# Resilient Reduced-State Resource Reservation

András Császár, Attila Takács, Róbert Szabó, and Tamás Henk

*Abstract:* **Due to the strict requirements of emerging applications, per-flow admission control is gaining increasing importance. One way to implement per-flow admission control is using an on-path resource reservation protocol, where the admission decision is made hop-by-hop after a new flow request arrives at the network boundary. The next-steps in signaling (NSIS) working group of the Internet engineering task force (IETF) is standardising such an on-path signaling protocol. One of the reservation methods considered by NSIS is reduced-state mode, which, suiting the differentiated service (DiffServ) concept, only allows per-class states in interior nodes of a domain. Although there are clear benefits of not dealing with per-flow states in interior nodes—like scalability and low complexity—, without per-flow states the handling of re-routed flows, e.g., after a failure, is a demanding and highly non-trivial task. To be applied in carrier-grade networks, the protocol needs to be resilient in this situation.**

**In this article, we will explain the consequences of a route failover to resource reservation protocols: Severe congestion and incorrect admission decisions due to outdated reservation states. We will set requirements that handling solutions need to fulfill, and we propose extensions to reduced-state protocols accordingly. We show with a set of simulated scenarios that with the given solutions reduced-state protocols can handle re-routed flows practically as fast and robust as stateful protocols.**

*Index Terms:* **Admission control, congestion, failover, reduced-state resource reservation protocol, re-routing.**

## I. INTRODUCTION

In recent years, the volume of streaming media transported over Internet protocol (IP) networks has increased noticeably [1]. This is not only due to emerging applications like Real Media, Windows Media, or QuickTime but also a consequence of old services appearing over IP networks, like telephony (e.g., via Skype). In a carrier-grade network, such applications will require stringent bandwidth guarantees to fulfill their throughput and delay requirements. In order to ensure bandwidth for *flows*,[1] one must use per-flow admission control. One way to implement such a method is to use an on-path resource reservation protocol. These protocols send a signaling packet across the future path of the new flow, which contains the descriptors of the flow (e.g., peak rate, token bucket, etc.). On every hop, each router makes a local decision whether it can accommodate the new request. The new flow will be admitted into the network only if all intermediate routers admit it.

Manuscript received March 10, 2004; approved for publication by Dan Keun Sung, Division III Editor, September 27, 2005.

A. Császár and A. Takács are with the TrafficLab, Ericsson Research, Budapest, Hungary, email: {Andras.Csaszar, Attila.Takacs}@ericsson.com.

R. Szabó and T. Henk are with the High Speed Networks Laboratory at the Department of Telecommunication and Media-Informatics at Budapest University of Technology and Economics, Budapest, Hungary, email: {Robert.Szabo, henk}@tmit.bme.hu.

In the Internet engineering task force (IETF), the next-steps in signaling (NSIS) working group [2] is responsible for standardising a next-generation IP signaling protocol for flow-level resource management as the first use case. The intention of the developed NSIS signaling layer protocol (NSLP) is to support different quality of service (QoS) models. Currently, the working group considers implementing two models, a *stateful* and a *reduced-state* variant [3]. These terms reflect the complexity of interior nodes in a domain. The stateful solution stores per-flow state information in interior nodes suiting the integrated service/resource reservation protocol (IntServ/RSVP) QoS model. On the other hand, the reduced-state solution relies only on aggregated, per-class states in interior nodes, and only edge nodes are permitted to dispose of per-flow states. This mode implements the DiffServ/RMD model, where RMD stands for resource management in DiffServ [4], [5].

In previous papers [6], [7], we argued that under normal circumstances in a high-speed network with many flows the reduced-state mode is preferable over stateful operation. The reason is that a reduced-state protocol poses less processing and storage capacity requirements to core routers and it has smaller protocol overhead, while the achievable performance is similar.

Before anyone can expect that the RMD-NSLP proposal (denoted shortly as RMD in the following) is implemented by router vendors, the protocol has to be resilient after node or link failures. After such outages, the routing protocol re-directs flows from their original paths to alternative paths. In general, the network faces the problem that there is not enough bandwidth to accommodate all re-routed flows. That is, congestion may occur. In fact, it has been shown by Iyer *et al.* [8] that most occurrences of link overload are caused by re-routing after link failures. Moreover, Markopoulou *et al.* [9] showed that link failures are part of everyday operation in a tier-1 network. The authors found that on the average link failures occur in every 30 minutes in the Sprint backbone. Failures are due to a number of causes, like optical fibre cut and other environmental effects, router hardware/software failures, or operator errors.

The above observations have been taken on a best-effort network. However, in a network where per-flow admission control is applied the consequences of re-routing are even less favourable. When the reservation protocol admitted flows, it guaranteed the requested bandwidth for all of them. The protocol, after re-routing, faces the problem that the admission of re-routed flows to the new path is uncertain, even severe congestion may occur. This means that the flows that cannot be maintained have to be terminated quickly to provide the required bandwidth for the remaining flows. Moreover, since the routing protocol re-directed the flows independently of the reservation protocol, the stored reservation states on the new path will be incorrect,

[1] A *flow* is defined as a series of packets transmitted during a session of a specific application by a specific host (e.g., a streaming audio or video session, or an FTP file transfer).

so the admission decisions for newly arriving requests will also be incorrect. We will show that stateful solutions are resilient by having a natural and quick solution to recover from such situations. Without per-flow states, a reduced-state protocol needs additional mechanisms for quick recovery.

The new contributions in this article are the following. First, we describe and formalise the two problems resulting from re-routing—severe congestion and incorrect admission control. Secondly, we set requirements that a solution has to fulfill, like fast severe congestion handling, policy control, and correction of incorrect admission decisions. Thirdly, we present protocol extensions for reduced-state protocols—specifically for RMD—that meet these requirements. Finally, with the help of simulations we prove that by applying these extensions, RMD can be made practically as resilient as its stateful competitors, specifically RSVP. The scalability, overhead, and reaction time properties of reservation protocols that do not need per-flow states have been investigated by many researchers. Moreover, the authors of previous signaling protocols also noted the problem of route changes (as shown in Section II). However, to the best of our knowledge, we are the first ones who deal with the re-routing aspects in detail, and give a satisfactory solution.

The structure of the article is the following. In Section II, we present the evolution of on-path reservation protocols. We outline the protocol and admission control mechanisms of RMD that are relevant for understanding the problems and algorithms in Section III. Section IV describes the problems caused by re-routing in detail: Severe congestion and incorrect admission decisions due to out-of-date reservation states. Section V explains the requirements we have set against a solution, it outlines the quick reaction of stateful protocols, and it looks at the problem from the aspect of reduced-state resource reservation protocols. We propose solutions for severe congestion handling in Section VI and for overriding incorrect admission control after re-routing in Section VII. In Section VIII, we evaluate our proposals with the help of extensive simulation experiments. In the last section, we conclude and summarize the article, and present possible future research directions.

## II. BACKGROUND

The current bandwidth provisioning routine of IP network operators is overprovisioning [1], [10], [11], i.e., the continuous cycle of monitoring the utilization of the network and re-dimensioning the capacities for an over-estimated traffic demand. Overprovisioning does not require new functionality from routers inside the network, therefore it is practical. Some properties of overprovisioning are favourable like negligible queuing delay and congestion. However, overprovisioned networks suffer from low utilization. Another drawback is the fact that it cannot provide bandwidth guarantees under unexpected traffic loads.

Per-flow admission control is capable of achieving higher network utilization with bandwidth guarantees but such methods are not widely used in operational networks, since these require new functionality from networking equipment. The task of a per-flow resource management scheme is to implement admission control for individual flows. The considered schemes are either centralized or distributed.

Bandwidth broker (BB) [12], [13] is a centralized per-flow resource reservation concept, which requires one new functional entity inside the network. A new flow request turns to the BB node to gain admission into the network. The BB administers the reservation states of the links and paths, so it can make an admission decision. As a centralized scheme, the BB concept raises questions about being a single point of failure; how to gather online network state information; and how to cooperate with arbitrary routing.

On-path resource reservation protocols send a signaling packet across the allocated path of the new flow, which contains the descriptors of the flow. On every hop, each router makes a local decision whether it can accommodate the new request. The new flow will be admitted into the network only if all intermediate routers admit it. These protocols are distributed in nature but require cooperation from each router on the path. Yet again, this also means that there is no single point of failure, and there is no need to gather network state information, since each router makes a local decision.

Resource reservation protocols can be categorized as soft or hard state, and as sender or receiver initiated. Soft-state protocols require that reservations are refreshed periodically, otherwise they are cleared. Hard state protocols maintain reservations until explicitly released, hence they are less robust as failing to release results in over reservations. For a detailed comparison hard-state and soft-state protocols, the reader is referred to the work of Ji *et al.* [14]. Sender initiated means that the reservation is started by the sender, while receiver initiation means that the party who wants to receive data has to start the reservation to join a data stream. From the ideology of receiver initiation it follows that this suits well the concept of multicasting (i.e., when joining a multicast stream). With a receiver initiated protocol, the reservation signaling message goes upstream: From the receiver towards the sender. However, the reservation is required downstream. Since asymmetric routing does not guarantee that the two paths coincide, the protocol itself has to ensure that the upstream signaling messages are routed on the downstream path. Therefore, receiver initiation requires the storage of per-flow forwarding information in intermediate routers.

A few years ago multicasting was believed to be going to be a widely used approach to reduce the load of networks. Therefore, many early reservation protocols were multicast oriented. Internet streaming protocol (STv2) [15], for example, was a sender initiated, hard-state protocol but since it was sender initiated it did not scale well with the number of receivers in a multicast group, and its reservation states and processing requirements were enormous. RSVP [16] was then designed to be receiver initiated for the sake of lower overhead in multicast, and it used soft states to be more reliable when tearing down reservations. RSVP was made an IntServ-standard but it was not scalable due to per-flow state requirements. Moreover, RSVP was too complex for unicast applications, especially with the evolution of DiffServ, which does not allow per-flow states in interior routers. YESSIR [17] was designed to reduce the complexity and improve the scalability of RSVP. It was still considered for the IntServ model with per-flow states. However, multicasting, although supported, was not any longer the target objective, so

YESSIR was again made sender initiated.

Completely ignoring multicasting, many lightweight, soft-state, and sender-initiated resource management protocols have been proposed, practically in parallel: Ticket signaling protocol (TSP) [18], boomerang [19], load control [20], and sender oriented signaling (SOS) [21]. These are reduced-state proposals, i.e., they only need per-class states in core nodes, therefore they are applicable in DiffServ networks. From load control evolved RMD [6], [7], [22]. The RMD specifications also appeared as Internet drafts [23], [24].

A more detailed survey of resource management and QoS signaling protocols can be found in [25] and [26]. From all these solutions, currently only RSVP is used in a wider range. Although RSVP was originally intended to support the flow-level reservation of real-time multicast applications, currently its popularity is due to the many extensions it received. The extensions include additional features, like security and scalability, or describe new applications but above all it is used in MPLS networks for traffic engineering purposes [27]. However, the original goal to support multicast streaming faded, since over the past years, the demand for multicast has never materialized.

