egress router receives an RSVP reject message, it frees the path state and reservation state for the specified flow and forwards the reject message toward the receiver.

This mechanism is illustrated by example in Fig. 1. The source of a reserved flow transmits its first path message toward the receiver. On receiving the path message, edge router 1 installs path state, stamps its address in the packet and forwards the packet toward the receiver. When edge router 2 receives the path message, it also installs path state, re-stamps its address in the path message, and forwards the packet toward the receiver. Upon receiving the path message, the receiver decides to place a reservation, so it issues a reservation message containing the flow's specification and a reservation specification. The reservation message arrives first at the flow's egress router, where the flow's path state is looked up. The path state is used to determine the next edge router that should receive the reservation. The reservation message is returned to this edge router using a route pinning mechanism. When the flow's ingress router receives the reservation message, it determines whether there are enough resources available within the network to support the reservation request and, if so, updates its reservation state and returns the reservation message to the flow source.

Note that, unlike the typical RSVP implementation, admission control decisions are made only by edge routers; core routers simply forward RSVP signaling packets like any other packet. This eliminates the requirement of maintaining per-flow state at core routers. In fact, it eliminates any modification to core routers at all.

B. Distributed Admission Control

The signaling mechanism is necessary to let end systems indicate their network resource requirements. However, the heart of the EQOS protocol is its distributed admission control mechanism, which allows edge routers to assume the core routers' responsibility for making admission control decisions.

In order for edge routers to make effective admission control decisions, they must maintain a consistent view of the reservations already in effect on each link in the network. EQOS uses a token passing mechanism to maintain this consistency. The token is simply a control packet that circulates among edge routers and describes the amount of bandwidth available on every link within the network.[2] A link's available bandwidth is defined as the difference between its bandwidth capacity and the amount of bandwidth reserved on the link. At any given time, there is only one token in circulation.

When an edge router receives a token, it uses the token's contents in conjunction with its knowledge of the network's topology to determine the state of the network. This information is used by the edge router to select routes for new reserved flows entering the network. Route selection may be carried out using any of a number of possible routing algorithms, the likeliest candidates being shortest path, shortest widest path, or shortest wide-enough path. We consider in this paper only the latter routing algorithm, which operates by removing all links with insufficient bandwidth to meet the reservation request and selecting the shortest path through the

[2] For now, we assume tokens travel between edge routers on a circulation path determined in advance by the network operator. We leave for future work the development of algorithms to dynamically select token circulation paths.

network using the remaining links.

If a route is successfully selected, the edge router pins the route to the flow, updates the available bandwidth on each link of the route, and sends an updated token to the next edge router. The reservation for the flow does not go into effect, however, until the flow's ingress router fully circulates the updated token among all edge routers. This provision is necessary for two reasons. First, it prevents several edge routers from over-allocating bandwidth by simultaneously reserving bandwidth on a single link. Second, it gives other edge routers the opportunity to reduce the rates of best effort flows sharing the same links as the new reserved flow. Without this provision, reserved flows would not be guaranteed a fixed allocation of bandwidth. Although this provision results in a delay while the reservation is propagated, packets from the reserved flow may be treated as best effort packets until the reservation is complete.

Tokens also contain information about the best effort flows being carried by the network's links. Since core routers are not capable of distinguishing between reserved and best effort flows, edge routers must take care to limit the amount of best effort traffic allowed onto each link. If an ingress router did not limit the rates of its best effort flows, they would negatively impact the performance of reserved flows, leaving the reserved flows with meaningless reservations. To prevent this from happening, tokens include the number of best effort flows being carried on each link. These numbers are called *flow counts*. When a new best effort flow arrives, the ingress router selects a route for the flow and increments the flow count for each link in the route.

Once an ingress router knows the flow count and the available bandwidth on each link, it must limit the rate that packets from each best effort flow enter the network. This rate, called the flow's *ingress rate*, can be calculated in a number of ways. The first approach we consider is a naive approach. In the naive approach, the ingress router first calculates the fair share of every link in the flow's route. A link's fair share is equal to its available bandwidth divided by its flow count. Once the ingress router has calculated the fair shares, it determines the flow's ingress rate. The ingress rate is set to the fair share of the flow's bottleneck link, which is the link in the flow's route with the smallest fair share.

One problem with the naive approach is that it cannot maintain consistency unless best effort flows are buffered at the ingress router until a token arrives. Added buffering is not desirable, however, because it imposes added delays on best effort traffic. To avoid this inefficiency, a slight modification to the naive approach can be made. Instead of buffering a flow's packets while the ingress router waits for a token to arrive, they can be allowed into the network immediately so long as there are already other flows traveling from the ingress router to the same egress router. The rates of these other flows are temporarily reduced in order to give the new flow an equal share of the route's available bandwidth. When a token arrives, the ingress router computes

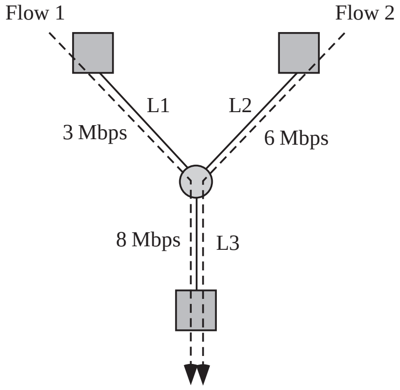
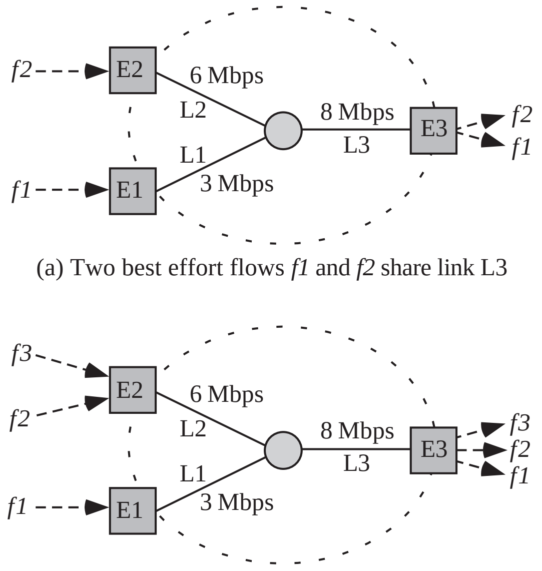


Fig. 2. Two best effort flows competing for a single link

a route for the new flow, updates the flow count for links in the new route, and continues to send the new flow on the old route until the token has had time to fully circulate the network. When the token returns again to the edge router, the ingress router assigns the new flow to its new route and limits the new flow's ingress rate accordingly.

