Distributed Admission Control to Support Guaranteed Services in Core-Stateless Networks

Sudeept Bhatnagar and Badri Nath
Dataman Lab, Dept. of Computer Science
Rutgers University
Piscataway, NJ, USA
Email: {sbhatnag,badri}@cs.rutgers.edu

***Abstract*— The core-stateless service architecture alleviates the scalability problems of the integrated service framework while maintaining its guaranteed service semantics. The admission control methods proposed to support core-stateless guaranteed services have significant drawbacks. We propose a scalable and robust distributed admission control architecture to support core-stateless guaranteed services. Our architecture maintains high network utilization while ensuring that resources are not over-allocated. In our architecture, admission control is performed at the ingress edge routers of a request on an edge-to-edge path basis. A token-passing mechanism is used as the resource management framework. The token helps in dynamic and fair division of bandwidth and allows completely distributed resource allocation on a link unless it is close to saturation. The edge routers co-operate to provide fault tolerance effectively acting as a resilient overlay network. Our admission control framework can support statistical guarantees and diffserv architecture's premium service as well. The resource management part of our architecture is well-suited to aid QoS routing algorithms. Analytical and simulation results are presented to show the effectiveness of our architecture.**

***Index Terms*— Distributed Admission Control, Core-stateless Guaranteed Services.**

I. Introduction

IntServ [1] and Diffserv [2] are the two most prominent architectures to support Quality of Service (QoS) in Internet. The *guaranteed service* [3] model of Intserv can provide per-packet delay guarantees but is not scalable because of its requirement of per-flow state maintenance at all routers. Diffserv is scalable because it does not maintain per-flow states in core routers, however, it can not guarantee per-packet delays but only guarantees per-flow bandwidth [4]. In [5], a novel core-stateless architecture was proposed which retained the guaranteed service semantics of Intserv while having scalability of Diffserv.

Like any other approach to provide QoS guarantees, core-stateless architecture also requires an admission control framework so that network resources are not over-allocated. Any admission control method to support guaranteed services in core-stateless architecture should have the following properties:

- *Scalability:* The mechanism should scale in number of flows and number of routers since scalability is the motivation behind the core-stateless approach in the first place.
- *Zero False Positives:* Since the aim is to support guaranteed services, a request should only be admitted if there is sufficient bandwidth to support it along its path.
- *High Network Utilization:* Ideally the mechanism should be able to allocate the entire link capacity if need arises and still maintain the zero-false-positives property.
- *Robustness:* The architecture should be able to gracefully handle any malfunctions like node failures, link failures and partial reservation failures.

As described in section VI(Related Work), the admission control methods proposed in literature are found lacking on one or more of these properties and hence are not suitable candidates to support core-stateless guaranteed services. The non-availability of such a solution is because the problem of performing distributed admission control is inherently hard. The following observation made in context of core-stateless admission control by Stoica and Zhang [5] sums it up perfectly: *"Maintaining consistent and dynamic state in a distributed environment is itself challenging. Fundamentally, this is because the update operations assume a transaction semantic, which is difficult to implement in a distributed environment."*

In this paper, we propose a new distributed, scalable and robust method of performing admission control to support core-stateless guaranteed services. To the best of our knowledge, our architecture is the first fully distributed architecture which decouples control plane from core routers and supports core-stateless service guarantees. It maintains the core-stateless semantics by performing admission control at edge routers on an edge-to-edge path basis. The basic framework of this architecture is provided by our prior work [6] where we used a token-passing mechanism to provide per-flow bandwidth guarantees without *modifying the core routers at all*. Essentially, the underlying philosophy is that of *co-operative networking* highlighted in context of application-level co-operation for content distribution in [7]. In our case, the edge routers co-operate for resource management, admission control and fault-tolerance.

The edge routers use a token-passing mechanism as a basis for resource management. All edge routers allocating resources on a link are allowed to allocate resources in parallel until some edge router senses imminent over-allocation. In such

This research work was supported in part by DARPA under contract number N-666001-00-1-8953

a case, the link is *marked* and the admission control on the link is controlled using the token. The scalability of our architecture emerges from the fact that no co-ordination for resource allocation is needed unless a link is very close to saturation. In case the token controlled regulation is needed, *anticipatory reservation* is used to amortize the delay in responding to requests. We prove that our admission control algorithm has the zero-false-positives property (which is a must for supporting guaranteed services). We show that our architecture is highly scalable, robust and maintains high link utilization.

Although our aim is to provide distributed, scalable and robust admission control framework for core-stateless guaranteed services, our architecture can be used in conjunction with other QoS approaches as well and thus can support service semantics other than guaranteed services. Moreover, the resource management component of our architecture provides a scalable way to provide accurate link-state information to QoS-routing algorithms operating in a domain.

II. Network Model & Problem Statement

We assume a network domain to be a collection of edge and core routers. The edge routers are the ingress and egress points for all traffic using the network. The core routers switch the traffic among the routers of the domain. The network uses a link state routing protocol like OSPF [8] so that the topology information is available at all edge routers. This is necessary because the edge routers perform admission test and reserve bandwidth on entire *edge-to-edge paths* rather than using the conventional hop-by-hop reservation semantics. We assume that some route-pinning mechanism like MPLS [9] or IP source routing is deployed in the network for intra-domain route pinning (which is a must for guaranteed services). The reservation request is signaled using a simple overlay-based version of RSVP as described in [6] or [5]. Effectively, both these mechanisms ensure that a flow treats the ingress to egress path in a domain as a *virtual link*.

In this framework, our aim is to define a method of performing admission control and reserving bandwidth on entire edge-to-edge paths of flows. On receiving a reservation request, the ingress edge router performs admission test on ingress-to-egress path for the flow and if the flow is admitted, it reserves the required amount of bandwidth on *all* links along the path. The admitted flow is pinned to the assigned path. This mechanism of admission control and bandwidth allocation would provide per-flow guarantees when used in conjunction with different core-stateless scheduling mechanisms [5], [10], [11].

The key to any admission control mechanism to support guaranteed services is to have the zero-false-positive property. We prove that this property holds in our architecture. Other measures to test the performance of our architecture are:

1) *Response Time:* The time it takes for the network to respond to a reservation request. The response may be positive or negative depending on the network state perceived by the ingress edge router. Note that the response time is intrinsically linked to the scalability of a mechanism.
2) *Utilization:* The fraction of link capacity that can be allocated at a given time. From an ISP's point of view, utilization is the most important factor. A good admission control mechanism should keep utilization at a high level.

Response time is the main criterion that concerns the end user. It is noteworthy that to have the best possible response time, an edge router could instantly admit or reject the request without taking into account available resources. The objectives of maintaining high utilization and zero-false-positives guard against being too pessimistic or too optimistic while taking an admission decision.