To remedy the scalability and complexity problems of RSVP, in 2001, the IETF started developing and standardizing the new NSIS protocol family. The work is still ongoing, the protocol specifications are expected to become standard RFCs in late 2005. Although the NSIS signaling layer protocol is not going to support multicast [28], it is made to be general to support bidirectional reservations, mobility, security, sender or receiver initiated operation, reliable signaling message transportation, and different QoS models. One such model, the adaptation of the RMD framework to NSIS [4], [5], is the subject of our investigations.

RMD was the first reduced-state resource reservation protocol paying specific attention to the fast and fair handling of the consequences of re-routing. The sender oriented signaling protocol [21], for example, assumes that there is no re-routing in the network and the authors admittedly ignore this problem. Boomerang assumes that re-routing is rare [19], therefore it is satisfied by the soft-state refresh solution that we will also cover in Section V-B. The authors of TSP proposed that the re-routing node drops re-routed signaling messages and remarks all re-routed packets to a lower priority class [18]. We will show that the soft-state refresh solution is slow, while the dropping/remarking solution does not consider the possibility that re-routing by itself is not necessarily a problem. However, before turning our attention to the consequences of re-routing, we first outline the relevant mechanisms of RMD.

## III. RMD OPERATION

In RMD, when a new flow request arrives at the ingress edge of a network domain, the ingress will send a *reserve* signaling message towards the destination. The message contains the resource needs of the flow. While the signaling packet is routed across the network, every node makes a local admission decision. If the new flow is rejected at a node, a bit will be set in the message. Eventually, the packet reaches the egress edge, which learns the per-domain admission decision from the status of that bit, and signals back the decision to the ingress edge with a *response* message. End-to-end reservations are concatenations of this per-domain procedure. In the rest of this paper, we restrict our attention to the reservation mechanism inside a single domain, and to the case, when the resource need of a flow is specified with a single parameter, the *bandwidth rate* (denoted as $p$).

RMD, conforming the DiffServ architecture, solves per-flow admission control in interior routers without actually storing per-flow states. Every outgoing link in a router is pre-configured with a per-class *admission threshold* ($T$), which reflects the maximum traffic volume that can be admitted in that traffic class. Every node records the aggregated sum of the signaled bandwidth rates of previously admitted flows. The last value of this counter is *last*, which is set to zero at startup. When a new flow request is received at the node, it checks whether the following *admission condition* is fulfilled

$$last + p \leq T \ . \tag{1}$$

If true, the packet is forwarded unmodified towards the destination, otherwise the $M$ bit of the packet is set before forwarding it.

### A. Implementing Soft States: The Sliding Window Approach

If a flow is admitted, the ingress edge is obliged to refresh the reservation of the flow with a *refresh* signaling message in periodical *refresh intervals* (denoted by $R$), which is a domain-wide constant (30 seconds by default). If a refresh message is not received, the aggregated reservation state is decreased. This is done by a sliding window mechanism first described by Marquetant *et al.* [29]

The approach is depicted in Fig. 1. In each router, the $R$ long time windows are divided to $N$ cells. By default, $N$ is 10, so the window is divided to 3 second cells. Each cell is represented by a counter $C_i$, $i = 1$ to $N$, which record the reserved and refreshed bandwidth in the corresponding cell.

During operation, the refreshed and reserved bandwidth is summarized in variable *count*, while the reserved bandwidth is added to *last* as well. That is, if a reservation message with a request of $p$ bytes per second is admitted, the variables are updated as follows

$$\begin{aligned} last &\leftarrow last + p \\ count &\leftarrow count + p. \end{aligned} \tag{2}$$

For refresh messages

$$count \leftarrow count + p. \tag{3}$$

After each $R/N$ seconds, i.e., after each cell, the states are updated by sliding the reservation window with one cell

$$\begin{aligned} &\text{for } i = N, \cdots, 2 : C_i \leftarrow C_{i-1} \\ &C_1 \leftarrow count \\ &last \leftarrow \sum_{i=1}^{N} C_i \\ &count \leftarrow 0. \end{aligned} \tag{4}$$

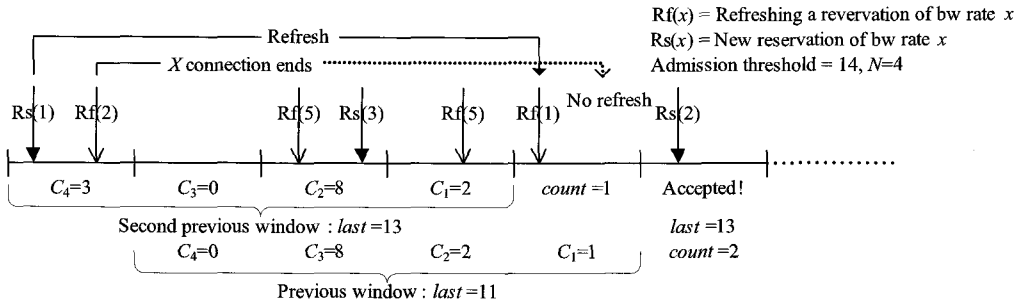


Fig. 1. Sliding refresh window: An example.

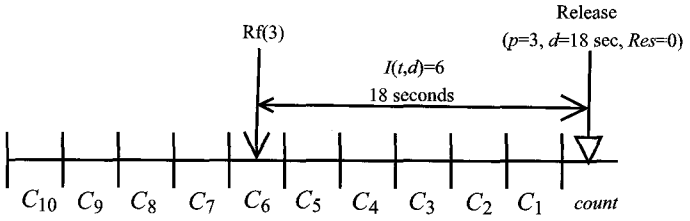


Fig. 2. Release a reservation in a cell of the sliding window.

With this procedure, a non-refreshed reservation times out after maximum $R + R/N$ seconds. For example, in Fig. 1, the second flow in the first cell terminates at the indicated time instance. Since its reservation is not refreshed in the fifth cell, *last* will be decreased at the end of the fifth cell, and as a result a new flow can be admitted in the sixth cell.

Besides soft-state timeout, RMD also handles explicit release signals. The *release* message contains $p$, the bandwidth rate that can be released; $d$, which identifies the time difference between the departure time of the release message and the departure time of the last reservation or refresh message; and *Res*, a bit, which is set to 1 if the message releases a first reservation (if a flow lasts less than a refresh interval) and set to 0 if it releases a refreshed. In each router, function $I(t, d)$ calculates the cell that has to be decreased, where $t$ is the local time. The way $I(t, d)$ is calculated is demonstrated in Fig. 2: When the release message arrived, it contained $d = 18$ sec. Cells are 3 seconds long, therefore in this case we count back 6 cells, i.e., $C_6$ needs to be decreased. The selected cell also depends on when the release message arrives in the cell, but the exact formula is left to the reader.

When releasing, the variables are updated as follows

$$\text{if } I(t,d) \begin{cases} > 0: & C_{I(t,d)} \leftarrow C_{I(t,d)} - p \\ = 0: & count \leftarrow count - p \\ & \text{if } (Res = 1): last \leftarrow last - p. \end{cases} \quad (5)$$

## IV. PROBLEM DESCRIPTION

### A. Severe Congestion

From the perspective of the resource reservation protocol, the routing protocol acts independently as part of the external environment when re-routing flows. Neither OSPF nor IS-IS has quality-of-service considerations when calculating new routes after a topology change. That is, flows may get re-routed to paths that do not have enough free capacity to accommodate all (e.g., because the re-routed traffic increases the existing load above the available link capacity). As a result, *severe congestion* may occur on certain links that may persist without intervention. Severe congestion degrades the performance of all flows passing that link, i.e., both the re-routed flows as well as those ones that were originally traversing that link. However, the resource reservation protocol guaranteed congestion free transmission through the network when it admitted the flows. After re-routing, this assurance may be violated. Hence, a prompt response is required.

Let us formalize the problem. Let $V_{\text{before}}$ denote the traffic volume of the examined link before it receives re-routed traffic. Due to admission control, it is true that

$$V_{\text{before}} \leq T.$$

Let $V_{\text{after}}$ denote the total traffic volume after re-routing, and $V_{\text{re}}$ the re-routed traffic volume

$$V_{\text{after}} = V_{\text{before}} + V_{\text{re}}\ .$$

Severe congestion occurs, if

$$V_{\text{after}} > T\ . \quad (6)$$

Note that the volume of *overload* ($V_{\text{overload}}$) after re-routing that should be freed up is

$$V_{\text{overload}} = V_{\text{after}} - T.$$

### B. Incorrect Admission Control

The second problem is that a re-routing event not just degrades the performance of previous flows, but new flows that arrive after re-routing took place, may develop congestion in spite of using admission control. The reason is that the reservation states along the path do not immediately reflect the presence of the re-routed flows, so new flows may be admitted incorrectly. Ultimately, the admission control may fail to achieve its primary goal to prevent congestion.

To formalize the problem, let us introduce $V_{\text{newreq}}$, denoting a sudden burst of new traffic requesting admission at the examined link after re-routing. Let $last_{\text{before}}$ be the reserved traffic volume before re-routing, and $last_{\text{after}}$ afterwards. Before

re-routing, $last_{\text{before}} = V_{\text{before}}$, however since routing is independent from the reservation protocol, $last_{\text{after}} = last_{\text{before}} = V_{\text{before}}$. Therefore, after re-routing, the router is willing to admit $T - last_{\text{after}} = T - V_{\text{before}}$ amount of new traffic. However, the free capacity after re-routing is only $T - V_{\text{after}}$. Admission control is incorrect if $\min\{T - V_{\text{before}}; V_{\text{newreq}}\} > T - V_{\text{after}}$.

The consequences are twofold. First, if re-routing immediately caused overload, the router may still admit new flows further increasing congestion. Second, if re-routing by itself did not cause overload, i.e., if $V_{\text{after}} \leq T$, due to newly arrived and admitted flows, the link could still transit to the state of severe congestion.

## V. SOLUTIONS

### A. Requirements

**Req. 1.** Severe congestion has to be resolved by terminating enough flows to free up $V_{\text{overload}}$. The word *terminate* means any of the following:

- Do not let any more packets of the flow into the network, e.g., by filtering its packets. This approach practically ends the connection of the corresponding flow.
- Re-map the packets of the flow to a lower priority class, e.g., to the best-effort class. The connection is not ended, although the previously agreed bandwidth is not guaranteed any longer in the original traffic class.
- Signal the application to switch to a lower bit-rate codec.