Another problem with the naive approach is that it may limit ingress rates in such a way that some links are persistently underutilized. Consider the example shown in Figure 2. Since flow 1 is bottlenecked at link L1, its ingress rate is limited to 3 Mbps. Flow 2, on the other hand, is considered by its ingress router to be bottlenecked at link L3, since it shares the link with one other flow. The ingress rate of flow 2 is therefore limited to 4 Mbps, which is the fair share of available bandwidth on link L3. The final result is that 1 Mbps of link L3's available bandwidth remains unutilized by best effort flows. The reason for this undesirable result is that ingress routers only know the number of best effort flows crossing each link; they do not know where flows from other ingress routers are bottlenecked. In the example of Figure 2, flow 2's ingress router does not know that the other flow sharing link L3 is bottlenecked by link L1.

Fortunately, an enhancement can be made to the naive approach in order to eliminate this effect. The enhancement entails modifying the token to include two additional pieces of information about each link in the network. In addition to the flow counts and the available bandwidths, the token lists for each link (1) the *residual bandwidth*, which is the total amount of available bandwidth that cannot be fully utilized by the link's carried best effort flows, and (2) the *claimed residual bandwidth*, which is the amount of residual bandwidth temporarily being used by best effort flows. Residual bandwidth is any available bandwidth left over when all best effort flows sharing the link are unable to fully utilize the link's available bandwidth due to the existence other bottleneck links. This residual bandwidth can be claimed by best effort flows on a first-come first-serve basis. Once a flow has claimed residual bandwidth, the claim is only good until another best effort flow needs



(a) Two best effort flows *f1* and *f2* share link L3

(b) Later, a new flow *f3* arrives at E2

Fig. 3. An example scenario for the admission of a new best effort flow

the bandwidth to achieve its fair share.

When deciding a flow's ingress rate using the enhanced approach, an ingress router calculates the *best share* of each link in the flow's route and selects the smallest value. The best share is defined as follows:

$$\begin{aligned} \text{best share} &= (\text{available bandwidth}/\text{flow count}) \\ &\quad +\text{residual bandwidth} \\ &\quad -\text{claimed residual bandwidth} \end{aligned} \tag{1}$$

Once the ingress rate has been determined, the values of the residual bandwidth and claimed residual bandwidth on each link of the flow's route are updated. In the example of Figure 2, this algorithm leads to an ingress rate of 3 Mbps for flow 1 and an ingress rate of 5 Mbps for flow 2.

Best effort ingress rate limiters are most naturally implemented as rate controlled fair queuing schedulers [8], [9]. A fair queuing scheduler is allocated for each active edge-to-edge route, and each flow following a given route is allocated a queue in the route's associated fair queuing scheduler. The advantage of using fair queuing schedulers is that they are work conserving; when one flow does not fully utilize its fair share of available bandwidth, another flow following the same route can exploit the unutilized bandwidth. Although on first glance it might seem that having one fair queuing scheduler per edge-to-edge route is excessive, in fact the number of active routes between two edge routers is typically not very large.

For an example of how best effort flows are treated by the EQOS protocol, consider the scenario depicted in Figure 3. Assume the token circulates in a clockwise manner from E1 to E2 to E3. Assume also that the most recent token received by edge routers E1 and E2 indicates that link L3 carries two best effort flows, and links

L1 and L2 carry one best effort flow each. For the purposes of this example, there are no reserved flows being carried by any of the links. Hence, flow *f1* is limited to a rate of 3 Mbps, so it has set the residual bandwidth on link L3 to 1 Mbps; flow *f2* has claimed this residual bandwidth and is therefore limited to a rate of 5 Mbps. At a later time, when the token is still at edge router E3, a new flow *f3* arrives at E2. Instead of waiting to receive the token, E2 allows packets from flow *f3* to follow the same route as flow *f2* and to share the 5 Mbps of bandwidth used by flow *f2*. When E2 receives the token, it computes the route for flow *f3* and increments the flow counts for links L2 and L3. However, it continues to serve *f2* and *f3* at a combined rate of 5 Mbps until the updated token has had a chance to fully circulate the edge routers. When E1 receives the modified token, it observes that there are 3 flows sharing link L3 and reduces the rate of flow *f1* to $1.\overline{6}$ Mbps, which is equal to flow *f1*'s fair share of link L3 minus the claimed residual bandwidth which E1 had made available on link L3. Since flow *f1* is not receiving its full fair share, edge router E1 also reduces the residual bandwidth on link L3 to zero and forwards an updated token to E2. At this point, E2's updated token has had a chance to fully circulate the network and so E2 begins regulating the rates of flows *f2* and *f3* independently. E2 also notices that the residual bandwidth has been reduced since it last saw the token, so it relinquishes its claimed residual bandwidth and calculates a best share for flows *f2* and *f3* of $2.\overline{6}$ Mbps each. When E1 sees the next updated token, it notices that another edge router has relinquished its claimed residual bandwidth and increases the rate of flow *f1* to $2.\overline{6}$ Mbps. At this point, all three flows are transmitting at equal rates across the shared bottleneck link L3.

C. Local Route Pinning

The EQOS protocol relies on a local route pinning mechanism to ensure that packets from all flows, both best effort and reserved, follow known routes through the network. Route pinning is necessary to maintain resource reservations; if routes followed by packets are not predictable, then no guarantees about resource allocation can be made.