III. Distributed Admission Control

In this section, we describe how the ingress edge routers co-operate to perform admission control. First we give the basic technique for which we prove the zero-false-positives property. The basic method has high utilization and is scalable in number of flows because the requests are handled entirely on ingress edge routers. However, it has limited scalability in number of edge routers and is not robust. Then we describe two enhancements, called anticipatory reservation and link marking, to the basic mechanism to make the admission control scalable in number of edge routers and robust against various types of malfunctions while maintaining the other properties. The resource management framework of the basic admission control technique is retained in the anticipatory reservation and link marking techniques and hence is described here first.

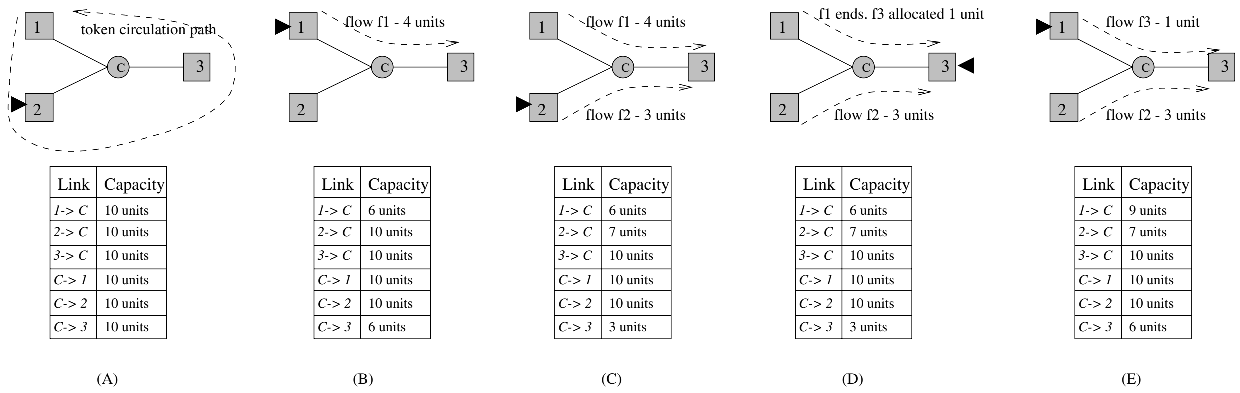
A. Basic Admission Control

The basic admission control technique relies on a simple token passing protocol. In [6], we described a similar token-passing protocol for providing per-flow bandwidth guarantees *without modifying any core routers*[1]. This paper shows how the framework can be modified for scalable admission control in core-stateless networks.

1) Resource Management: We treat the edge routers of the domain as a *logical token ring*. A special packet, called the *token*, is circulated around the logical ring in a pre-determined sequence. Initially, the token contains the capacities of all the links the network[2]. Thus, if the network has m links $l_1, l_2, \ldots, l_m$, the content of the token is essentially a set $C = \{C_1, C_2, \ldots, C_m\}$, where C_i represents the available capacity of link l_i. On receiving the token, an edge router updates its view of the network state using the contents of the token and updates the contents of the token using the information about the flows it has admitted since the last

[1] In [6] we assumed an architecture where the domain had legacy core routers which were unable to distinguish between best-effort and QoS packets and only had FIFO scheduling. As a result, the only possible way of service differentiation was to use co-operative flow control at edge routers. Here we are not concerned with best-effort flows and also assume some specialized mechanism deployed in the network, e.g. core-stateless scheduling, in order to provide the desired service. The problem we address in this paper, is to provide distributed, scalable and robust admission control in such networks.

[2] There can be many tokens with each carrying a subset of the links of the domain. However, each link should be part of only one token.



Link	Capacity
1-> C	10 units
2-> C	10 units
3-> C	10 units
C-> 1	10 units
C-> 2	10 units
C-> 3	10 units

(A)

Link	Capacity
1-> C	6 units
2-> C	10 units
3-> C	10 units
C-> 1	10 units
C-> 2	10 units
C-> 3	6 units

(B)

Link	Capacity
1-> C	6 units
2-> C	7 units
3-> C	10 units
C-> 1	10 units
C-> 2	10 units
C-> 3	3 units

(C)

Link	Capacity
1-> C	6 units
2-> C	7 units
3-> C	10 units
C-> 1	10 units
C-> 2	10 units
C-> 3	3 units

(D)

Link	Capacity
1-> C	9 units
2-> C	7 units
3-> C	10 units
C-> 1	10 units
C-> 2	10 units
C-> 3	6 units

(E)

Fig. 1. An example illustrating the working of basic admission control mechanism. There are 3 edge routers labeled $1, 2, 3$ and one core router labeled C. All links are assumed to have 10 units of bandwidth. The token circulates around the network in the order 1,2,3 as shown in (A). The token contents are shown in the tables in each of the figures as it is circulated. The black arrowhead represents the edge router which currently has the token and the token contents are shown just after the router has forwarded the token. (A) Token is at router 2. A request for 4 units of bandwidth for flow $f1$ going to router 3 arrives at router 1. The request is enqueued at router 1. (B) Router 1 receives the token and finds sufficient bandwidth in links $1 \longrightarrow C$ and $C \longrightarrow 3$. Flow $f1$ is admitted, the token contents are updated as shown and passed to router 2. Request for flow $f2$ arrives at router 2 with destination router 3 and requiring 3 units of bandwidth. The request is enqueued at router 2. (C) Router 2 receives the token and finds sufficient bandwidth on links $2 \longrightarrow C$ and $C \longrightarrow 3$. It admits $f2$ and updates the token contents. (D) Token is at router 3. Flow $f1$ terminates and another flow $f3$ arrives at router 1 requesting 2 units of bandwidth with destination router 3. Router 1 immediately admits $f3$ allocating it some bandwidth released by $f1$. (E) Router 1 receives the token and updates its contents by adding the bandwidth released by $f1$ minus the bandwidth used by $f3$.

arrival of the token. It then sends the token to the next edge router in sequence.

2) Admission Control: Clearly, while state at an edge router becomes consistent within one token circulation delay, over-allocation of link bandwidth is still possible when different edge routers allocate bandwidth on the *same* link at the *same* time. To alleviate this problem, the basic admission control mechanism uses the token as a *semaphore*. The edge routers treat the network state as a *common resource* and the token governs which edge router has *the right to change the state of that resource*. Thus, on receiving a reservation request, an ingress edge router does not take admission decision immediately. Instead, the request is buffered till the router receives the token. Then the router performs admission test using the network state contained in the token. If the flow is admitted, it subtracts the reserved bandwidth from the links on the flow's path[3]. This is repeated for all waiting requests. Once all requests have been served, the token is sent to the next router. However, a request can be immediately admitted if some flows terminate after the token was last released and if the terminated flows have left sufficient bandwidth on the ingress-to-egress path of the new request. The basic admission control method is illustrated in fig 1.

3) Zero-False-Positives Property: Zero-false-false positives property is a must to support guaranteed services. A simple proof that the property holds in the basic admission control mechanism is given below.

Theorem 1: The basic admission control method has the zero-false-positives property.