The flows are terminated in order to ensure the required bandwidth rate for the remaining flows. If the protocol did not bring this sacrifice, all flows would experience congestion. As long as congestion persists, the perceived end-user quality is degraded. In worst case, the quality may not be tolerable. For example, for voice services, the American National Standards Institute reports [30] that recovery times below 200 ms do not likely have impact on the service, and restoration times between 200 ms and 2 s have only a minimal impact on voice applications. Iannaccone *et al.* showed [31] that with current router hardware and tuned timer configuration the IS-IS routing protocol is capable of achieving sub-second routing convergence after a topology change. Therefore, to stay within the 2 s boundary for recovery time, a sub-second severe congestion handling time is desirable.

**Req. 2.** A network operator should be able to temporarily provide a bigger capacity for the re-routed flows of certain traffic classes. That is, (6) changes to

$$V_{\text{after}} > T_{\text{RR}} \tag{7}$$

where $T_{\text{RR}} > T$. This way, the amount of overload decreases to $V_{\text{overload}} = V_{\text{after}} - T_{\text{RR}}$. For example, the operator may want to achieve that its voice traffic class, which is normally allocated 40% of the link capacity, is able to use up to 60% after a possible re-routing. In this case, if $c$ denotes the link capacity, the operator would set $T = 0.4c$ and $T_{\text{RR}} = 0.6c$.

**Req. 3.** Operator policies should be able to determine which flows are terminated during severe congestion handling. Flows are terminated as long as $V_{\text{overload}}$ is positive, but as a last resort, the operator may want to save certain flows. For example, emergency voice calls may be preferred over other voice calls, or the operator may opt to select high- or low-bandwidth flows to terminate.

**Req. 4.** Incorrect admission control has to be resolved by repairing the bases of bad decision, the stored reservation variables. While the variables are not corrected, a more conservative admission control algorithm has to override the regular admission control so that excess flows are not admitted.

### B. The Built-In Mechanism: Soft-State Refreshes

All soft-state protocols have a built-in mechanism to resolve congestion and to repair the reservation variables. This is done with the help of the periodical refresh messages, which are sent maximum within a refresh interval. Refresh messages are similar to new reservation messages, they contain the bandwidth rate attribute that needs to be refreshed. On a link, where it had no previous reservation, a refresh message acts like a reservation message. This way, the randomly arriving refresh messages will update the reservation states, and the reserved bandwidth will begin to increase as more and more messages arrive. As soon as the reserved traffic volume calculated from the reservation states reaches the admission threshold $T$, two things will follow. First, new flows will be rejected, so admission control will operate correctly. Secondly, those flows that have not yet refreshed will also be rejected. Rejecting a refresh message is done in the same way as rejecting a new reservation request message but the consequence is different, since here an ongoing flow will be terminated. As a result, after a complete refresh interval, all excess flows are guaranteed to get terminated, so congestion is eliminated.

The drawback of this solution is that it is slow. That is, congestion is resolved only after a refresh interval. For example, it may be possible that a flow refreshed its reservation on the old path one second before getting re-routed, so it will not refresh again within $R - 1$ seconds after re-routing, and so it cannot be terminated for that long. Moreover, until the reservation states do not reach $T$, newly arriving flows may be admitted incorrectly (because the real traffic volume may be higher). Therefore, Requirements 1 and 4 are not fulfilled.

Requirement 2 is not fulfilled either, although the solution is simple. We propose to change the admission condition in case of refresh messages only by replacing $T$ with $T_{\text{RR}}$.

Requirement 3 is not fulfilled because those flows are terminated that refresh later. This is not influenced by the operator.

### C. The Stateful Solution: RSVP Local Repair

By utilizing per-flow information in every node, stateful protocols may offer a much faster solution. By receiving a trigger signal in the router that changes its forwarding tables, a stateful protocol can initiate the refresh procedure immediately. Per-flow states make such an action feasible in two ways. First, since the re-routing node has a per-flow database (which store, e.g., the destination address for all flows) and the node knows which destination addresses have been given a new next-hop, it can select which flows need to re-reserve their reservations. Secondly, each router receiving a refresh message knows whether that particular flow has installed a state previously. If not, it is

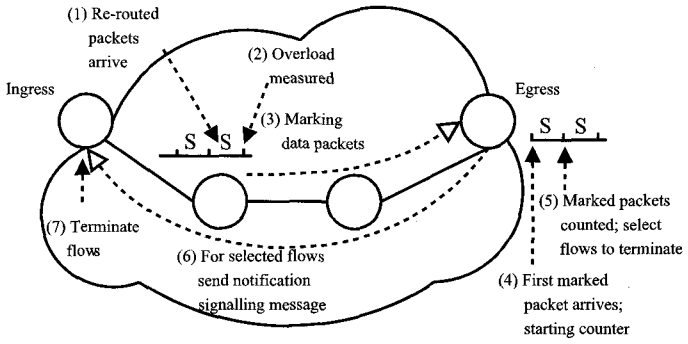


Fig. 3. Basic severe congestion handling mechanisms of RMD.

considered as a new reservation and admission control is performed.

This solution fulfills both Requirements 1 and 4, since the refresh procedure can begin instantly after re-routing, so excess flows are terminated and the states are repaired within the order of round-trip times. Although even during this short time new flow requests may be admitted incorrectly, their effect is negligible. In RSVP, this solution is called *local repair.*

Since local repair is essentially a speed-up version of the soft-state refreshes, the same is true for Requirements 2 and 3 as above.

### D. Without Per-Flow States

Since local repair builds on having per-flow states in interior nodes, it is not applicable with a reduced-state reservation protocol. First, there is no per-flow database in the routers, so the re-routing node cannot select any flow to re-reserve its resources. Secondly, without per-flow states the router cannot know whether a particular flow already has a reservation. If all flows were to re-reserve, this would lead to double reservations. For example, on Fig. 10(a), if the link between Miami and Dallas goes down, the traffic from Miami to LA that originally took the Miami, Dallas, LA path takes a detour on the Miami, Atlanta, Dallas, LA path, i.e., the link between Dallas and LA is not changed, the flows already have a reservation on the link.

Therefore, our objective was to find new solutions that are capable of fulfilling all requirements, thereby providing a similar performance like local repair. Our first priority was the handling of severe congestion, the proposed algorithm is explained in Section VI, which fulfills Requirements 1 to 3. Remedying incorrect admission decisions after re-routing (Requirement 4.) was our next objective, the proposed solutions are described in Section VII.

## VI. SEVERE CONGESTION HANDLING

To fulfill Requirement 1 with reduced-state protocols, severe congestion in the communication path has to be notified to the edge nodes, since core nodes do not have per-flow identification and so cannot initiate flow termination. Edge notification is done by the mechanism depicted on Fig. 3.

**Step 1.** Re-routed packets arrive at a core router.

**Step 2.** The core router detects and estimates the overload ($\hat{V}_{\text{overload}}$) on the link. Overload can be detected and measured in one of the following ways

- packet drop count, or
- bandwidth measurement of incoming traffic.

With packet drop count, the router counts the amount of dropped bytes, denoted by $D$, on its outgoing interface for every traffic class. In periodical $S$ intervals, it estimates the overload as $\hat{V}_{\text{overload}} = D/S$. The advantage of a packet drop based detection is that it is very simple. The disadvantage is configuration complexity since it requires the application of a rate-limiting scheduling discipline for every traffic class configured according to its respective admission threshold. For example, if the queue did not drop packets when the incoming traffic rate exceeds the admission threshold, overload would not be detected. Requirement 2 is fulfilled since the operator can rate-limit the class to $T_{\text{RR}}$ instead of $T$.

In case of bandwidth measurement, the amount of all arriving bytes, denoted by $A$, is counted before packets may get dropped. After an $S$ interval, the average incoming bandwidth during the interval is $A/S$. There is overload on the link, if this value is higher than the admission threshold. That is, overload is estimated as $\hat{V}_{\text{overload}} = A/S - T_{\text{RR}}$, i.e., Requirement 2 is fulfilled.

**Step 3.** If $\hat{V}_{\text{overload}} > 0$, the congested core router stamps transmitted user data packets in order $i$) to identify the packets traversing through a congested node, and $ii$) to inform the edge nodes about the amount of the estimated overload. A packet can have three stamping states: *none*, *encoded*, and *affected.*

With the help of the *encoded* stamp, the measured overload is encoded and signaled in an in-band fashion to the egress nodes. The value of $\hat{V}_{\text{overload}}$ is encoded in the packet sizes of the stamped packets as follows. At the end of each $S$ long measurement period ($S$ is a domain-wide constant) the congested core node transforms $\hat{V}_{\text{overload}}$ to a number of bytes to mark as $B = \hat{V}_{\text{overload}} \cdot S$. In the new measurement period, the core router will stamp leaving packets as *encoded*, so that the total size of *encoded* packets is $B$.

As long as severe congestion persits, all packets that have not been stamped as *encoded* will be stamped as *affected.* As a result, if a packets is received stamped as *affected* or *encoded*, the egress edge node knows that it has passed a congested node. Moreover, *encoded* packets also carry information about the value of overload. Stamping can be done by

- setting two reserved bit in the IP packet header (3 stamps need at least two bits to encode), or
- having assigned two other DSCPs for every traffic class and changing the DSCP of the packet, which is interpreted at the egress nodes as if the packet had been marked with one of the two stamps.

**Step 4.** After having received the first stamped packet, every egress node counts the number of received *encoded* bytes with counting interval $S$.

**Step 5.** When the counting interval is over, the egress node transforms the counter back to an overload bandwidth value as follows. Let $\mathcal{E}$ denote the set of egress routers receiving stamped packets. $B_e$ is the amount of bytes stamped as *encoded* as counted by egress router $e \in \mathcal{E}$. Then $B_e/S$ gives the amount of overload seen by egress router $e$. Note that if marked packets are not dropped then the summarized overload bandwidth seen by all affected egress routers is the same as the overload estimated

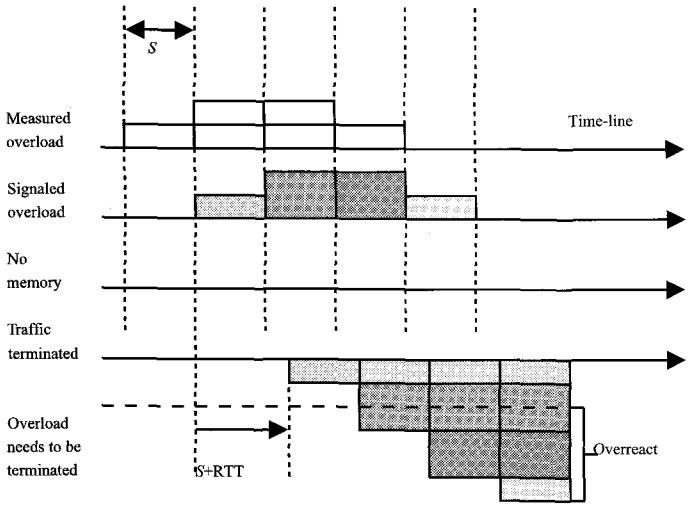


Fig. 4. Overreacted severe congestion handling without memory.