To support route pinning, there are basically two options. One option is to use MPLS, which is a standardized route pinning technique for switch-based routers. When MPLS is used as the local route pinning mechanism for EQOS, all edge and core routers are assumed to support MPLS.

Unfortunately, there are a large number of networks that do not support MPLS. For EQOS to be effective in these networks, another option must be considered. Since many routers implement IP source routing, we consider this to be a suitable alternative for local route pinning between edge routers. An ingress router ensures that a packet follows a particular route by including the IP addresses of all core routers between the two edge routers in the IP header's options field. When the egress router receives a source routed packet, it strips the source route and forwards the packet to the next network. By stripping source routes from packets as they leave the network, EQOS ensures that the route pinning mechanism is local to the network in which it operates. It also makes IP source routing a feasible option for route pinning, since it is only operates within a single administrative domain, which is not likely to have a large number of core routers.

III. Mathematical Results

The EQOS algorithm is meant to provide bandwidth guarantees to reserved flows. As the core routers are oblivious of the existence of a special category of flows, namely the reserved flows, the only way to provide guarantees is to ensure that the combined rates of flows assigned to a link, do not exceed its capacity. In this section we formally establish thar EQOS never over-allocates the capacity of a link.

Claim: EQOS does not allow the sum of rates of flows passing through a link l, to be more than its capacity.

Proof:

Consider a link *l* with capacity *C*.

Assume that the state of a link is defined by the number of reserved flows, number of bottlenecked flows and the number of non-bottlenecked flows traversing it.

Let,

n_r = number of reserved flows on *l*

n_1 = number of best effort flows which are not bottlenecked (the ones which contribute some residual bandwidth) at *l*

n_2 = number of best effort flows which are bottlenecked (the ones which do not contribute to the residual bandwidth) at *l*

$n_b = n_1 + n_2$ = total number of best effort flows on *l*

Also,

C_q = total bandwidth reserved on *l* by n_r reserved flows

$C - C_q$ = Total available capacity for all best effort flows.

C_r = Total residual capacity, which is the unused portion of fair share of the n_1 non-bottlenecked flow.

Assume that for the some values of n_r, n_1 and n_2 the total rates of these flows are not more than *C*.

The bandwidth distribution in the current state of the link is as follows:

Reserved flows use C_q in all.

Non-bottlenecked flows have a fair share $\frac{n_1(C-C_q)}{n_b}$. Of this they leave C_r as residual bandwidth. Hence, their total rate is $\frac{n_1(C-C_q)}{n_b} - C_r$.

The bottlenecked flows use $\frac{n_2(C-C_q)}{n_b}$ which is their fair share and claim the residual bandwidth C_r making their total rate $\frac{n_2(C-C_q)}{n_b} + C_r$.

Hence, the total allocated capacity is:

$C_q + (\frac{n_1(C-C_q)}{n_b} - C_r) + (\frac{n_2(C-C_q)}{n_b} + C_r)$ which is equal to C.

We trace our algorithm's state transitions and show that in all possible transitions from the current state , the total rate of flows is never more than C.

Case 1:

The number of reserved flows on *l* increase by 1 or $n'_r \leftarrow n_r + 1$. The total bandwidth reserved on *l* is now C'_q with $C'_q > C_q$. Note that this reservation does not change the total number of best effort flows, but it can change the number of non-bottlenecked and bottlenecked flows to n'_1 and n'_2 respectively.

During the first token cycle the best effort flows reduce their rates (the new reserved flow can starts *after* 1 token cycle). Now with lesser bandwidth available for the best effort flows, the number of the bottlenecked flows(n'_2) can not reduce when the total reservation increases or $n'_2 \geq n_2$. Since $n_b = n'_1 + n'_2$ hence $n'_1 \leq n_1$. This also means that now the total residual bandwidth $C'_r \leq C_r$ since the newly bottlenecked flows (if any) will take away their part of residual bandwidth. Since EQOS makes flow continue at their current rate minus the provided residual bandwidth for 1 token cycle, the rate of flows using *l* at the end of this cycle is:

$$C_q + (\frac{n'_1(C - C'_q)}{n_b} - C_r) + (\frac{n'_2(C - C'_q)}{n_b} + C'_r)$$

$$= C + C_q - C'_q + C'_r - C_r < C$$

After the second token cycle the new reserved flow starts making the total rate of reserved flows C'_q. The bottlenecked best effort flows reclaim the residual bandwidth whichmakes their rate $(\frac{n'_1(C-C'_q)}{n_b} - C'_r)$. At this point, the total rate of all flows on *l* is:

$$C'_q + (\frac{n'_1(C - C'_q)}{n_b} - C'_r) + (\frac{n'_2(C - C'_q)}{n_b} + C'_r)$$

$$= C'_q + C - C'_q = C$$

Case 2a: A new best effort flow starts. It is bottlenecked at some other link. The number of best effort flows on *l* increase by 1 or $n'_b \leftarrow n_b + 1$. Since the arrival of a new best effort flow reduces the fair share of each of the other best effort flows sharing *l*, the number of bottlenecked flows can increase as well. Assume that the number of bottlenecked flows does not increase, instead the number of non-bottlenecked flows increase by one or $n'_1 \leftarrow n_1 + 1$.

In the first token cycle, the new flow shares bandwidth with the flows following the same path through the domain. So the total rate does not increase during the cycle. The best-effort flows which are already present, reduce their rates to take into account the new flow. The new flow will add some bandwidth to the residual bandwidth as it is a non-bottlenecked flow. The other non-bottlenecked flows will reduce their share of residual bandwidth. Clearly, the new value of total residual bandwidth on the link, C'_r, cannot be more than C_r. The non-bottlenecked flows continue to send at a rate consistent with C_r for one token circulation cycle. The bottlenecked flows reduce their rate during the cycle so that the total claimed bandwidth becomes C'_r.