Proof: The proof follows by induction on network state. Consider an edge router which receives the token indicating an available capacity of C_i on a link l_i. If the edge router allocates some capacity c to a flow on l_i, then C_i should be at least equal to c and available capacity on l_i is set to $C_i - c$ (which is non-negative). Thus, a link's available capacity is always non-negative when a token is received at an edge router and when it leaves it for the next router in the ring. Since a flow is admitted only if there is enough bandwidth on the path it follows and the available bandwidth on the path is never less than the actual available bandwidth, an admitted flow cannot be a false positive.[4] ■

4) Properties of Basic Technique: As proven, the basic mechanism has the zero-false-positives property and thus can support core-stateless guaranteed services. It is easy to see that the trade-off here is to be *absolutely sure* about the available bandwidth (resulting in zero-false-positives) while having a worst case response time equal to one token circulation cycle delay. Note that, the token circulation delay could be kept to a minimal level by treating the token as a high priority packet and thus letting the network size govern the token circulation delay. For large networks, the token circulation duration could be large enough to make the response time to requests unacceptably high. Implicitly this means that the basic admission control mechanism is not scalable in number of edge routers thus limiting its usefulness only to small domains.[5] As observed in context of IP telephony, it is meaningless to try and have a call-setup delay which is imperceptibly small for humans [13].

Another feature of this mechanism is that it results in a high utilization. A request is enqueued at its ingress edge

[3] If the network has the capability of having multiple paths between an ingress-egress pair (for example, using MPLS), the flow can be assigned to any path which has bandwidth to support it.

[4] The actual available capacity might be more than that reported in the token because some reserved flow might have terminated and the released bandwidth may not have been accounted for in the token at a given instant.

[5] One measure of acceptable delay for response times might be the standard target response times required by the ITU standards for telephony [12]. The average target values are 3 seconds, 5 seconds and 8 seconds for local, toll and international calls respectively with the corresponding 95th percentiles set at 6.0, 8.0 and 11.0 seconds respectively.

router until the router gets the token. If there is sufficient bandwidth on a path to the request's destination egress router, the request is admitted. Thus, the only case when a request is denied while there are sufficient resources is when enough resources are released on the flow's bottleneck link *during the last token cycle* and the corresponding ingress routers have not yet updated the token since their termination. Clearly, if the token circulation time is small, the number of such *false negatives* will be small. Moreover, the ingress edge routers can allocate bandwidth to new requests out of the bandwidth released by terminated flows.

Finally, the robustness of this simple protocol is better than that of a centralized broker based system. We treat the token to be the highest priority packet in the network to limit the circulation delay, making the token being dropped an improbable scenario. Token loss due to equipment failure can still occur but it is easier to deal with (as described in section III-D) than a centralized broker node failing.

5) The Need for Improvement: In a larger perspective, the admission decision being taken only at *one* edge router at a time is essentially the same as centralized admission control. Instead of request being sent to the centralized broker, the perfect knowledge of the broker comes to the corresponding edge router. Thus, while this method exhibits the zero-false-positive and high utilization properties of a centralized broker, its scalability is limited because of the response time being proportional to the token circulation delay. Furthermore, while the single point of failure problem is alleviated and recovery process simplified, a token loss due to equipment failures could lead to service unavailability. In subsequent sections, we show how we make our mechanism highly robust and scalable while maintaining its high utilization and zero-false-positives properties.

B. Anticipatory Reservation

A simple enhancement to the basic technique can reduce the response time significantly while maintaining the zero-false-positives property. There are two key observations with respect to the basic mechanism that lead to this enhancement:

- The prohibiting portion of the basic technique is that the edge routers perform admission tests sequentially. Allowing them to perform admission tests on the requests as they arrive, in parallel, would reduce the response time significantly.
- The routers are allowed to allocate bandwidth to flows from the bandwidth *released* by terminated flows. Thus, increasing the size of this available bandwidth pool at each edge router would allow them to allocate more bandwidth to new requests without having to wait for the token.

These observations lead to a new model which we call the *anticipatory reservation model*. Under this model, when an edge router receives the token, along with deciding the admission status of the waiting requests, it reserves a small fraction of bandwidth on all the network links in anticipation of new requests arriving. This bandwidth is also subtracted from the available bandwidth values reported in the token. Conceptually, this bandwidth can be considered allocated to a dummy flow which terminates immediately after the token is sent from the router. Clearly, the zero-false-positives property holds with the proof remaining identical to that for the basic mechanism.

The trade-off here is between response time and utilization. When an edge router reserves extra bandwidth, the new flows whose arrival it anticipated, may not arrive at all. Instead a new request arriving at another edge router might have to be rejected because it does not see enough resources when actually the resources are available (and needlessly reserved by the first edge router). Thus, the mean response time is reduced while increasing the chances of false negatives.

This trade-off limits the amount of extra bandwidth that an edge router can reserve on a link. Clearly, this mechanism is only useful when the available bandwidth is scarce because waiting for the token in an uncongested network is meaningless. Moreover, the robustness of this mechanism is only slightly better than the basic method because token loss could still lead to service unavailability despite having ample resources available. Link marking technique is designed for admission control in such uncongested network.

C. Link Marking

When the network is uncongested, it is overkill to wait for token for admission test because resources are available with a high probability for everyone. Under such conditions, *link marking* is used along with anticipatory reservation.

The idea is to allow all edge routers to allocate bandwidth on all *unmarked* links without waiting for the token. While the token is constantly circulating to update the network state at edge routers as described in section III-A.1, it only acts as admission control semaphore for the *marked* links. Initially all links are unmarked. On sensing imminent link over-allocation, an edge router *marks* the link. Once a link is marked, only the edge router having the token can allocate bandwidth on that link. Similarly, if the available bandwidth increases beyond a threshold, the link is unmarked and is available for unrestricted allocation.

Since all edge routers can simultaneously allocate resources on an unmarked link, the response time is reduced to the lowest possible level. However, we would have a non-zero probability of false positives if the parallelism in admission control is not controlled. This could happen when edge routers see a sufficient amount of bandwidth to admit all the requests that they get but as a whole the number of requests exceeds the available bandwidth.

Thus, there are two important issues to be addressed in order to use link marking technique:

1) How to control the simultaneity in allocation of resources so that the zero-false-positives can be guaranteed.
2) How does the transition between link marking and anticipatory reservation techniques take place.

1) Controlling the Parallelism: Multiple edge routers allocating bandwidth on the same link at the same time can result in violation of zero-false-positive property. In order to prevent this from happening, we need to limit the maximum allocation by an edge router on a link. It is possible to statically limit each of the N edge routers to a maximum of $\frac{1}{N}$ allocation. But static partitioning would result in under-utilization because some edge routers may not allocate resources on a link at all and the ones that do, might do so in different proportion. Hence, limiting the maximum allocation (link partitioning) should be done dynamically taking into account the actual load on a link originating from the given edge router. The following theorem allows us to do so.