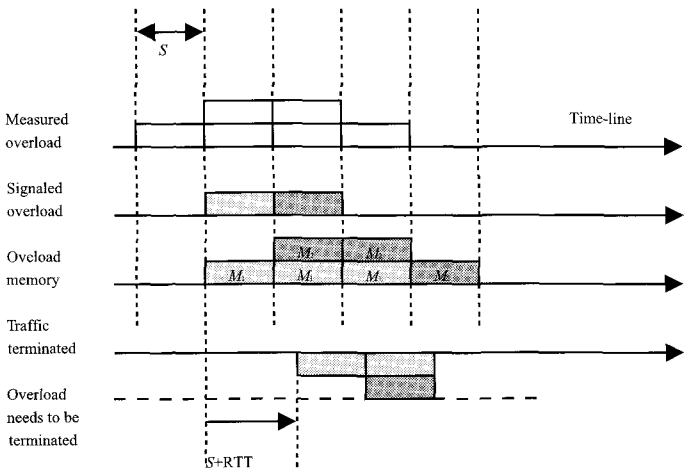


Fig. 5. Precise severe congestion handling with memory.

on the core node

$$\sum_{\forall e \in \mathcal{E}} B_e = B \Rightarrow \sum_{\forall e \in \mathcal{E}} \frac{B_e}{S} = \hat{V}_{\text{overload}}.$$

After calculating $B_e/S$, any egress router $e$ chooses a set of flows to terminate, which are together responsible for generating a $B_e/S$ bandwidth. The egress node can choose from those flows that received either *affected* or *encoded* stamped packets. Since the selection can happen according to any arbitrary algorithm, Requirement 3 is fulfilled.

**Step 6.** For every chosen flow, the egress sends back a notification signaling message to the ingress node of the flow, requesting it to terminate the connection.

**Step 7.** Finally, the flows that receive a notification message will be terminated by the ingress nodes.

Since stamping is done in core nodes, the decisions are made at egress nodes, and termination of flows are done in ingress nodes, there is a significant delay until the overload information is learned by ingress nodes. The delay consists of the trip time of data packets from the congested core node to the egress, the counting interval in the egress ($S$), and the trip time of the notification signaling messages from egress to ingress (see Fig. 3). Moreover, until the overload decreases at the congested core node, an additional trip-time from the ingress node to the core node must expire. This is because immediately before receiving the congestion notification, the ingress may have sent out packets in the flows that were selected for termination. That is, a terminated flow may contribute to the congestion for a time longer that is taken from the ingress to the core node. Considering all the delays involved, we get that signaling the congestion has no influence on the overload for as long as $S$+RTT, where RTT is the round-trip time.

Without additional functions, core nodes would continue marking the packets until the measured utilization falls below the congestion threshold. Thus, it can happen that the necessary number of flows are already being terminated but the core node still signals overload to the egress as shown in Fig. 4. This way, in the end more flows will be terminated than necessary. Such over-reaction due to the delayed feedback of the congestion handling control loop is a well-known phenomenon in control theory.

To solve the problem of over-reaction, in a previous paper [32] we proposed that core nodes use a memory to keep track of the signaled overload in a couple of previous measurement intervals. At the end of a measurement period before encoding and signaling the overload as stamped packets, the actual measured overload is decreased with the sum of already signaled overloads stored in the memory, since that overload is already being handled in the severe congestion handling control loop.

The memory is implemented with a sliding window consisting of an integer number of cells. At the end of every measurement interval, the newest calculated overload is pushed into the memory, and the oldest cell is dropped. If $M_i$ is the $i$ memory cell ($i \in [1, \cdots, n]$), then at the end of every measurement interval the signaled overload ($V^{\text{sig}}_{\text{overload}}$), as shown in Fig. 5, is calculated as the following

$$V^{\text{sig}}_{\text{overload}} = \hat{V}_{\text{overload}} - \sum_{i=1}^{n} M_i \,.$$

Next, the memory is updated as

$$\text{for } i = 1, \cdots, (n-1) \,:\, M_i \leftarrow M_{i+1} \tag{8}$$

$$M_n \leftarrow V^{\text{sig}}_{\text{overload}}. \tag{9}$$

With the help of worst case calculations, in [32], we gave guidelines for configuring $n$ and $S$ to achieve fast handling time without over-reaction. According to these, over-reaction can be eliminated if

$$n = 1 + \left\lceil \frac{\text{RTT}_{\max}}{S} \right\rceil$$

where $\text{RTT}_{\max}$ is the maximal RTT value in the domain. As a rule of thumb, we have shown that a quick handling time can be achieved if $S \leq \text{RTT}_{\max}$, which also results in sub-second handling time for all practical round-trip times (i.e., Requirement 1 is fulfilled).

Although this severe congestion handling algorithm fulfills Requirements 1 to 3, we are only half way to a complete solution, since incorrect admission decisions are still to be dealt with.

## VII. CORRECTING CAC AFTER RE-ROUTING

In Section IV-B, we explained that due to out-of-date reservation states, admission control may incorrectly admit new flows. This may further increase an already existent congestion, or it may form a new congestion. Section V-B described how regular refreshes repair the states within a refresh period. Although this is very slow, in a reduced-state environment it is not possible to repair the reservation variables faster than a refresh interval. Therefore, in fulfillment of Requirement 4, in a previous paper [33] we proposed solutions to override the incorrect admission control during this time with more conservative algorithms. This means that during a refresh interval after the presence of re-routed traffic is detected on a link, every new flow request, if admitted by regular CAC, also has to be admitted by another admission control algorithm, which may override the other's decision.

### A. Pure MBAC

Since the heart of the problem is that the reservation variables cannot be updated quickly, our first proposed method is to temporarily use a *pure* measurement-based admission control (MBAC) algorithm. Since such methods do not rely on any stored variables, they are immune against their imprecision. For a survey of some measurement-based admission control algorithms the reader is referred to [34]. Note that not all measurement-based algorithms are suitable for our specific purpose. Those, which are not based purely on measurements (e.g., which need the number of flows, or different descriptors) are not feasible. In the rest of this article, we use the *measured sum* algorithm. Using the override, the original admission condition shown in procedure (1) changes to

$$(last + p \leq T) \quad \text{AND} \quad (meas + p \leq T)$$

where *meas* is the measured arrival rate of existing traffic in the given traffic class. The first factor is the normal CAC, the second factor is the CAC formula of the measured sum algorithm.

Whatever MBAC algorithm is applied, it must be able to quickly measure a sudden traffic increase (e.g., within 1–2 seconds), but it must not be too sensitive to block new flows without reason (e.g., due to normal burstiness of the traffic).

### B. Detection of Re-Routed Flows from Refresh Messages

The implementation of the pure MBAC approach is very simple if bandwidth measurements are already included in routers. However, if the operator does not want to or cannot rely on measurements (e.g., because the measurement implemented by the router vendor is too slow for this purpose), the operator will need another solution relying on RMD protocol messages.

The key observation with respect to detecting re-routed flows is the following (Fig. 1 provides assistance). Under normal circumstances, supposing that explicit release is used, at the end of a cell the refreshed part of *count* is equal to $C_N$, the counter of the oldest cell in the refresh window. To show this, let us split *count* into *refcount* and *rescount* counting only refreshes and new reservations respectively. The update procedures (2) and (3) are modified as follows. For admitted reservations

$$\begin{aligned} last &\leftarrow last + p \\ rescount &\leftarrow rescount + p. \end{aligned} \tag{10}$$

For refresh messages

$$refcount \leftarrow refcount + p. \tag{11}$$

Instead of (4), the cell shifting procedure also changes

$$\begin{aligned} &\text{for } i = N, \cdots, 2 : C_i \leftarrow C_{i-1} \\ &C_1 \leftarrow refcount + rescount \\ &last \leftarrow \sum_{i=1}^{N} C_i \\ &refcount \leftarrow 0 \\ &rescount \leftarrow 0. \end{aligned} \tag{12}$$

The above mentioned key observation now can be formalized. Just before shifting the cell, normally it is true that

$$C_N \geq refcount.$$

This is true because if a flow is included in *refcount*, then it has sent its last refresh or reserve message a window earlier, which is now in $C_N$. Inversely, if a flow is not included in *refcount* but it was present a window earlier, then it must have left the network in the meantime.

If explicit release is used, the release procedure decreased the necessary cell, i.e., now it is not included in $C_N$ either, so the two variables are equal

$$C_N = refcount.$$

Now let us see, how it may happen that $C_N < refcount$. First, some refresh messages that were due in the previous cell may have arrived a little late, or some that are due only in the next cell may have arrived a little earlier. This could be caused by network jitter. The more important possibility is that there were flows which did not reserve bandwidth a window earlier (now in $C_N$) but which have refreshed now. And these are the re-routed flows that we are looking for. To detect re-routing in this way, all we need to do is to run a condition check at each cell shifting before executing the cell shifting procedure (12)

$$\text{if } (C_N + L < refcount) \Rightarrow \text{re-route occurred}$$

where the constant $L$ is in the formula to exclude the effects of late or early refresh packets. The advantage of this detection method is that it solely relies on protocol messages. The disadvantage is that in an extreme case, if, by chance, all re-routed flows refresh at the end of the first refresh interval after re-routing, they cannot be quickly detected.

This observation can be used to create two new admission control override solutions.

### B.1 Greedy Blocking

If re-routed flows are detected at the end of a cell, a greedy approach is that the router does not accept any new flows for a complete refresh interval. The advantage of this approach is that it certainly prevents a new congestion and therefore it keeps the QoS agreements of previously admitted flows. The disadvantage is that during greedy blocking the utilisation may be poor since re-routed flows might not have caused overload (i.e., there is some free capacity to admit new flows even after re-routing), or during the 30 seconds some existing flows may leave the network, thus freeing up capacity for new flows.

### B.2 Refresh Estimation

To avoid the disadvantages of greedy blocking, we can squeeze out more knowledge from the refresh messages at the end of the cell. That is, if *refcount* is greater than $C_N$, then this can be used not only to conclude that there were re-routed flows, but their difference also gives the volume of the additional traffic that refreshed in the cell. Of course, we cannot know the exact volume of re-routed flows that will refresh in forthcoming cells, but we can make an estimation. Formalized, if re-routed flows are detected the first time, the additional traffic volume in the first cell ($A_1$) can be calculated as

$$\text{if } (C_N + L < \mathit{refcount}) \text{ then } A_1 \leftarrow (\mathit{refcount} - C_N).$$

That is, we now know the additional traffic volume in the first cell, but we need an estimation for how many re-routed flows will refresh in the remaining $(N-1)$ cells

$$\mathit{est} \leftarrow (N-1) \cdot A_1. \tag{13}$$

The estimation may seem too simple but we have to mention that the router has absolutely no information about the distribution of the refresh messages of the re-routed flows (which is the same as their connection arrival distribution), it can only suppose a uniform distribution. However, this assumption can be scientifically supported. Opposed to packet arrivals, session arrivals can be modelled with a Poisson-arrival process even for the Internet [35], and it can be shown that Poisson-arrival instants are uniformly distributed along the time axis [36].