After 1 circulation cycle the total rate of flows would be :

$$C_q + (\frac{n_1(C - C_q)}{n'_b} - C_r) + (\frac{n_2(C - C_q)}{n'_b} + C'_r)$$

$$= C_q + \frac{(n_1 + n_2)(C - C_q)}{n_1 + n_2 + 1} - (C_r - C'_r) \leq C$$

After 2 token circulation cycles, the combined rate of flows becomes:

$$C_q + (\frac{n'_1(C - C_q)}{n'_b} - C'_r) + (\frac{n_2(C - C_q)}{n'_b} + C'_r) = C$$

Case 2b:

A new best effort flow arrives on the link. It causes a decrease in the fair share of all the best effort flows and increases the number of bottlenecked flows on *l* or $n'_2 > n_2$ and $n'_b \leftarrow n_b + 1$.

The new flow does not increase the total rate during the first token cycle as it shares bandwidth with the flows following the same path through the domain. The existing best-effort flows reduce their rates to account for the new flow. The non-bottlenecked flows will reduce their share of residual bandwidth as their excess of fair share is now lesser than it was before the arrival of new best effort flow. So the new value of total residual bandwidth on the link, C'_r, is at most equal to C_r. Since the flows which had allowed some residual bandwidth during the last cycle (all the flows which were non-bottlenecked in the last cycle) continue at their old rate minus their allowed residual bandwidth for one cycle and the bottlenecked flows reduce their rate to get consistent with the new value of residual bandwidth, the total rate after 1 token circulation cycle is:

$$C_q + (\frac{n_1(C - C_q)}{n'_b} - C_r) + (\frac{n_2(C - C_q)}{n'_b} + C'_r)$$

$$= C_q + \frac{(n_1 + n_2)(C - C_q)}{n_1 + n_2 + 1} - (C_r - C'_r) \leq C$$

After 2 token circulation cycles, the combined rate of flows becomes

$$C_q + (\frac{n_1'(C - C_q)}{n_b'} - C_r') + (\frac{n_2(C - C_q)}{n_b'} + C_r') = C$$

Case 3:

One of the reserved flows ends; $n_r' \leftarrow n_r - 1$. The new reserved capacity $C_q' < C_q$. The bandwidth available to the best effort flows is more than the last cycle. This implies that the number of bottlenecked flows at *l* cannot increase or $n_2' \leq n_2$. As the total number of best effort flows, n_b, is the same, $n_1' \geq n_1$. The non-bottlenecked flows will leave some extra residual bandwidth as their fair share increases and they cannot use any more of that fair share or $C_r' \geq C_r$. Hence the bottlenecked flows will use $\frac{n_1'(C-C_q')}{n_b} - C_r'$. By the end of the cycle, the non-bottlenecked flows will utilize some of the residual bandwidth, C_r''

After the first token cycle the total rate will be

$$C_q' + (\frac{n_1'(C - C_q')}{n_b} - C_r') + (\frac{n_2'(C - C_q')}{n_b} + C_r'') \leq C$$

After the second circulation cycle, the entire residual bandwidth will be taken up by the non-bottlenecked flows. Then the total rate of the flows will be C.

Case 4:

One of the best effort flow on *l* ends (it may be a bottlenecked flow or a non-bottlenecked flow). The number of best effort flows reduce by 1 or $n_b' = n_b - 1$. Now the fair share of the best effort flows increases. This means that some of the flows which were bottlenecked at *l* earlier, may not be bottlenecked any more or $n_2' \leq n_2$. Hence they may contribute some residual bandwidth or $C_r' \geq C_r$.

After one token cycle, the total rates of the flows on *l* is:

$$C_q + (\frac{n_1'(C - C_q)}{n_b'} - C_r') + (\frac{n_2'(C - C_q)}{n_b'} + C_r)$$

$$= C - C_r' + C_r \leq C$$

After the second token cycle, the entire residual bandwidth is taken up by the bottlenecked flows and thus the total rate becomes :

$$C_q + (\frac{n_1'(C - C_q)}{n_b'} - C_r') + (\frac{n_2'(C - C_q)}{n_b'} + C_r')$$

$$= C - C_r' + C_r' = C$$

Hence the claim.

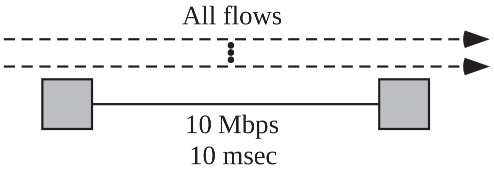


Fig. 4. Network topology used in simulations experiments 1-3

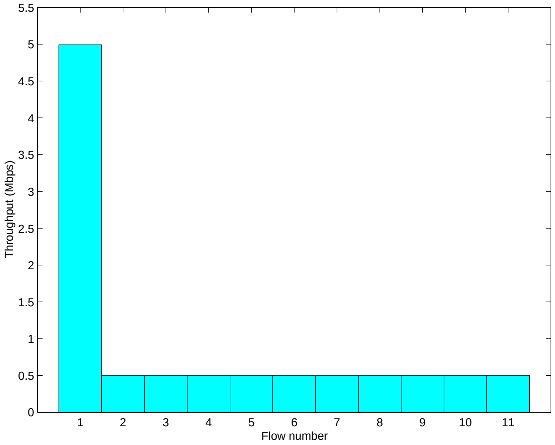


Fig. 5. Throughputs of 1 reserved flow and 10 UDP flows sharing a single link

IV. Simulation Experiments

We have carried out a number of simulation experiments to study the performance of the EQOS protocol. The primary objective of the simulations was to determine whether EQOS is able to provide rate guarantees to reserved flows in the face of competing best effort flows. Another objective was to study the impact of EQOS on best effort flow throughput and delay. All simulations were performed using a modified version of the UC Berkeley ns-2 simulator [11].

The first set of simulation experiments relies on the simple network model shown in Figure 4. The single link in this model is terminated by two edge routers and has a bandwidth capacity of 10Mbps and a propagation delay of 10 msec. Among the flows sharing this link are a single reserved flow and 10 best effort flows.