Consider a set of N nodes labeled $e_1, e_2 \ldots, e_N$ having constituent-values $u_1, u_2, \ldots, u_N$ for some common variable U which is defined as $\sum_{i=1}^{N} u_i$. The nodes use token passing to update the value of U so that it reflects the recent values of all nodes. Let the token circulation be started at node e_1 going in order $e_2, e_3, \ldots, e_N, e_1$. The time from the initiation of token circulation is divided into cycles where t^{th} cycle starts at the time when the token leaves e_1 the t^{th} time and ends when e_1 receives that token after the circulation. Let u_j^t represent the value reported by router e_j in the t^{th} token circulation cycle. If the node receives U' as the value in the token in the $t+1^{th}$ token cycle, then it updates it to $U' - u_j^t + u_j^{t+1}$. Let the last value of U seen in the token at node e_i be $\widetilde{U}_i$. In these circumstances, the following theorem holds:

Theorem 2: At any given instant, $\widetilde{U}_i$ for *all* nodes e_i is at least equal to $\sum_{j=1}^{N} min(u_j^{t_j}, u_j^{t_j-1})$ where t_j is the last token circulation cycle in which node e_j reported its value.

Proof: The proof follows from the observation that $\widetilde{U}_i$ is the summation of *last N values* used to update the token contents when the token leaves router e_i. The *stalest* possible value of $\widetilde{U}_i$ at node e_i is just before arrival of the token (all other nodes will have a more recent value). Consider that the token is just arriving at node e_i in the t^{th} circulation cycle. The last $2N$ *reported* values that were used to update the token contents are $u_i^{t-2}, u_{i+1}^{t-2}, \ldots, u_N^{t-2}, u_1^{t-1}, u_2^{t-1}, \ldots, u_N^{t-1}, u_1^t, \ldots, u_{i-1}^t$. These values contain the last 2 reported values by each node (and hence also the minimum of the last two reported values of each node).

Just before the arrival of the token, the U value seen by node e_i is the sum of the second to the $(N+1)^{th}$ values in the above set of $2N$ values.

$$\widetilde{U}_i = \sum_{j=i+1}^{N} u_j^{t-2} + \sum_{j=1}^{i} u_j^{t-1}$$

$$\geq \sum_{j=i}^{N} min(u_j^{t-2}, u_j^{t-1}) + \sum_{j=1}^{i-1} min(u_j^{t-1}, u_j^t)$$

Hence, $\widetilde{U}_i$ is at least equal to the sum of minimum of last two reported values by all nodes. After e_i updates the token contents, the set of last $2N$ updates becomes $u_{i+1}^{t-2}, u_{i+2}^{t-2}, \ldots, u_N^{t-2}, u_1^{t-1}, u_2^{t-1}, \ldots, u_N^{t-1}, u_1^t, u_2^t, \ldots, u_i^t$ which leaves a similar invariant for node e_{i+1}.

■

This theorem is general and applies to *any* variable which is updated in a similar manner as the updation of U above. In our case, the value we pass around is the *request load* for that link. Request load refers to the sum of *all* reservation requests for the link, which are active or which arrived during the last estimation window. The estimation window size that we used is the mean flow duration which is calculated using exponential averaging. Note that the requests which arrived during the last window includes the requests that are queued or which were rejected. Thus, each edge router e_i has an estimate u_i of its share of the total request load on the link.

We use the above theorem in controlling parallel allocation. In our context, the theorem says that the total request seen by *any* edge router at any instant is an upper-bound on the sum of the minimum of last two reported values by *all* edge routers. Thus,

$$\frac{\sum_{j=1}^{N} min(u_j^{t_j}, u_j^{t_j-1})}{\widetilde{U}_i} \leq 1$$

for all routers e_i where t_j is the last cycle in which router e_j updated the token. Thus, at any instant, if the maximum that an edge router e_i, is allowed to reserve on the link (with capacity C) is $\frac{C.min(u_i^{t_i}, u_i^{t_i-1})}{\widetilde{U}_i}$, the link bandwidth is never over-allocated. It is important to see that the practicality of our approach arises from the fact that for computing its share of a link, an edge router just needs the last two values that it used to update the token, along with the last aggregate request load value that it received from the token.

By using the above allocation limit, we ensure two things: 1) The zero-false-positive property is maintained and 2) Within one token circulation cycle, each edge router is entitled to its *fair* share of the link (given by its fraction of request load). This fast convergence to fair division of link bandwidth is a salient feature of our architecture. Moreover, only the routers that are *using* a link have a share on it.[6]

2) Limitation of Link Marking: Clearly, under low load conditions, an edge router's share of a link would be sufficient to satisfy all the requests that arrive. Hence, we have full parallelism in resource allocation and a link need not be marked. Under heavy load, however, we may need to revert to the anticipatory reservation control. This is because even though the above theorem guarantees that the combined maximum limits of all edge routers at any instant is never more than the capacity, the theorem *does not predict the future states*. For example, consider a case where an edge router has allocated its *entire* share of bandwidth on a link. When it receives the token in the next cycle, it finds that its fair share has reduced. This could happen when another router's request load has a sudden increase causing an increase in that router's share. Thus, if the network never marks the links, the only way to have a guaranteed prevention of over-allocation is to *revoke* the bandwidth allocated. Clearly, this is not a desirable option. A graceful alternate to handle such a case is to switch to the anticipatory reservation model.

3) Switching to Anticipatory Reservation: The case of possible over-allocation described above, provides an implicit signal to move to anticipatory reservation based admission control. Note that it is likely that the above case occurs when the request load on the link is high. This is so because the allocation limit of the over-allocating router is insufficient to accommodate all its requests and at the same time leave sufficient bandwidth to release to revert to its new fair share. The

[6] In practice, if the routers do not have any request load on a link, they can use dummy request load to keep some portion of the links for their own use.

maximum allocation limits of all edge routers are allocated dynamically and fairly, which means that the request load for other routers in the network is also high. A combination of the two arguments shows that in all likelihood, the link must be close to saturation when such a possible over-allocation occurs.

Fortunately, we can have a smooth transition into the anticipatory reservation phase while still maintaining the zero-false-positive property. The router which senses the imminent over-allocation (the router which would have to revoke allocations to remain within its fair limit), serves as the starting point for anticipatory reservation phase. It marks[7] the link under consideration and substitutes the request load field of the link with its *actual allocation* on the link. The other routers seeing the link marked for the first time, add their actual allocations on the link in the field. After one circulation cycle, the link is effectively available for anticipatory reservation control. During the cycle in which this switching takes place, the routers are only allowed to allocate the bandwidth that was released by some flows which terminated during the cycle.

The transition back to link marking phase occurs when a router sees that the link usage has gone down below a certain threshold. The transition phase is the same as described for switching to anticipatory reservation.

An important point that should be noticed here is that the the token could have some links unmarked and some using anticipatory reservation. This is a key feature of our architecture because the congested links of the network form part of the same token that carries information about uncongested links as well. Thus, parallel allocation can continue on uncongested paths despite some links being congested.

D. Robustness

At an initial glance, this architecture may appear very fragile because of its susceptibility to token loss. In fact, as described in [14], this architecture of co-operating edge routers can actually be highly fault tolerant and quick to recover from network failures when it operates as a *resilient overlay network (RON)*.