*est* is calculated by (13) before shifting the window according to (12). Therefore, *last* is going to contain the re-routed flows of the just ended cell, since they were counted in *refcount*. However, *last* is missing the additional traffic that is going to refresh later and what we just estimated. That is, the override admission condition should be the following after the first cell that received re-routed traffic

$$\mathit{last} + \mathit{est} + p \leq T. \tag{14}$$

As time progresses and cells are shifting one can improve *est* step-by-step. At the end of $i$-th cell after re-routing has been detected, the additional traffic volume learnt so far is

$$A_i \leftarrow A_{i-1} + \mathit{refcount} - C_N.$$

Note that since cells are shifted, at every step one can use (the just current) $C_N$. From the additional traffic in the first $i$ cells, one can estimate the additional traffic in the remaining $(N-i)$ cells as

$$\mathit{est} \leftarrow \frac{N-i}{i} \cdot A_i.$$

When $i$ reaches $N$, *est* will be zero, which also supports the fact that after $N$ cells, that is, after a refresh interval, there is no need for admission control override any longer, since *last* will be correct.

The advantage of this approach is that it does not require bandwidth measurement but it still allows higher network utilization than greedy blocking. The disadvantage of refresh estimation is that if the distribution of refreshes is not uniform along the time axis, *est* will not be a good estimation of re-routed traffic. For example, if most re-routed flows refresh early, *est* will be an over-estimation. On the other hand, if many re-routed flows will refresh only at the end of the refresh interval *est* will under-estimate the traffic volume.

## VIII. SIMULATION RESULTS

For simulations, we used the discrete-event, packet-level network simulator (ns) tool [37]. We implemented RMD ourselves, and we used Marc Greis' RSVP/ns implementation [38], which we updated to suit the latest ns version.

First, we present the simulation results on the simplest topology where re-routing can occur, namely on a three-node delta network. After that, we show the results of a more complex scenario to underline that the solutions given here also behave similarly in a realistic scenario.

### A. Delta Network

The simple delta network scenarios presented in this subsection make it possible to easily follow the events occurring during and after the re-routing transient.

The topology consists of routers $A$, $B$, and $C$, see Fig. 6(a), the interconnecting links were set to 10 Mb/s with 2 milliseconds propagation delay. The admission control thresholds were set to 100% capacity on all links, i.e., $T = T_{\mathrm{RR}} = 10$ Mb/s.

We generated three types of CBR traffic. The aggregated data rate of the flows was 2 Mb/s in each traffic type, so in the end, 6 Mb/s was occupied by the whole traffic mix meaning 60% link utilisation. The first traffic type consisted of 16 kb/s streaming voice flows, the second type consisted of 64 kb/s low quality video and the third type of flows were 256 kb/s high quality video streams. The flows are randomly generated during the startup phase. Every flow is admitted and starts to send its data packets. At the 100-th second, link $A-C$ goes down, so its flows are re-routed to the $A-B-C$ path. The result figures will show the real (measured) user data load and the traffic volume derived from the reservation states. The first one characterizes the end-user experience, while the second one characterizes the precision of normal admission control.

In the first example, see Fig. 6, only node $A$ generated the above 6 Mb/s traffic mix towards node $C$, so after re-routing there is no congestion on the new path. This example shows the convergence speed of the reservation states of link $B-C$ to the real traffic load. After re-routing both RSVP and RMD require one refresh period to precisely set their reservation states, since

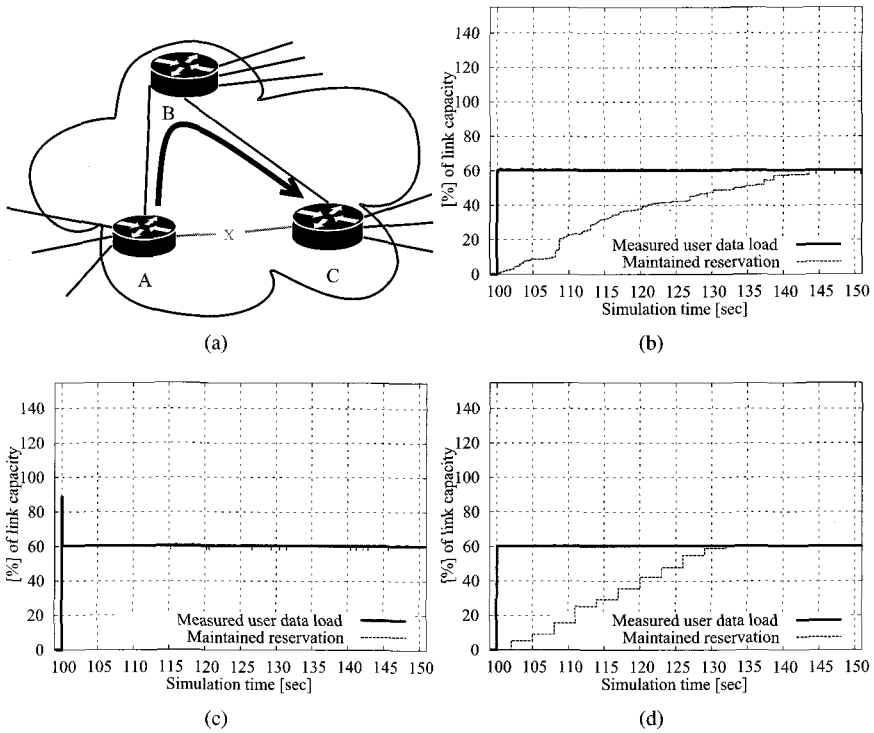


Fig. 6. Example without immediate congestion and without sudden traffic burst after re-routing (link $B - C$): (a) Scenario, (b) RSVP, (c) RSVP with local repair, (d) RMD.

the flows will refresh their reservation on the new path uniformly distributed. Moreover, it is easy to see that the *last* variable of RMD, which is plotted, is updated with the refreshed bandwidth only at cell boundaries, hence the stepwise figure.

If RSVP interacts with the routing protocol and uses its local repair feature, the reservation states are accurate almost instantly after re-routing (precisely after 53 ms). Therefore, in the plotted time-scale the bandwidth derived from the reservation states and the measured traffic load coincide. Though, Fig. 6(c) also shows that during the short repair transient, RSVP signaling messages occupy quite some bandwidth.

This example demonstrated a scenario, where re-routing did not have serious consequences on the perceived end-user quality. That is, congestion did not occur.

In the second example, node $B$ also generated a similar 6 Mb/s traffic mix towards node $C$ like node $A$, see Fig. 7. As a result, the flows are re-directed to an already occupied $B - C$ link, therefore this link immediately gets overloaded. The overload is 20% (2 Mb/s). Neither the basic RSVP nor the basic RMD recognises this immediately (Figs. 7(b) and 7(d)). The re-routed flows start refreshing on the new path but until the reservation variables of the link do not represent an overload, every refresh is admitted. Only after the reservation variables reach the threshold, will the refreshes get refused and the corresponding flows get terminated after the protocol informs the respective ingress nodes. However, until all the excess flows are dropped, the data rate of the arriving traffic is higher than the link capacity so there is severe packet loss experienced on link $B - C$ for a prolonged time.

RSVP with local repair (Fig. 7(c)) does not have to wait for regularly timed refreshes, it is again able to immediately refresh the re-routed flows throughout the new path, which is followed by a quick termination of some flows corresponding to 2 Mb/s. RMD, on the other hand, relies on the severe congestion handling algorithms to dissolve congestion (Fig. 7(e)). It is done in 127 ms, so the end-user experiences a similar behaviour like with RSVP local repair. Although internally in the transport network the reservation states are not correct even after severe congestion handling, in this example it is not a problem, since no new flows arrive where the reservation states would play a role in the admission decision.

In the previous two scenarios, one can observe how the reservation states converge to the precise value with RMD and also with RSVP when local repair is not used. However, we are yet to show what happens if new flow requests (represented by a dashed arrow in Fig. 8(a)) arrive during this imprecise period. To this end, we did another bunch of simulations where the traffic scenario was the same until 101 seconds, but after that a sudden and intense burst generated a lot of new flow requests from all three types of traffic from node $B$ towards $C$. The traffic burst was so intense that the generated traffic, if admitted, increases to 40% of the link capacity during a fraction of the refresh interval. The result is that if the reservation states are not correct, even in the first example, where there was no immediate congestion just because of re-routed flows, overload and congestion occurs.

The results of the first example with a sudden traffic burst can be seen on Fig. 8. First look at the basic RSVP (Fig. 8(b)) and RMD (Fig. 8(d)). You can see that both blindly admit new flows until around 114 seconds, however from around 108 seconds the link is overloaded, and the admitted flows in reality cause severe congestion, which is definitely in contrast with the purpose of admission control and resource reservation. Moreover,

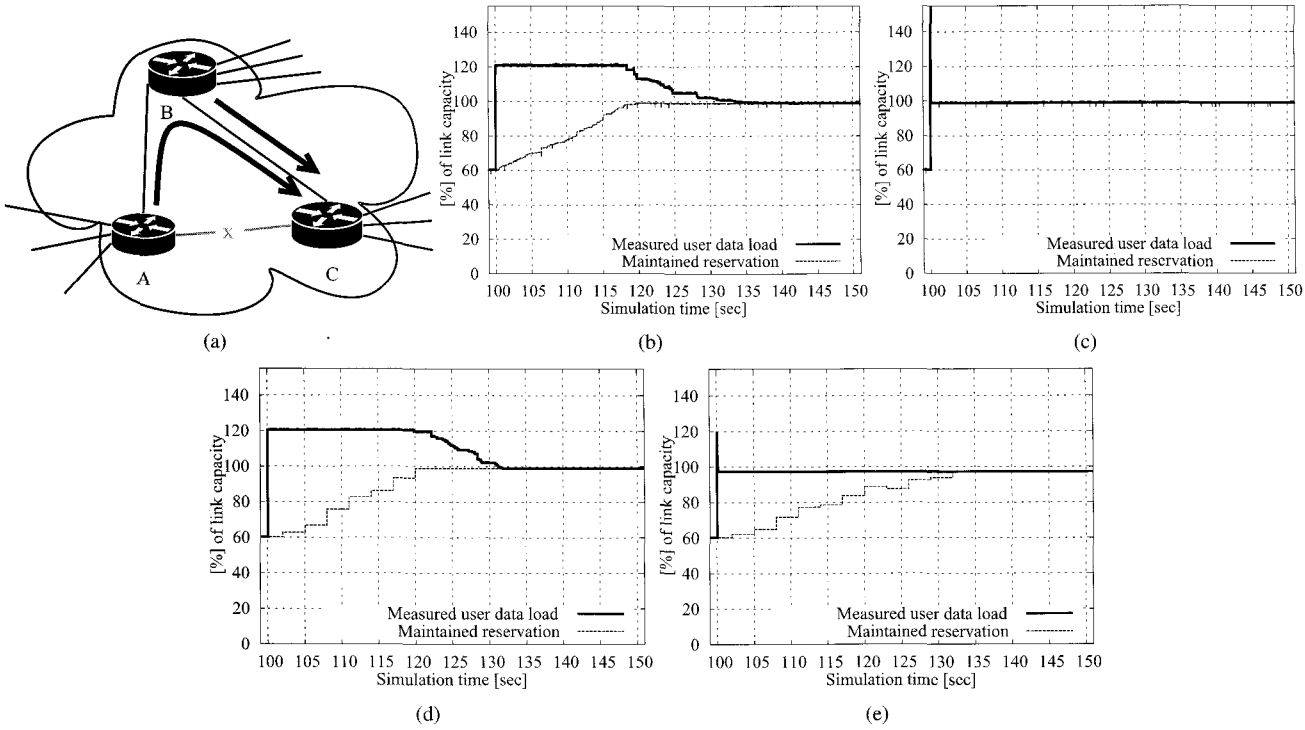


Fig. 7. Example with severe congestion and without sudden traffic burst after re-routing (link $B - C$): (a) Scenario, (b) RSVP, (c) RSVP with local repair, (d) RMD, (e) RMD with severe congestion handling.

after 114 seconds, excess flows slowly get terminated so that, in the end, after a refresh interval the situation is normalised. RSVP with local repair (Fig. 8(c)) again brings the ultimate performance: Since the reservation state is precise, only that much flows are admitted that can really be served, so it avoids all adverse effects.