In the first experiment, the source of the reserved flow (flow 1) has installed a reservation of 5 Mbps and transmits at this rate. The best effort flows (flows 2-11) are generated by UDP sources that transmit at rates of i Mbps, where i is the index number of the flow. The results of the simulation, shown in Fig.5, indicate that the reserved flow (flow 1) obtains a throughput of 5 Mbps, while the best effort flows share the remaining bandwidth equally and obtain throughputs of approximately 0.5 Mbps each.

In the second experiment, the same network model is used, but the sources of the best effort flows use TCP

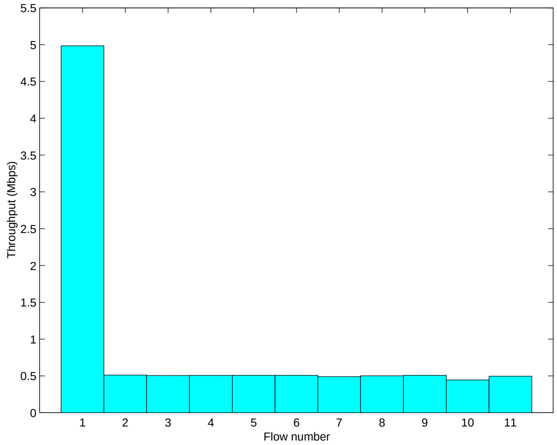


Fig. 6. Throughputs of 1 reserved flow and 10 TCP flows sharing a single link

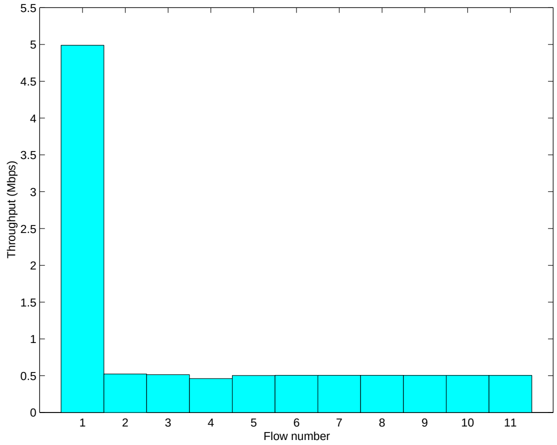


Fig. 7. Throughputs of 1 reserved flow, 5 UDP best effort flows, and 5 TCP best effort flows sharing a single link

Reno instead of UDP. The TCP sources are configured to be persistent; that is, they transmit data whenever allowed by the transport protocol. The results of this experiment are shown in Figure 6. As in the previous experiment, the reserved flow (flow 1) achieves a throughput of 5 Mbps, while the best effort flows achieve equal throughputs of approximately 0.5 Mbps.

The third experiment again uses the network model in Figure 4. It is designed to explore whether EQOS can protect best effort TCP flows from best effort UDP flows whose sources do not back off in the face of congestion. Flow 1 is a reserved flow with a reservation of 5 Mbps on the link, flows 2-6 are best effort UDP flows that transmit at rates of i Mbps, where i is the flow's index number, and flows 7-11 are best effort TCP flows. The results shown in Figure 7 indicate that the shares of link bandwidth allocated to the best effort TCP and UDP flows are nearly identical, while the reserved flow continues to obtain its reserved allocation of 5 Mbps.

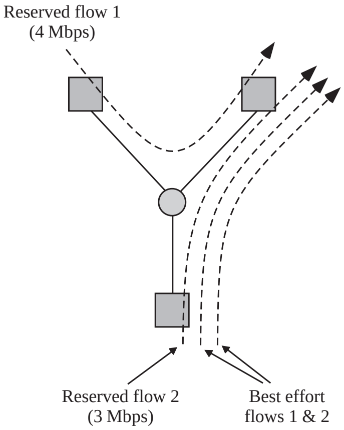


Fig. 8. The star model used in experiment 4

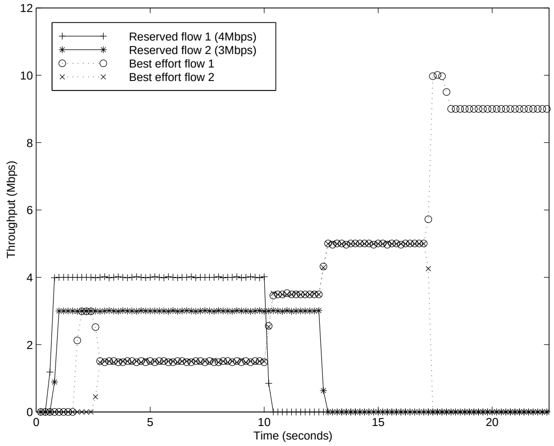


Fig. 9. Simulation results for experiment 4

In experiment 4, we study how effectively EQOS is able to allocate bandwidth on links shared by flows from multiple ingress routers. We consider the simple star topology of Figure 8 in which all links have capacities of 10 Mbps and propagation delays of 10 msec. Two reserved flows and two best effort UDP flows enter the network from different ingress routers and depart the network through a common egress router. The results of this simulation are shown in Figure 9. At time $t = 0.5$ seconds, reserved flow 1 initiates transmission and requests a reservation of 4 Mbps. Shortly thereafter, the reservation is accepted and the flow is able to obtain its desired throughput. At time $t = 0.7$, reserved flow 2 initiates transmission and requests a reservation of 3 Mbps, which it obtains. At this point, the shared bottleneck link has 3 Mbps remaining for use by other flows. At time $t = 1.5$, the source of best effort flow 1 begins transmitting at a rate of 9 Mbps. Since only 3 Mbps of bandwidth is available on the bottleneck link, the flow's ingress router limits the flow's ingress rate to 3 Mbps. At time $t = 2.5$, the source of best effort flow 2 begins transmitting at a rate of 8 Mbps. Now there are two best