We treat the token as the highest priority packet in the network effectively reserving some bandwidth along its circulation path. However, this overhead in terms of bandwidth is distributed across the entire network. This minimal overhead in making the token the highest priority packet leads to a major benefit in that it is unlikely to be dropped inside the network. Using a centralized broker node as in [11] is significantly worse because each edge router has to reliably communicate each reservation request and flow termination message to the broker. Moreover, all edge routers sending requests to the broker would lead to high overhead on the links leading to the broker node.

Having token as the highest priority packet would lead to it not being dropped. However, a router holding the token or a link carrying the token could fail. In such a case, the RON based technique of maintaining a robust overlay is useful. The edge router ring is essentially an overlay of admission control nodes. Since our purpose is maintenance of the logical ring structure, instead of pinging all edge routers, each router can periodically ping its one and two hop ring-neighbors (assuming at most one router failure at a time) to be sure that they are alive. As soon as the failure of a neighbor router is detected (for example, if k pings were unanswered), the token could be routed around the failure to the second hop neighbor.

In case the failed router had the token, there is a need to regenerate the token. This is easily accomplished using a simple technique. The router next in the token sequence to the failed router, would have detected the failure of its previous router. Thus, it can contact its two-hop neighbor previous to the failed router to check the sequence number of the last token that it had sent. If the token sequence number that neighbor sends is higher than what it had last received, it would know that the failed router took the token with it. It then creates a new token (tags it using a flag) and puts the *local* values that it last reported in the token. Seeing a new token, the other edge routers, add their last reported values (request load or actual allocation depending on whether a link is marked or not) in the token and after one token circulation cycle the token contents are consistent and the normal circulation process continues. Note that, this procedure is necessary to eliminate the resource share of the failed router and thus to ensure soft-state semantics. A similar technique is used for routing around link failures.

A significant advantage in using the link marking technique is that at any given point of time, the link bandwidth is apportioned to routers according to their fair share (based on request load). Thus, in case of token loss, while the process of regenerating the token takes place, the edge routers can continue to operate in the fair share of bandwidth that they got before the token was lost. The fast recovery from token failure ensures that unless drastic traffic changes occur within a few seconds, the routers will keep on operating within their fair shares without noticing the failure. In contrast, if the central broker node fails, the entire network's admission control mechanism fails and no requests can be processed until the node comes back up. Similarly, if a link fails, certain edge routers might not be able to communicate at all with the broker node until the network routing creates a new path between them. As shown in [14], a RON is significantly better than having to wait for routing updates to provide failure information. Moreover, having a centralized control node leads to the formation of hot spot in the network. Thus, congestion and link failures lead to unavailability problems in any architecture relying on a central node. Essentially, our architecture moves the possible failures from hardware to software. The key advantage obtained form this transformation is that if a packet is dropped, it could be regenerated, whereas a node crash requires some time (possibly human intervention) before it can come up again.

Lastly, our architecture doesn't suffer from the inconsistent state problem caused by partial reservation failures. Since we are using an overlay-based version of RSVP and the edge-to-edge path is just a virtual link to RSVP, the soft-state semantics of RSVP will ensure state consistency at the edges (the core routers are not affected by RSVP in any way).

[7] Marking requires using 1 bit per link in the token.

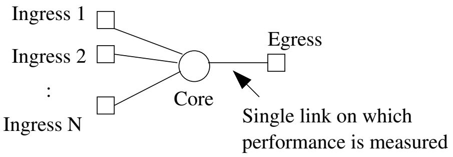


Fig. 2. The topology on which simulations are run. There are N ingress routers connected to one core router. The core router is connected to the egress router by a single link.

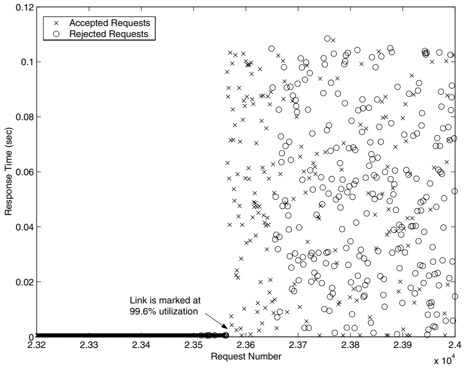


Fig. 3. The response times to individual requests around the time that the link is marked.

IV. Simulation Results

In this section, we show some simulation results showing the effectiveness of our admission control architecture. As the zero-false-positives property of our mechanism has already been proven, the simulations are primarily aimed at showing the level of network utilization and the scalability of our architecture in number of flows and number of routers.

We used a simple single link topology shown in fig. 2. A single link connects a core router and the egress edge router with capacity varying across simulations. The core router is also connected to a set of ingress edge routers (the number of edge routers is varied) with links of infinite capacity with propagation delay varying across simulations. Recall that, all edge routers treat each link as a single entity irrespective of where it is located in the network. Thus, the key criteria that governs the performance of our architecture is not the underlying topology but (1) the number of contending ingress edge routers, (2) the capacity of the link (relative to that of an average request) (3) the rate at which traffic intended for the link arrives and (4) the token circulation delay. We evaluate our architecture by varying these parameters while measuring the performance on a single link. The default values of the simulation parameters are: core-egress link bandwidth = 20000 units, propagation delay of access links (ingress-core links) = 1ms, number of requests arriving during the simulation = 40000, request arrival rate = 50 requests/sec, mean flow duration = 500 sec exponentially distributed.

For all simulations, we assume that each flow requests 1 unit of bandwidth at a time. The arrival of requests is assumed to be Poisson with mean varying over different simulations. The processing time per-request is assumed to be a constant $0.5ms$.[8] The offered load is controlled using different mean request arrival rates and different flow durations. Each simulation was run 10 times on different traffic pattern files generated with the same control parameters and the results show the mean of these runs.

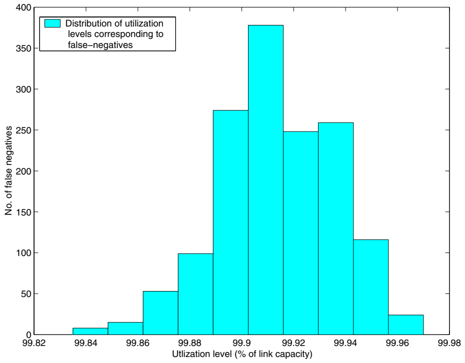


Fig. 4. The distribution of link utilization levels at the time when false negatives occur.

The first simulation intends to show the relative differences between the parallel allocation phase (where link is unmarked) and the token-controlled phase (where link is marked) of the admission control framework. For this simulation, the number of ingress routers was set to 50 with a request arrival rate of 50 request/sec (with a mean flow duration of 500 sec implying an offered load of 25000 units/sec). The response times to individual requests in our architecture are shown in fig 3. A small portion of the x-axis is shown with the center being the point where the link gets marked. The plot has two classes of points representing the response times for accepted requests and the rejected requests. As the dark line at the bottom shows, before the link gets marked, all the requests are responded to within $0.5ms$ (the minimum possible time). The response times are higher after the point when link is marked. Hence, for response times to remain small, it is essential for a link to remain unmarked for as long as possible.