RMD with severe congestion handling (Fig. 8(e)) begins just like the basic RMD, since there is no immediate severe congestion after re-routing. As soon as the arriving data rate reaches the $T_{\mathrm{RR}} = 100\%$ threshold, the algorithm reacts and terminates some flows to decrease the utilization below the threshold. However, without intervention the RMD admission control still admits newly arriving flows based on its incorrect reservation variables. The result is a series of recurring severe congestion—fall-back transients, until the reservation variables finally reach $T_{\mathrm{RR}}$, where new flows are not admitted anymore. Although packet drops did not occur because the severe congestion handling algorithm is quick enough to terminate the required number of flows, the provided quality is still not optimal because of the forced termination of flows. While RSVP with local repair works preventively, the severe congestion algorithm in RMD is just a reactive algorithm which quickly solves an acute problem but cannot solve the overall problem in the background, which is the incorrect reservation states.

To overcome this problem, we implemented and applied the override algorithms described in Section VII. The pure measurement based algorithm (Fig. 8(f)) works perfectly since it quickly measures the real user data load and so it can block newly arriving traffic that otherwise would be in excess of the link capacity. The refresh estimation, see Fig. 8(g), is quite accurate so it can also avoid congestion. Although with these two methods the variables themselves are not repaired, the false admission decisions are overridden. As a result the perceived network behaviour is similar to the reference RSVP local repair solution. The greedy blocking method (Fig. 8(h) is very conservative. It blocks all newly arriving flows after detecting the presence of re-routed flows, but the achieved utilization during this 30 seconds is low: 6 Mb/s compared to the 10 Mb/s link capacity.

If a burst of new requests appears in the second example, as shown in Fig. 9, there is immediate severe congestion plus the newly admitted flows also stress the network. The results show similar behaviour as in the previous scenario. However, in this scenario the simple greedy blocking approach proves to be as efficient as RSVP with local repair or RMD with measurement based override. In turn, the refresh estimation is not quick enough. At least, it could not avoid a smaller second congestion, which formed before the end of the first cell.

With the help of the delta network scenarios, we demonstrated $i$) the slow convergence speed of the reservation states due to regular soft-state refreshes; $ii$) the overload and severe congestion after re-routing; $iii$) the fast local repair solution of the stateful RSVP; $iv$) the similarly quick severe congestion handling solution of RMD; $v$) the effect of incorrect admission decisions in the form of overload or recurring severe congestions; and $vi$) the success of our proposed admission control override solutions in avoiding new congestions.

### B. Complex Network

We also simulated a backbone topology with a more realistic traffic model. We used Aleron's backbone network topology[2]

[2]Source: `http://www.aleron.com/access/map.html`.

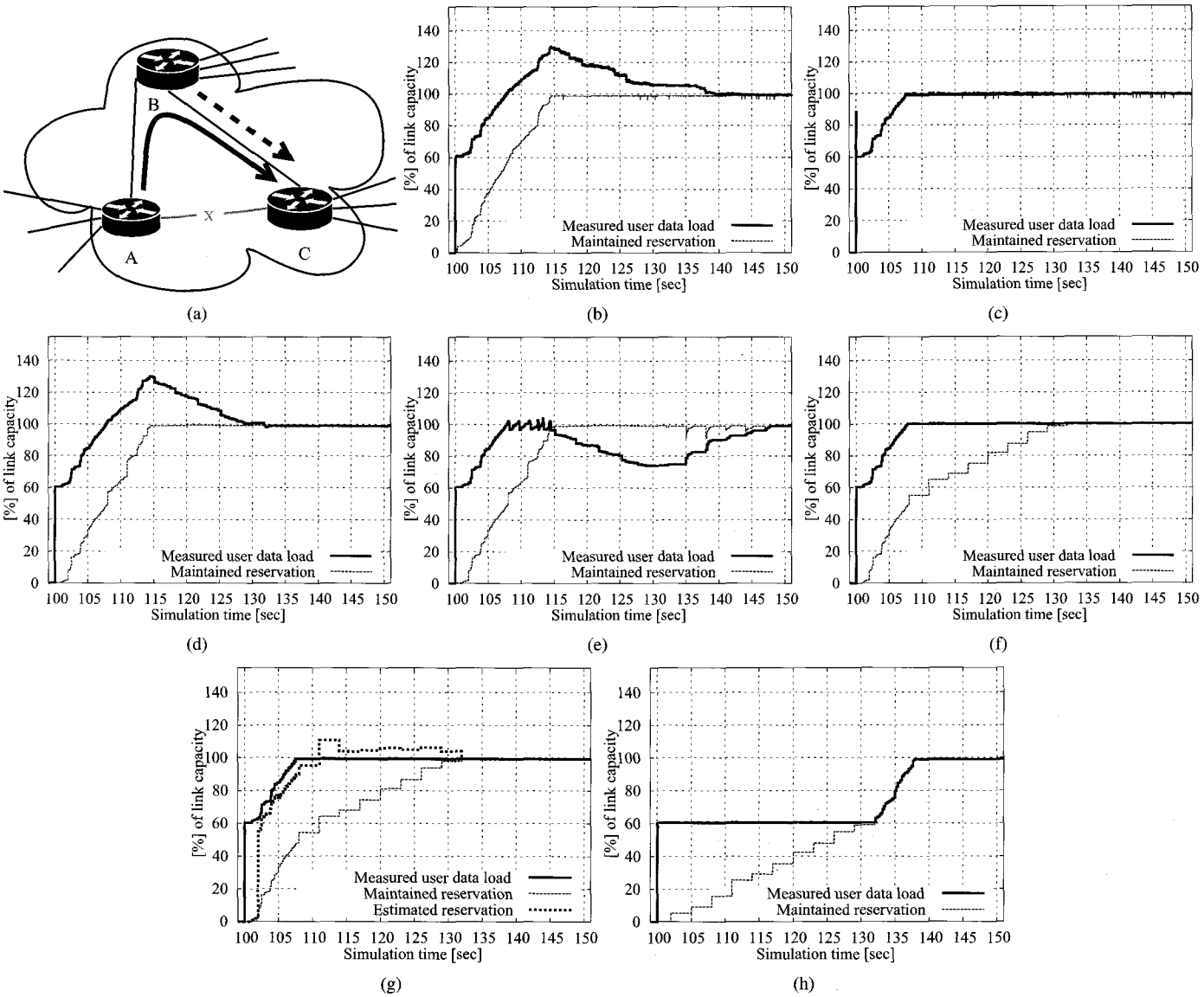


Fig. 8. Example without immediate congestion but with sudden traffic burst after re-routing (link $B - C$): (a) Scenario, (b) RSVP, (c) RSVP with local repair, (d) RMD, (e) RMD with severe congestion handling, (f) RMD with pure MBAC override, (g) RMD with refresh estimation, (h) RMD with greedy blocking.

shown on Fig. 10(a). We simulated a video library server in Los Angeles, which offered 64 kb/s and 256 kb/s video streams for the clients connected to every other point-of-presence (POP) nodes. At every POP, requests arrived according to an independent Poisson-process for the two video types. According to [39], video lengths can be modelled with a normal distribution with 102 minutes average length and 16 minutes standard deviation. The video library contained seven H.263 VBR video traces in both rates acquired from [40]. The actual video was chosen with uniform distribution. The arrival intensity of the 64 kb/s flows were set to four times of the 256 kb/s flows so that on the average the two traffic types occupy the same aggregated bandwidth.

The traffic class of the video streams was configured with an admission threshold of $T = T_{\mathrm{RR}}$ as 50% of the link capacity on every link. Note that we do not address the effective bandwidth and equivalent capacity questions of the variable bit-rate sources (which are otherwise important questions). In this scenario we had so many flows that the reservation counted as the sum of the average rates is very close to the actual traffic load.

On this topology five destinations used the link between Los Angeles and Palo Alto, while the other four destinations used the link between Los Angeles and Dallas. The arrival intensities were set so that the latter link is utilized to around 25%. After 1500 seconds simulation time the link between Los Angeles and Palo Alto failed, so every flow was directed towards Dallas, utilizing this link was to around 60%. Therefore, some flows have to be terminated in order to take the utilization back to the 50% admission threshold. Afterwards, only as many new sessions may be admitted as left the network in the meantime. We again simulated an increased traffic burst by rising the arrival intensity of new flow requests a few seconds before re-routing to ten times of the original.