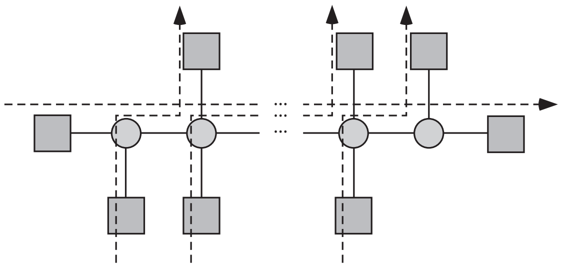


Fig. 10. The parking lot topology

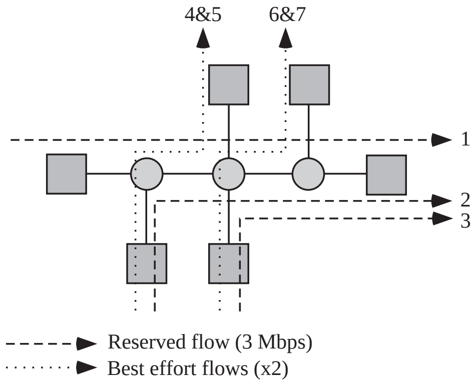


Fig. 11. The parking lot model used in experiment 5

effort flows sharing the available bandwidth on the bottleneck link, so each is rate limited to 1.5 Mbps. At time $t = 10$, reserved flow 1 releases its reservation and stops transmitting, allowing each of the best effort flows allocate an additional 2 Mbps shortly thereafter. At time $t = 12.5$, reserved flow 2 releases its reservation and stops transmitting, allowing the best effort flows to capture yet an additional 1.5 Mbps. Finally, best effort flow 2 ends transmission at time $t = 17$, and the remaining best effort flow is able to transmit at its sending rate of 9 Mbps. The small surge in the flow's throughput at $t = 17$ is caused by the emptying of router buffers.

For the remaining simulation experiments, we consider several variants of the parking lot topology shown in Figure 10. To evaluate the performance of reserved flows when competing on multiple bottleneck links with other reserved and best effort flows, we begin with the particular case of the parking lot model shown in Figure 11. In this model, there are 3 reserved flows, labeled 1-3, each of which reserves 3 Mbps of bandwidth and begins transmitting at time $t = 1$ seconds. There are also 4 best effort UDP flows, labeled 4-7, each of which starts transmitting at time $t = 2$. Figure 12 shows the results of this simulation experiment for three selected flows. The reserved flows achieve their desired throughputs of 3 Mbps, meeting their reservations regardless of the fact that several best effort flows compete with them on multiple links. The best effort flows also perform as expected, allocating approximately half of the available bandwidth on their bottleneck links.

In the final simulation experiment, we evaluate the responsiveness of the EQOS protocol as network size

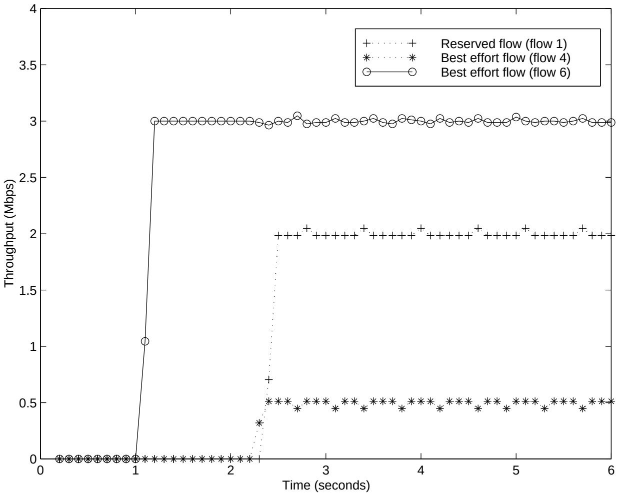


Fig. 12. Flow throughput for the parking lot model

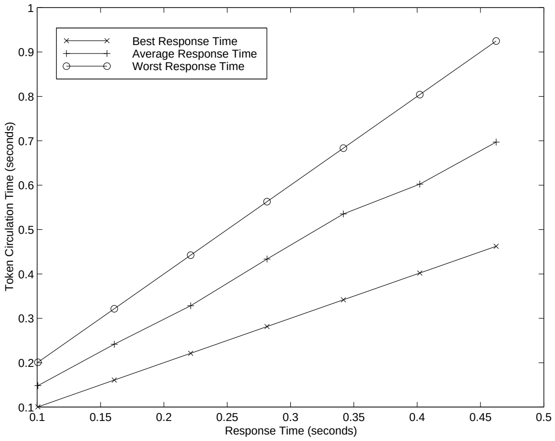


Fig. 13. Reservation response times with increasing token circulation times

and end-to-end delay increase. In order for a reserved flow to obtain its desired bandwidth, the EQOS protocol first needs to update the token and allow it to fully circulate the edge routers. In the worst case, this delay may be as great as twice the token circulation time. The worst case scenario occurs when a reservation message arrives at an ingress router just after the ingress router has passed the token to another edge router. The ingress router must then wait for the token to return, update the token's contents, and allow the token to fully circulate once more. The parking lot topology shown in Figure 10 is used in this experiment since it has the property of a particularly large token circulation time; in a single circulation of all edge routers, the token must traverse every link twice. All links in the network are assigned capacities of 10 Mbps and propagation delays of 10 msec.

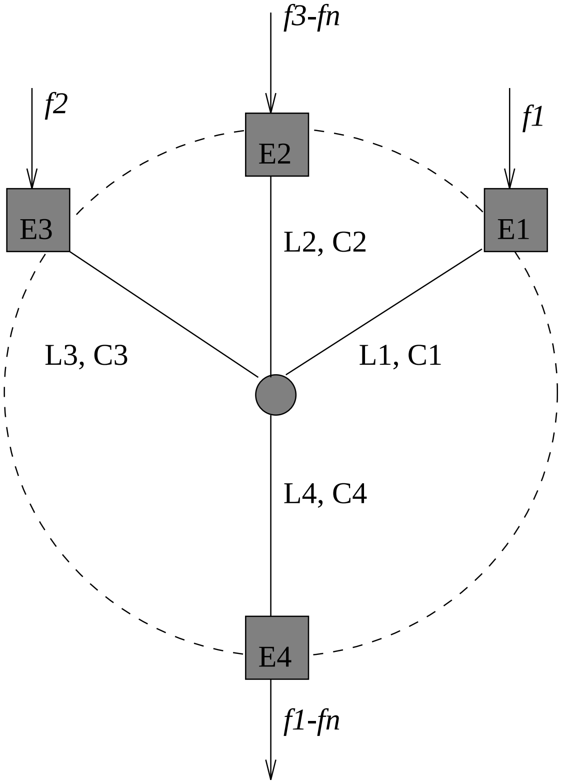


Fig. 14. An example illustrating the unfairness of EQOS.