Another thing evident from the figure is that the number of accepted requests after the link gets marked is relatively lower than the number of rejected requests. Note that this depends on the mean flow duration which is expected to be relatively high for flows which request QoS. Hence, in the anticipatory reservation phase, we would expect more requests being rejected than accepted and more acceptances than rejections in parallel allocation phase. This is a desirable feature because the mean response time for accepted requests would be lower than the mean response time for rejected requests. Having a higher response time for rejected requests is not worrisome because they are rejected anyway. Buffering these requests just gives them an extra chance to be accepted in case another flow terminates.

The second simulation is aimed at showing the effectiveness

[8] We ran a simple test on a 1Ghz Pentium machine running Linux 2.4.9 to perform admission test for requests in a network of 18 edge nodes with an average of 5 hops per path and found the mean time per request to be around $0.5ms$ over 10000 requests.

of our architecture in terms of utilization. The metric we chose for this purpose is the link utilization level when false negatives occur, i.e., if a request is denied even when requests are available, we would like to know how much resources were available at the time. For this simulation we set the core-egress link capacity to 20000 units and had 50 ingress routers. 40000 requests arrive at the routers with the first 20000 of them arriving at 30 requests/second with equal probability of arrival at each router. The next 20000 requests arrive at 70 requests/second with one of the routers having 10 times higher probability of being the ingress. The reason behind using such a traffic arrival pattern was to test the architecture where there is sudden traffic pattern change at the time when link is close to be marked. The histogram shown in fig 4 shows the actual link utilization levels when the false negatives occurred in the simulation. The effectiveness of our mechanism is evident from the fact that false negatives occur only after link utilization is above 99% mark. We found this to be true across different simulation scenarios with different traffic patterns. In fact, one of the key reasons to use this contrived traffic pattern was that in our simulations with constant traffic patterns, we found similar high levels of link utilization before false negatives occur.

The ability of our architecture to have high link utilizations before having false negatives is explained as follows: There are two sources which may cause false negatives: 1) Flows may have terminated leaving enough bandwidth for a new flow to be admitted, but this information is not available to the edge router at the time it takes admission decision and 2) Due to dynamic nature of request load and the finite delay in information propagation introduced by the token, the sum of shares of the link over all edge routers could be less than its capacity. The impact of first cause is limited because the ingress router of the terminated flow could immediately admit new requests that arrive at it. Thus, the likelihood of a request at another router becoming a false negative is very less. The case where the ingress router of the terminated flow *does not* have new requests and at the same time another edge router rejects a request due to unavailability of *this* freed unit of bandwidth, is likely to occur at a high utilization level anyway. The second cause of false negatives not showing a significant impact on the number of false negatives is an indication of the ability of our architecture to quickly converge to fair shares of the routers. Moreover, the buffering of requests when moving into anticipatory reservation stage eliminates its impact in the token-controlled regime. It is possible to construct pathological cases where false negatives could occur at lower utilization levels. However, in our simulations we did not see false negatives occur at below 99% utilization levels in any runs, with different network parameters, different traffic parameters and irrespective of the link marking point (the utilization level where allocation changes from parallel to token-controlled).

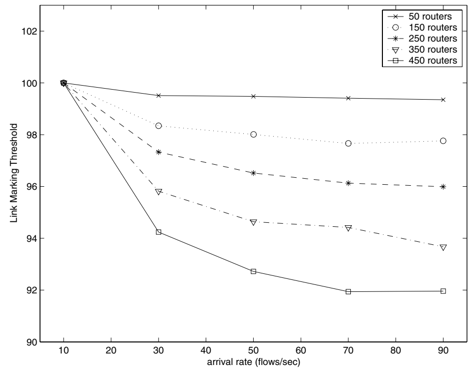


Fig. 5. The utilization levels at which the link was marked for the single link case when the ingress routers are chosen uniformly.

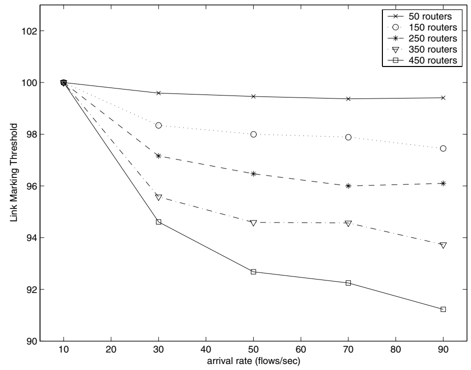


Fig. 6. The utilization levels at which the link was marked for the single link case when one of the ingress router was 10 times more likely to get a request than any other routers.

The previous simulation highlights the ability of our mechanism to keep utilization level high by not rejecting requests if there are resources to support them. Thus, minimizing response time becomes the main performance criteria to be evaluated. As is evident from the first simulation (fig 3), to have minimal response time, it is necessary for the link to remain unmarked for as long as possible. Thus, the *link marking threshold* (the utilization level at which the link is marked) is of utmost importance. Subsequent simulations aim at testing the impact of various parameters on link marking threshold.

First we test the impact of increasing number of routers contending for the link. Note that, an increase in number of routers increases the token circulation delay along with the number of contenders for the same link. The number of ingress routers is increased from 50 to 450 (in increments of 100). The number of requests was set to 40000 and the mean arrival rate into the system was increased from 10 flows arriving per second to 90 flows per second. The mean flow duration is exponentially distributed with a mean of 500 second. The results in fig. 5 are for the case when the ingress point of the flow was chosen uniformly and those in fig. 6 are for the case when one of the ingress routers had a ten times higher probability of being the ingress point than the other routers (leading to a biased traffic profile). There are several points indicated by these figures: 1) In both cases our admission control mechanism performs well by allowing the routers to allocate more than 90% of the link capacity before having to revert to anticipatory reservation. 2) Traffic bias does not seem to make any significant difference in link marking threshold indicating that our mechanism partitions links fairly

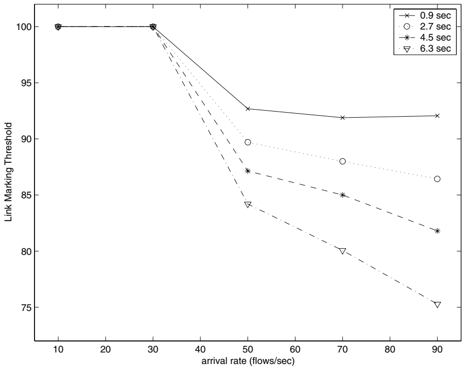


Fig. 7. The impact of increasing link delay of access links between ingress edge routers and the core routers on the utilization level at which the link is marked.