In Fig. 10, similar observations can be made for the critical

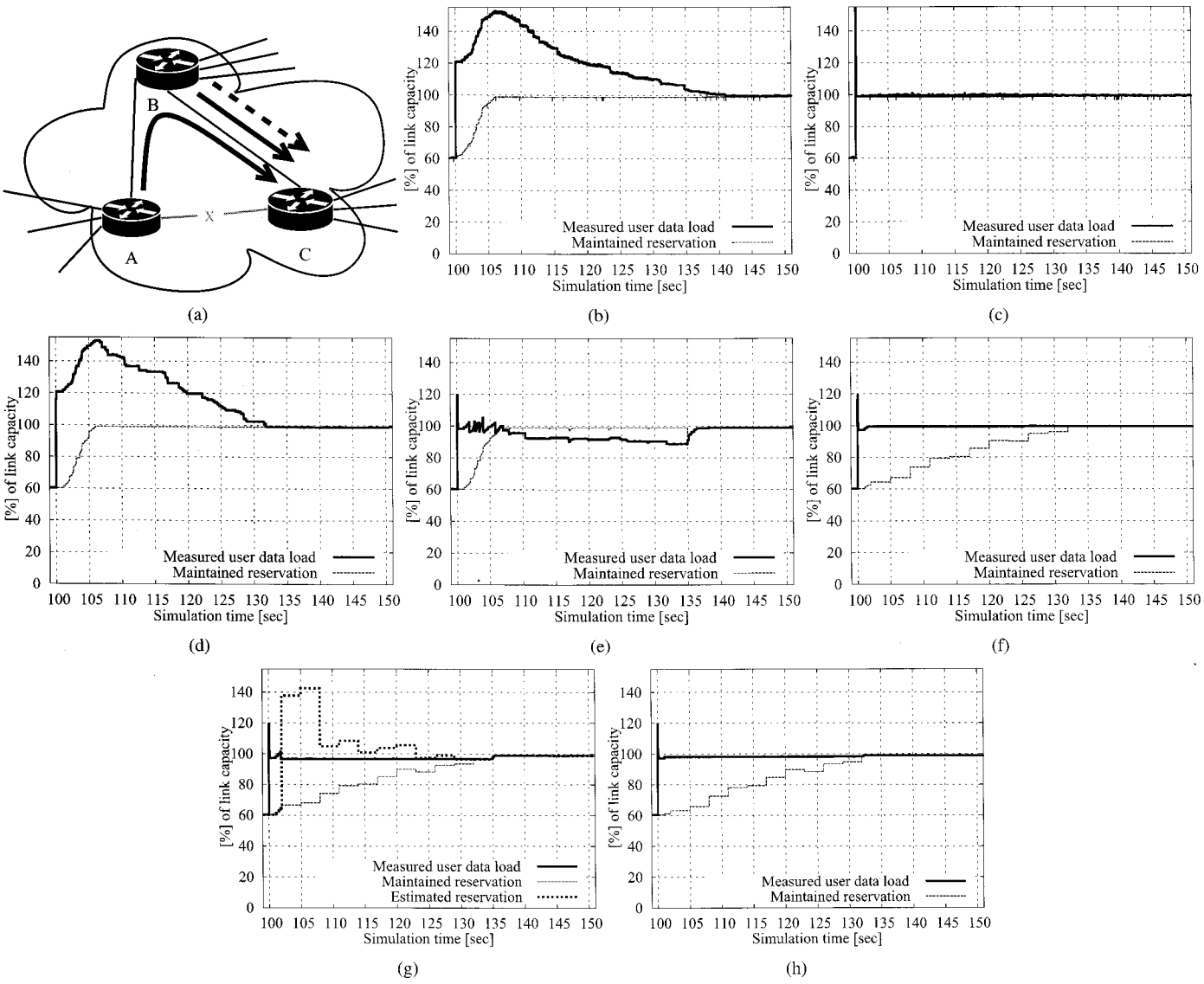


Fig. 9. Example with immediate severe congestion and with sudden traffic burst after re-routing (link $B-C$): (a) Scenario, (b) RSVP, (c) RSVP with local repair, (d) RMD, (e) RMD with severe congestion handling, (f) RMD with pure MBAC override, (g) RMD with refresh estimation, (h) RMD with greedy blocking.

link between Los Angeles and Dallas like on the $B-C$ link of the delta topology. The basic RSVP (Fig. 10(b)) and RMD (Fig. 10(d)) falsely admit many flows, even though the utilization is immediately higher than $T$ after re-routing. RSVP with local repair (Fig. 10(c)) quickly repairs its states, therefore, it allows only a few new flows to fill in the gap of the flows left in the meantime.

RMD with only severe congestion handling (Fig. 10(e)) terminates the necessary amount of flows within 380 ms, however its false reservation states let it admit new flows, which after some time again cause congestion, and we can again identify the recurring pattern of congestions like on the delta scenario. Here we must point out that the setup of the measurement algorithm for severe congestion detection was not so trivial as with static traffic. We had to use an exponential weighted moving averager (EWMA) to smooth out the fluctuations of the short time-scale measurements but a very smoothed bandwidth estimation resulted in slow congestion detection. In the end, we found that an $S = 50$ msec and an EWMA weight of 0.5 gives a quickly reacting yet smooth enough bandwidth estimation.

The measurement based override (Fig. 10(f)) and refresh estimation based override (Fig. 10(g)) provide similar performance like local repair. Greedy blocking (Fig. 10(h)) is too conservative since during 30 seconds it does not let new flows to take advantage of the 5% capacity released in the meantime.

Finally, to show the fulfillment of Requirement 2, we changed $T_{\mathrm{RR}}$ to 60%, but we left $T = 50\%$. This is to ensure that after re-routing the admitted flows can refresh their reservation 10% higher than the admission threshold for newly arriving flows. The 10% extra capacity is temporarily borrowed, e.g., from the best-effort class. In such a situation the greedy blocking CAC correction method has the best cost/performance ratio since it is the simplest method, and since re-routed flows utilize the link higher than the admission threshold, the other methods would also not admit any newly arriving flows.

The result is shown on Fig. 10(i). Here, flows are only termi-

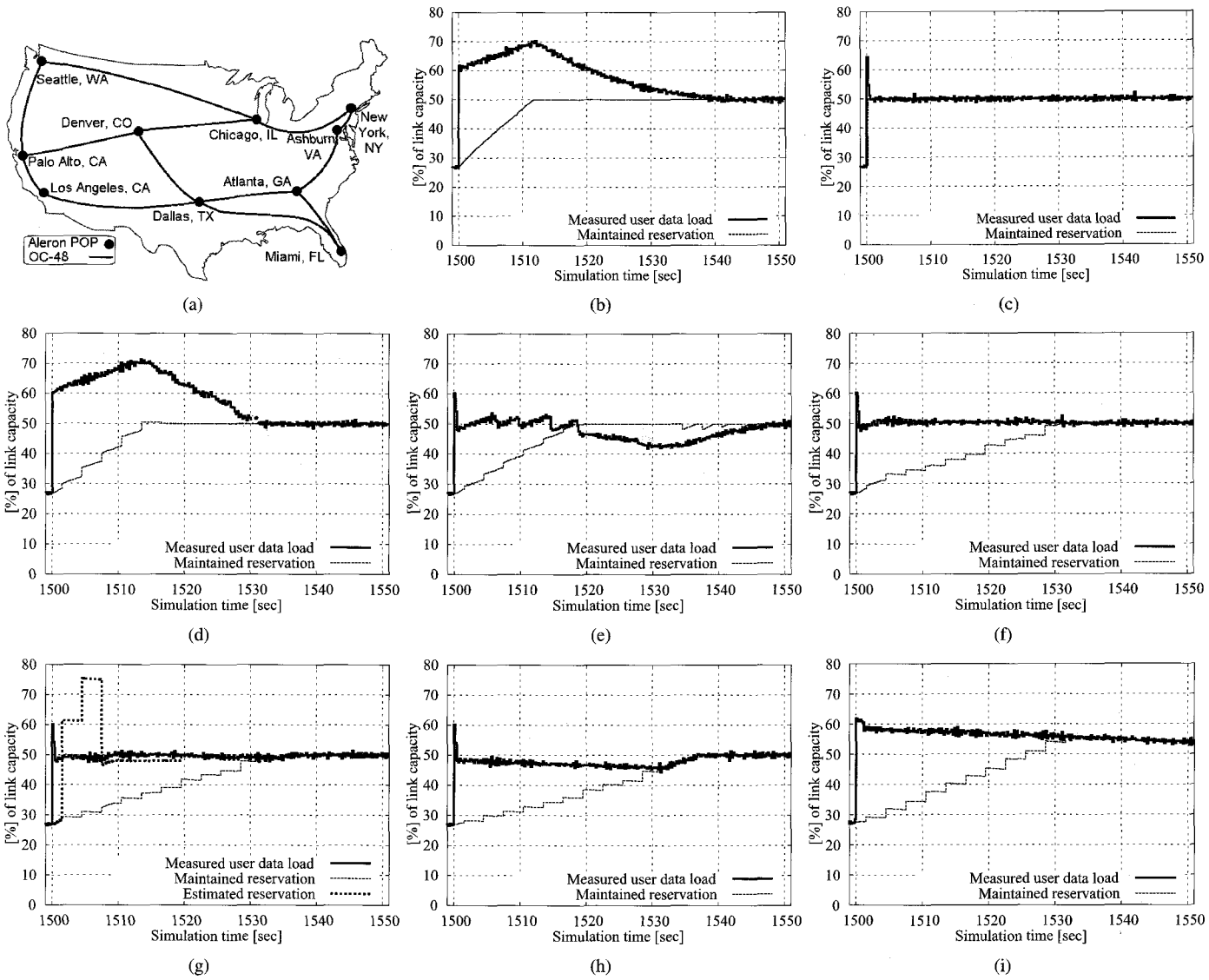


Fig. 10. Simulation of a complex scenario (link Los Angeles–Dallas): (a) Topology, (b) RSVP, (c) RSVP with local repair, (d) RMD, (e) RMD with severe congestion handling, (f) RMD with pure MBAC override, (g) RMD with refresh estimation, (h) RMD with greedy blocking, (i) RMD with greedy blocking, $T_{\mathrm{RR}}$=60%.

nated as long as the load of the QoS class is above 60%. After that flows are not forced to terminate, they leave by themselves. With the simulated traffic model the link was utilized above 50% until 1590 seconds simulation time and until that no new QoS flows were admitted.

## IX. CONCLUSION

In this article, we have shown that a reduced-state reservation protocol, specifically RMD, can be made resilient to re-routing. Since re-routings occur frequently in operational networks, the consequences must be addressed appropriately. We have shown that re-routing may cause severe congestion and incorrect admission decisions due to outdated reservation variables. To address these problems we not only gave requirements on what we expect from a solution resolving these problems, but we also presented appropriate solutions.

In case of a severe congestion, one has to sacrifice and terminate some flows in order to ensure proper QoS for the remaining ones. Since core nodes do not dispose of per-flow states we gave an in-band signaling solution to let the edge nodes learn the presence and the volume of congestion, after which the edge nodes are able to terminate the needed amount of flows. The proposed method is capable of preferring certain flows selected by the operator policies, and it is flexible in the sense that it is capable of letting re-routed flows borrow bandwidth from other traffic classes. In these respects, our methods are even better than the local repair of the stateful RSVP. Simulation results showed the effectiveness of our proposals.