The experiment is conducted by randomly initiating reserved flows at each of the ingress routers. When an ingress router receives a reservation message, it starts a timer. After the ingress router has updated the token and fully circulated it among the other edge routers, the flow's timer is examined to obtain the reservation response time. To vary the token circulation time, the number of core routers is increased from 2 to 8. Figure 13 shows the average, best case and worst case reserved flow response times for increasing token circulation times. We find that the average response time generally falls half-way between best and worst case scenarios. This suggests that EQOS should be used only in networks where the token circulation time is reasonable; it should not be applied on an end-to-end basis in the Internet.

V. Fairness

It is a well-accepted fact that providing some measure of fairness to competing flows is desirable. It has been shown that fair-queuing combined with leaky bucket policing, can be used to provide statistical guarantees. Moreover, it protects the compliant flows from the misbehaving ones and provides lower delays to flows using not more than their fair share of bandwidth. In this section we evaluate the degree of fairness of our protocol; how fairly is the bandwidth left after the allocation to reserved flows, allocated among competing best effort

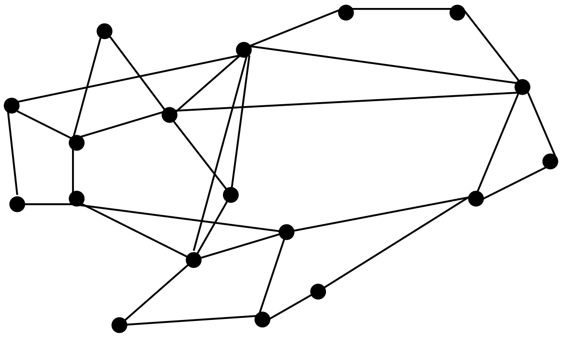


Fig. 15. A typical ISP topology.

flows.

EQOS divides the bandwidth among the best effort flows using "best share" mechanism. A flow gets its fair share on its bottleneck link and on the non-bottleneck links it indicates the residual bandwidth of its fair share. The residual bandwidth is claimed by the flows which encounter the token and which can utilize this bandwidth to increase their rate. We compare this best share allocation with max-min fairness.

Fig. 14 shows an example in which the allocation is significantly different from max-min fair allocation. E1, E2, E3 and E4 are edge routers connected in a star topology to a core router with links L1, L2, L3 and L4 having available capacity C1, C2, C3 and C4. Assume that the token circulates in order E1-E2-E3-E4. Suppose that C2 is approximately zero and C1 and C3 are greater than C4. Consider *n* flows, *f1 …fn,* of which *f1* follows the path L1-L4 and *f2* follows the path L3-L4. The remaining n-2 flows, *f3 …fn,* follow the path L2-L4. The fair share of flows *f3 …fn* is the entire bandwidth on L2 and $\frac{(n-2)C4}{n}$ on L4. Since these flows are bottlenecked by L2, the total residual bandwidth on L4 is approximately $\frac{(n-2)C4}{n}$ (since the available bandwidth on L2 is nearly zero). This entire bandwidth is claimed by *f2* and thus it gets a bandwidth of approximately $\frac{(n-1)C4}{n}$ on L4. *f1* gets its fair share of $\frac{C4}{n}$. A max-min fair allocation in this scenario would have been approximately $\frac{C4}{2}$ for both *f1* and *f2* and approximately zero for the remaining flows. This shows that the best share fairness of EQOS can be significantly different from max-min fairness.

However, this is a contrived case to show that EQOS is not max-min fair. We simulated the case where the best effort flows have randomly selected source and destination on a typical ISP topology shown in Fig. 15 (taken from [10]) and we varied the number of flows from 200 to 1000. Fig. 16 shows the average absolute percentage by which the actual allocations differed from the max-min allocation. We see that the difference is typically less than 20% averaged over all the flows. This indicates that EQOS is not too unfair. We note that in this paper we are not trying to solve the problem of fairness among best effort flows. The fairness comes into picture just because it seems to be the best way of achieving distributed rate control while simultaneously

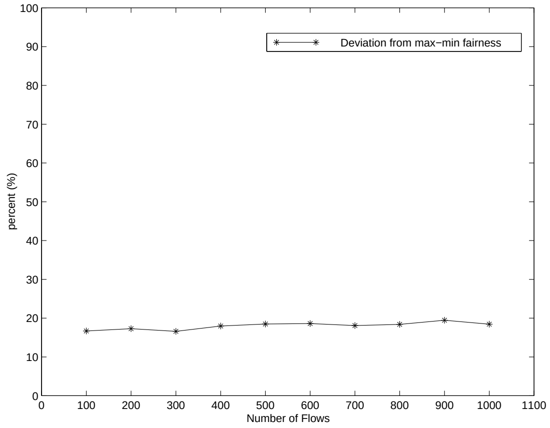


Fig. 16. Percentage deviation of best share fairness from max-min fairness with increase in number of flows.

ensuring that best effort flows don't starve and network utilization doesn't suffer.

VI. Discussion

In the EQOS protocol's description, we have left several important issues unaddressed. In this section, we briefly address these issues and propose several possible mechanisms to deal with them.

A. Selection of the Token Circulation Path

Thus far, we have assumed that tokens circulate edge routers on a predefined path. As shown by the simulation results, the response time to reservation requests is low so long as the token circulation time remains reasonable. Hence, the token circulation path should be chosen to minimize the token circulation time. The problem of selecting an optimal token circulation route is identical to the well-known traveling salesman problem, which is known to be NP-complete [12]. For small enough domains, the traveling salesman problem can be solved by brute force. For larger domains, however, a heuristic may be necessary to achieve a near-optimal circulation path. One such heuristic is the Christofides 1/2-approximation algorithm [13].