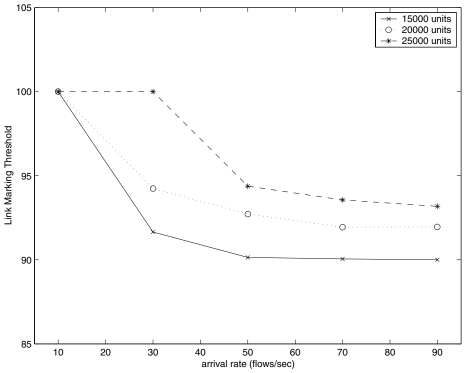


Fig. 8. The impact of increasing link bandwidth of the link between core and egress router on the utilization level at which the link is marked.

(otherwise the router getting a higher number of requests is more likely to mark the link at a lower utilization level.) 3) Link marking thresholds tend to become constant with increase in arrival rate. This is an indication that link marking thresholds are more dependent on the network characteristics than traffic characteristics. 4) These results hold even under a heavy offered load (90 flows arriving per second requiring 1 unit bandwidth with a mean holding time of 500 seconds gives roughly a 45000 units per sec load which is more than twice the capacity of the link) 5) The link remains unmarked under lighter request loads which again is an indication of the mechanism's fast convergence to fair sharing and its ability to remain in unmarked phase for sustained durations under relatively lower loads.

The next simulation result tests the impact of token-circulation delay on our architecture. For this simulation we had 450 ingress routers. The bandwidths of access links connecting ingress routers to the core routers is infinite but the link propagation delay on each link was set to $1ms$, $3ms$, $5ms$ and $7ms$ in different runs giving token circulation delays of $0.9sec$, $2.7sec$, $4.5sec$ and $6.3sec$. The other simulation parameters were set to their default values. The link marking thresholds under these conditions are shown in fig. 7. The link starts getting marked at lower utilization level with increase in propagation delay. This is expected because the number of flows arriving at the router in a token cycle increases with the circulation delay. Thus it is more likely to have used up its entire fair share and hence more likely to have the need to mark the token when its fair share shrinks. The link marking threshold reducing to lower levels indicates the need to have anticipatory reservation in order to maintain high utilization.

Finally, we test the impact of link capacity on our admission control mechanism. For this simulation, we set the core-egress link bandwidth to 15000, 20000 and 25000 units in 3 sets of simulation and test their impact on link marking threshold with 40000 requests arriving at different arrival rates. The topology has 450 ingress routers connected via infinite bandwidth link to the core and with each link having propagation delay of $1ms$. The flow durations are exponentially distributed with mean 1000 sec. The utilization level at which the link is marked is shown in fig. 8. As the available bandwidth increases, the link marking threshold increases with it. This result indicates that the link marking threshold is dependent on the ratio of available bandwidth to the total offered load on the link. This makes our mechanism fit well with the Internet architecture. As we move towards the core of Internet, the available bandwidth increases to very high levels. On the other hand, the collective offered request load also increases in the core because more routers would have flows utilizing the core. As this result suggests, despite increase in the offered load in the core, the increased bandwidth counters the possible reduction in link marking threshold. Thus, at each level of links in the Internet, the link marking threshold could be expected to remain high thus requiring infrequent use of token-controlled admission control.

The gist of all the above results is that our architecture provides the ability to have all edge routers operate in parallel over extended periods of time until a link is very nearly saturated. These results effectively show both a high utilization (when used with anticipatory reservation) and low response time (because of very high degree of parallelism). Moreover, having a high degree of parallelism along with a simple result from basic queuing theory establishes the scalability of our architecture vis-a-vis a centralized broker: N servers operating in parallel gives roughly an N times improvement in the mean response time over a single server because each of the N servers would be serving $\frac{1}{N}$ requests (if the requests are uniformly distributed) [15].

V. Discussion

In this section, we discuss possibilities and issues regarding our architecture. Primarily the focus is on how the token passing mechanism could be useful for other purposes.

A. *Support for QoS Routing*

QoS routing is used to find suitable paths for flows under the given network conditions. Assuming that the route is decided at the ingress edge router, the correct state of the network is needed at the edge for providing a good route to a flow. However, this means that each link should be updating the edge about its available bandwidth constantly. As noted in [16], the overhead in updating link state constantly (with each observed change) is significant and hence the authors propose that the links propagate updates only if their states change "significantly" (e.g. the available bandwidth halves or doubles). The distributed resource management of our architecture would serve as an ideal mechanism to provide the desired information to the edge routers at a minimal cost.

B. Statistical Admission Control

The discussion in the paper assumes that admission control is meant for core-stateless guaranteed service. This makes the zero-false-positive property of utmost importance. However, if we wish to support only statistical guarantees, our architecture needs to be modified slightly. In this case token passing can be completely decoupled from the admission control portion of the architecture. Thus, the token would only serve to periodically update the network state at an edge router. The admission decisions would be based on estimation of link state based on the link state history reported by the token. For example, a simple exponential averaging or Gaussian predictor algorithm can give an estimate of the link state at any given instant and the admission decision would be based on the this estimate. The token periodically corrects any deviations in the estimate. Note that the response time to requests in this case is minimal, however there is an inherent trade-off between network utilization and the number of false positives. We plan to investigate this aspect of the token passing architecture in future.

C. Using Traffic Prediction

The link marking mechanism just uses the lower of the last two reported values (divided by the total traffic as seen from the token) as the indicator of fair share. It does not mandate a particular way of *choosing* the values to report. This leads to an interesting question of determining which functions provide the best view of the fair share under different conditions. We chose to use request load (based on exponential averaging) to serve as the indicator of fair share. On the other hand, complicated traffic prediction algorithms may be used to estimate an expected value of traffic. The better the algorithm, the fairer the division of link bandwidth and the higher the probability to remain in the unmarked phase.

D. Limitations

Relying on edge routers to perform complete path based admission control has some disadvantages as well. Internet is fundamentally a hierarchical network with topology information available only in routing domains. At a higher level of hierarchy, a domain is just a virtual node to which connectivity is provided using BGP. Thus, using complete topology knowledge limits our solution to edge-to-edge admission control.

For large networks, the number of links could be high enough not to fit into a single token. In such cases, there could be more than one tokens such that each link in the domain is part of exactly one token. Moreover, different tokens could circulate among different, possibly overlapping, subsets of edge routers. For example, if a link is used only by a subset of routers, the token carrying its information need only circulate among the corresponding subset.

In general, devising any mechanism which uses the entire Internet topology is neither feasible nor practical. We believe that having per-domain QoS framework is much more likely in practice rather than having a single framework across the Internet. In this regard, our solution fits perfectly within the purview of domain-specific protocols.

VI. Related Work

Admission control has been a well-researched topic. The main focus of this discussion is on the approaches explicitly meant to support core-stateless guarantees. We also give a brief overview of why modified versions of some of the other solutions are unsuitable for the purpose.

In [5] admission control is performed hop-by-hop. Edge routers maintain per-flow state and core routers estimate an upper bound on the available link capacity on their adjacent links and perform local admission test. While not maintaining per-flow states makes this method scalable, the network can remain underutilized because of over-estimation of aggregate reservation on a link. This algorithm puts an additional burden on core routers by requiring them to update the aggregate reservation state at each packet arrival by performing computations on the state carried in packet header. Lastly, the correct functioning of the algorithm requires that maximum inter-packet departure time for a flow's packet be small(for a reasonably small estimation window size). Thus, authors propose sending dummy packets from edge, in case no packets from a flow arrive at the ingress within the specified duration.