The token circulation path may be selected statically, in which case it is configured once by the network manager, or dynamically, in which the path may change as the network's topology changes. A dynamic algorithm is more robust, because network links often fail. We leave for future work the development of a dynamic token circulation path selection algorithm.

Since token passing is critical to the EQOS protocol, a small amount of bandwidth should be reserved on the token circulation path in order to reduce the likelihood that the token is discarded or buffered for long periods of time. The edge routers should also preempt the processing of other packets when a token arrives in order to minimize the token circulation time.

B. Regeneration of Lost Tokens

We have also assumed that tokens are never lost by the network. However, this is not a safe assumption. If a router fails or a buffer overflows, the token may not continue to circulate, in which case the EQOS network will eventually come to a halt. What is needed is a mechanism to regenerate a token after a particular time-out interval.

To support token regeneration, one edge router may be designated as the token monitor. Every time the monitor receives a token, it stores the token's state, increments a sequence number in the token, and initiates a timer. If the monitor fails to receive the token again before the timer expires, it regenerates the token, sets the token's sequence number to the last observed token sequence number, and inserts the token state carried by the last observed token. When other edge routers obtain the regenerated token and notice that the sequence number has not incremented, they reissue any reservations or update any flow counts that were placed in the previous token.

C. Handling Link and Router Failures

When a router or a link fails, the link state updates will reflect it, and the next edge router to receive the token updates its contents, removing the affected links. As edge routers receive the updated token, they recompute routes for all affected flows and update the token's contents accordingly. After one circulation of the token, all edge routers become aware of the new topology and have adjusted their flow rates to take it into account.

D. Flow State Expiration

When an ingress router creates state for a new reserved or best effort flow, it must also make provisions for removing that state. Otherwise reserved flows that do not issue tear-down messages may hold onto reserved bandwidth indefinitely, and best effort flow counts may continue to accumulate needlessly. To prevent this from happening, ingress routers rely on a soft-state mechanism. At the creation of a new flow, ingress routers initiate a timer, which is reset whenever a new packet from the flow arrives. If a flow does not generate any packets before the timer expires, then its resources are freed by updating the contents of the next arriving token. In the case of a reserved flow, this entails updating the available bandwidth on each of the affected route's links. In the case of a best effort flow, the flow counts, residual bandwidth and claimed residual bandwidth of links on the affected route are updated.

E. Handling Short-Lived Flows

The best effort traffic controlling mechanism of EQOS relies on the ingress router reporting a new best effort flow in the token. It may seem at an initial glance that the network utilization would suffer if the ingress router reports a flow which is short-lived. However, this is not true. EQOS uses a fair rate scheduler to police an active path. A feature of fair rate schedulers is the fact that they are work-conserving. Although the short-lived flow dies out quickly, the other flows sharing the path, utilize the link. This may lead to a little higher degree of unfairness for the duration of a few token circulation cycle but not under-utilization. However, if this slight unfairness is of concern, a heuristic that can be applied is to report a best effort flow only after it has lasted for 1 token circulation cycle. This would eliminate the reporting of flows which last for a short duration. This would significantly reduce the unfairness that might be created by short-lived flows.

VII. Conclusion

We have presented the Edge-assisted Quality of Service protocol, a distributed resource reservation protocol that provides strict guarantees to application flows and supports fair allocations of bandwidth to best effort flows. It also ensures that no matter how much traffic arrives at the network border, the resources assigned to reserved flows are not affected. To provide bandwidth guarantees, EQOS uses a combination of RSVP signaling, edge router token exchange, and route pinning. The EQOS protocol's primary distinction is the fact that it is implemented entirely in edge routers and requires no changes to core routers. This makes the EQOS protocol scalable and simple to deploy on a network-by-network basis within the Internet. Furthermore, since it relies on RSVP signaling, an EQOS network can interoperate seamlessly with an RSVP-based non-EQOS network.

Another point that merits consideration is that the token-based approach of EQOS can be easily altered to serve as a distributed bandwidth brokerage mechanism for differentiated services networks. Since EQOS is designed to control admission in a distributed manner according to the policy of the network domain, any network requiring a distributed bandwidth broker can tailor the EQOS algorithm to meet its particular requirements.

References

[1] C. Topolcic, "Experimental Internet Stream Protocol: Version 2 (ST-II)," RFC 1190, IETF, October 1990.

[2] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new Resource ReSerVation Protocol," *IEEE Network Magazine*, pp. 8–18, September 1993.

[3] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210, IETF, September 1997.

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

[5] I. Stoica and H. Zhang, "Providing Guaranteed Services Without Per Flow Management," in *Proc. of ACM SIGCOMM*, September 1999, pp. 81–94.

[6] J. Moy, "OSPF Version 2," RFC 2178, IETF, April 1998.

[7] E.C. Rosen and A. Viswanathan, "Multiprotocol Label Switching Architecture," Tech. Rep. Internet draft (work in progress), IETF, August 1999.

[8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a Fair Queueing Algorithm," in *Proc. of ACM SIGCOMM*, 1989, pp. 3–12.

[9] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control – the Single Node Case," *IEEE/ACM Transactions on Networking*, pp. 344–357, June 1993.

[10] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi, "Quality of Service Based Routing: A Performance Perspective," in *Proc. of ACM SIGCOMM*, September 1998, pp. 17–28.

[11] LBNL Network Research Group, *UCB/LBNL/VINT Network Simulator - ns (version 2)*, http://www-mash.cs.berkeley.edu/ns/, September 1997.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Fransico, 1979.

[13] N. Christofides, "Worst-case Analysis of a New Heuristic for the Travelling Salesman Problem," Tech. Rep. 388, Carnegie-Mellon University, Graduate School of Industrial Administration, 1976.