With these limitations in mind, a case for decoupling control functionality from core routers in the core-stateless framework was proposed in [11]. A bandwidth broker based architecture was proposed where the ingress edge router redirects a reservation request to a centralized broker node which has the complete network state database. The broker performs admissibility test and informs the ingress router of the admission decision along with parameters (like rate, delay and path allocated) for the flow. Since, the broker is a centralized node this architecture is not scalable in number of requests. This flaw is highlighted when we consider the case where the rate of arrival of requests is high but the bandwidth requested by each flow is small (for example, lot of requests for VoIP calls on a gigabit link). Moreover, the robustness of the architecture is questionable as it is a single point of failure. In fact, even if the broker node is assumed to be fail-safe, link failures, which occur quite frequently, could make the broker unavailable to edge routers for long durations.

The scalability limitations of the centralized broker architecture were addressed in a recent work [17]. A two-tier hierarchy of brokers was proposed where at the higher level there is a central broker node which is only supposed to maintain link state and allocates path-based quotas to edge brokers. The edge brokers can allocate resources on paths to individual flows as long as their quotas are sufficient. If an edge broker's quota on a path is used up, it requests additional quota from the central node. Our architecture retains the scalability of this approach while being more robust because it does not rely on a central node to control the allocation. The single point-of-failure problem exists in this approach because if the central node dies, the edge brokers would not be able to allocate bandwidth once their quotas expire (despite bandwidth being available). Lastly, this approach requires the communication between central broker and the edge brokers to be reliable in order to maintain consistency and the central broker should be able to allocate and revoke quotas from edge brokers without failure. In congested networks, the requests and state updations might get delayed inside the network resulting in an increased response time. We can leverage the guaranteed service framework to reserve small amount of bandwidth for token to ensure a fast transmission.

Another body of work exists which is core-stateless in nature but is only aimed at *statistical guarantees*. The distributed bandwidth broker architecture of [18] arranges brokers into hierarchy. It is easy to see that the zero-false-positive property could be violated here when different broker entities allocate bandwidth on the same link at the same time resulting in over-allocation. Measurement based admission control [19–21] is another core-stateless approach for admission control which is aimed at controlled load service in Intserv and in general at providing statistical guarantees [22]. Admission control is performed at edges (or endpoints) by taking decisions based on service received by probes. This approach is fundamentally unsuitable for guaranteed services because it relies on QoS received by probes which may be incorrect because of transient events like a burst of cross-traffic. Moreover, in a comprehensive comparison of measurement-based admission control schemes in [21], it has been shown that none of the schemes were able to meet a target packet loss rate.

VII. Conclusion

Core-stateless architecture retains the guaranteed service semantics of Intserv while having the scalability of diffserv. We addressed the problem of providing admission control support to a core-stateless network in a scalable and robust fashion while maintaining the zero-false-positive property. In our architecture, the edge routers perform admissibility test for requests on an edge-to-edge path basis. Our method uses a token passing scheme for resource management with the token doubling up as a semaphore for admission control on congested links. The token contents are used to dynamically partition a link's bandwidth among contending edge routers in proportion to their traffic for the link. The routers perform admission control in parallel until one of them senses imminent over-allocation on a link. At that point the token starts serving as the regulator of allocation on that link.

Our simulation results are very promising indicating that even under very high request load it is possible to perform resource allocation in parallel without suffering in terms of response time or utilization. Under different simulation conditions the ability to fairly and dynamically partition link bandwidth allows edge routers to serve requests independently till very high utilization levels. These results need to be verified in a real setting, possibly in a diffserv domain. Another area worth investigating would be to formally establish the interdependence of various parameters and their impact on core-stateless admission control. Our work is a first step in what we believe should be a concerted effort in studying the fundamental and practical aspects of distributed admission control for guaranteed services in core-stateless networks.

Acknowledgment

We would like to thank Scott Shenker for his valuable comments. Thanks are also due to the anonymous reviewers whose comments improved the quality of this paper.

References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reSerVation Protocol (RSVP) version 1, Functional Specification," IETF, RFC 2205, September 1997.

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

[3] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service," IETF, RFC 2212, September 1997.

[4] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding phb," IETF, RFC 2598, June 1999.

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

[6] S. Bhatnagar and B. Vickers, "Providing Quality of Service Guarantees Using Only Edge Routers," in *Proc. of IEEE Globecom*, November 2001.

[7] V. Padmanabhan and K. Sripanidkulchai, "The Case for Co-operative Networking," in *Proc. of First IPTPS*, March 2002.

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

[9] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF, RFC 3031, January 2001.

[10] J. Kaur and H. Vin, "Core-Stateless Guaranteed Rate Scheduling Algorithms," in *Proc. of IEEE Infocom*, April 2001.

[11] Z. Zhang, Z. Duan, L. Gao, and Y. Hou, "Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services," in *Proc. of ACM SIGCOMM*, October 2000, pp. 71–83.

[12] I. T. Union, *Network Grade of Service Parameters and Target Values for Circuit-Switched Services in Evolving ISDN*. Recommendation E.721, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, 1999.

[13] T. Eyers and H. Schulzrinne, "Predicting internet telephony call setup delay," in *Proc. 1st IP-Telephony Wksp., Berlin, Germany, Apr. 2000.*, 2000.

[14] D. Anderson, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. of ACM SOSP*, October 2001.

[15] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, New Jersey, 1992.

[16] G. Apostolopoulos, R. Guérin, and S. Kamat, "Implementation and Performance Measurements of QoS Routing Extensions to OSPF," in *Proc. of IEEE Infocom*, March 1999.

[17] Z. Zhang, Z. Duan, and Y. Hou, "On Scalable Design of Bandwidth Broker," *IEICE Transactions on Communications*, vol. E84-B, no. 8, pp. 2011–2025, August 2001.

[18] C. Chuah, L. Subramanian, R. Katz, and A. Joseph, "QoS Provisioning Using a Clearing House Architecture," in *Proc. of International Workshop on Quality of Service*, June 2000.

[19] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A Measurement-based Admission Control Algorithm for Integrated SErvices Packet Networks," in *Proc. of ACM SIGCOMM*, September 1995, pp. 2–13.

[20] D. Tse and M. Grossglauser, "Measurement-based Call Admission Control: Analysis and Simulation," in *Proc. of IEEE Infocom*, April 1997.

[21] L. Breslau, S. Jamin, and S. Shenker, "Comments on the Performance of Measurement-Based Admission Control Algorithms," in *Proc. of IEEE Infocom*, April 2000.

[22] E. Knightly and N. Shroff, *Admission Control for Statistical QoS: Theory and Practice*. IEEE Network, vol. 13, no. 2, pp. 20–29, 1999.