Considering incorrect admission decisions, we proposed that after re-routing, another admission control algorithm has to override the bad admission decisions of the primary CAC as long as the reservation states are not corrected. A pure MBAC method, not relying on any stored states, is a good solution. By simulations we confirmed that it was quick and precise in each and every case, providing a similar end-user experience

as the quick stateful solution RSVP local repair. For the case, when measurements cannot be used we proposed a method to detect re-routed flows only from their refresh messages. After detection, the greedy blocking method does not admit any new flows during the refresh interval. The refresh estimation method collects even more knowledge from the refreshes by estimating the volume of re-routed flows. Simulations also confirmed that greedy blocking is conservative because it does not admit new flows, even if it could, resulting in low utilisation in certain cases. Although the refresh estimation method also worked well, greedy blocking is probably preferable because it gives 100% guarantee that new flows will not cause congestion, and low utilisation for 30 seconds may be a tolerable price for that.

All in all, we proposed extensions to the reduced-state reservation protocol RMD, and by means of a series of simulations we demonstrated that it can be made practically as resilient as the stateful RSVP protocol. From the aspect of preferential treatment of terminated flows, our solution is even better than the stateful solution, since it is capable of, e.g., keeping emergency calls.

Future work includes the development of a bandwidth measurement solution that is smooth enough not to react to regular traffic oscillation, and at the same time it should be capable of quickly detecting bigger shifts in the bandwidth (due to re-routing). Another research direction is the investigation of how congestion decreases when users "self-terminate" their sessions due to poor quality during severe congestion.

## ACKNOWLEDGMENT

The authors would like to express their special thanks to Péter Barta (Ericsson Research) for his rigorous pre-submission review of the revised manuscript.

## REFERENCES

[1] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurement from the sprint IP backbone," *IEEE Network Mag.*, Nov. 2003.

[2] "Next-steps in signaling (NSIS)," IETF working group web page.

[3] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next steps in signaling (NSIS): Framework," *RFC 4080*, IETF, June 2005.

[4] A. Báder, L. Westberg, G. Karagiannis, C. Kappler, and T. Phelan, "RMD-QOSM—the resource management in DiffServ QoS model," *Internet Draft*, draft-ietf-nsis-rmd, IETF, June 2005, work in progress.

[5] G. Karagiannis, A. Báder, G. Pongrácz, A. Császár, A. Takács, R. Szabó, and L. Westberg, "RMD—a lightweight application of NSIS," in *Proc. Networks 2004*, (Vienna, Austria), June 2004, pp. 211–216.

[6] L. Westberg, A. Császár, G. Karagiannis, A. Marquetant, D. Partain, O. Pop, V. Rexhepi, R. Szabó, and A. Takács, "Resource management in DiffServ (RMD): A functionality and performance behavior overview," in *Proc. PfHSN 2002*, (Berlin, Germany), vol. 2334 of *Lecture Notes in Computer Science*, Springer Verlag, Apr. 2002, pp. 17–34.

[7] A. Császár and A. Takács, "Comparative performance analysis of RSVP and RMD," in *Proc. QoFIS 2003*, (Stockholm, Sweden), vol. 2811 of *Lecture Notes in Computer Science*, Springer Verlag, Oct. 2003, pp. 41–51.

[8] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE INFOCOM 2003*, (San Fransisco, CA, USA), Mar. 2003.

[9] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE INFOCOM 2004*, (Hong Kong), Mar. 2004.

[10] L. W. McKnight and J. P. Bailey, eds., *Internet Economics*, The MIT Press, 1998.

[11] C. Filsfils and J. Evans, "Engineering a multiservice IP backbone to support tight SLAs," *Computer Networks: The Int. J. Computer and Telecommun. Networking, Special Issue: Towards a New Internet Architecture*, vol. 40, pp. 131–148, Sept. 2002.

[12] Internet 2 QBone Bandwidth Broker Advisory Council.

[13] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *RFC 2638*, July 1999.

[14] P. Ji, Z. Ge, J. Kurose, and D. Towsley, "A comparison of hard-state and soft-state signalling protocols," in *Proc. ACM SigComm 2003*, (Karlsruhe, Germany), Aug. 2003, pp. 251–262.

[15] L. Delgrossi and L. Berger, "Internet stream protocol version 2 (st2)," *RFC 1819*, IETF, Aug. 1995.

[16] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP)—version 1 functional specification," *RFC 2205*, IETF, Sept. 1997.

[17] P. Pan and H. Schulzrinne, "YESSIR: A simple reservation mechanism for the internet," *ACM SIGCOMM Computer Commun. Review*, vol. 29, pp. 89–101, Apr. 1999.

[18] A. Eriksson and C. Gehrmann, "Robust and secure light-weight resource reservation for unicast IP traffic," in *Proc. IWQoS'98*, (Napa, CA, USA), May 1998, pp. 168–170.

[19] G. Fehér, K. Németh, M. Maliosz, I. Cselényi, and J. Bergkvist, "Boomerang a simple protocol for resource reservation in IP networks," in *Proc. IEEE Workshop on QoS Support for Real-Time Internet Applications*, (Vancouver, Canada), June 1999.

[20] Z. R. Turányi and L. Westberg, "Load control: Congestion notifications for real-time traffic," in *Proc. 9th IFIP Working Conference on Performance Modelling and Evaluation of ATM and IP Networks*, (Budapest, Hungary), July 2001.

[21] E. Ossipov and G. Karlsson, "SOS: Sender oriented signaling for a simplified guaranteed service," in *Proc. QofIS 2002*, (Zurich, Switzerland), Oct. 2002, pp. 100–114.

[22] A. Császár, A. Takács, R. Szabó, V. Rexhepi, and G. Karagiannis, "Severe congestion handling with resource management in DiffServ on demand," in *Proc. Networking 2002*, vol. 2345 of *Lecture Notes in Computer Science*, (Pisa, Italy), Springer Verlag, May 2002, pp. 443–454.

[23] L. Westberg, M. Jacobsson, G. Karagiannis, S. Oosthoek, D. Partain, V. Rexhepi, R. Szabó, and P. Wallentin, "Resource management in DiffServ (RMD) framework," *Internet Draft*, draft-westberg-rmd-framework, IETF, Sept. 2003, work in progress.

[24] L. Westberg, M. Jacobsson, G. Karagiannis, M. de Kogel, S. Oosthoek, D. Partain, V. Rexhepi, and P. Wallentin, "Resource management in DiffServ on demand (RODA) PHR," *Internet Draft*, draft-westberg-rmd-oda-phr, IETF, Sept. 2003, work in progress.

[25] J. Manner and X. Fu, "Analysis of existing quality-of-service signaling protocols," *RFC 4094*, IETF, May 2005.

[26] D. Vali, S. Paskalis, and A. Kaloxylos, "A survey of Internet QoS signaling," *IEEE Commun. Surveys & Tutorials*, vol. 6, no. 4, pp. 32–43, 2004.

[27] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," *RFC 3209*, IETF, Dec. 2001.

[28] J. Manner, G. Karagiannis, A. McDonald, and S. V. den Bosch, "NSLP for quality-of-service signaling," *Internet Draft*, draft-ietf-nsis-qos-nslp, IETF, July 2005, work in progress.

[29] A. Marquetant, O. Pop, R. LSzabó, G. Dinnyés, and Z. Turányi, "Novel enhancements to load control—a soft-state, lightweight admission control protocol," in *Proc. QofIS 2001*, (Coimbra, Portugal), vol. 2156 of *Lecture Notes in Computer Science*, Springer Verlag, Sept. 2001, pp. 82–96.

[30] "Technical report on enhanced network survivability performance," Tech. Rep. T1.TR.68, ANSI, Feb. 2001.

[31] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier 1 backbone," *IEEE Network*, vol. 18, pp. 13–19, Mar. 2004.

[32] A. Császár, A. Takács, and A. Báder, "A practical method for the efficient resolution of congestion in an on-path reduced-state signalling environment," in *Proc. IWQoS 2005*, (Passau, Germany), no. 3552 in *Lecture Notes in Computer Science*, June 2005, pp. 282–293.

[33] A. Császár, A. Takács, R. Szabó, and T. Henk, "State correction after rerouting with reduced state resource reservation protocols," in *Proc. IEEE GLOBECOM 2004*, (Dallas, TX, USA), Nov. 2004.

[34] L. Breslau, S. Jamin, and S. Shenker, "Comments on the performance of measurement-based admission control algorithms," in *Proc. IEEE INFOCOM 2000*, (Tel-Aviv, Israel), Mar. 2000.

[35] V. Paxson and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Trans. Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[36] H. Bruneel and B. G. Kim, *Discrete-Time Models for Communication Systems Including ATM*, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, 1993.

[37] K. Fall and K. Varadhan, *ns Manual*, UC Berkeley.

[38] M. Greis, "RSVP/ns: An implementation of rsvp for the network simulator ns-2," document, Computer Science Department, University of Bonn, 1998.
[39] S. W. Carter, D. D. E. Long, and J.-F. Pâris, "An efficient implementation of interactive video-on-demand," in *Proc. MASCOTS 2000*, (San Fransisco, CA, USA), Aug. 2000, pp. 172–179.
[40] "Video traces for network performance evaluation," Tech. Rep., Arizona State University.

**András Császár** received his M.Sc. degree in computer science from the Budapest University of Technology and Economics in 2001. Between 2001 and 2004, he attended the Ph.D. course at the same university at the Dept. of Telecommunication and Media-Informatics with scholarship. He has attended a post-gradual course in banking informatics and received his degree in 2005. Since 2001, he is a research fellow at TrafficLab, Ericsson Research Hungary. His research is dedicated to IP networks, and his interests include resource reservation and congestion handling in high-speed networks, load sharing, routing protocols, and resilience.

**Attila Takács** received his M.Sc. degree in computer science from the Budapest University of Technology and Economics (BUTE) in 2001. Following graduation, he started his Ph.D. studies at BUTE working in the field of IP based telecommunication networks. He has attended a post-gradual course in banking informatics and received his degree in 2005. Since 2001, he is a research fellow at TrafficLab, Ericsson Telecommunications Hungary. His research interests include resource reservation, congestion handling, load sharing, and routing.

**Róbert Szabó** has received his M.Sc. in electrical engineering from Budapest University of Technology (BUTE) in 1996. He followed his studies in the same institution and earned his Ph.D. in 2002. Since 1999, he is an employee of the BUTE, currently as associate professor. In 2003, he obtained the Bolyai scholarship of the Hungarian Academy of Science. He leads the Quality of Service Information Technologies Laboratory, part of the Department of Telecommunication and Media-Informatics, BUTE. Since 2005, he is the president of the Telecommunications Section of the Scientific Association for Infocommunications, Hungary. His main research interests are architectures, protocols, and performance of communication networks. He co-authors more than 10 journal and 30 conference papers.

**Tamás Henk** received his M.Sc. degree with an excellent certificate from the Budapest University of Technology and Economics in 1973. He received a candidate degree from the Hungarian Science Academy in 1985, which was honoured by the same university with a Ph.D. degree in 1997. He is currently an associate professor at the Dept. of Telecommunication and Media-Informatics. He is a founder and head of the High Speed Networks Laboratory at the university. He is also the research project director of the Inter-University Centre for Telecommunications and Informatics in Hungary. His research interests include networking theory, electronic filters, high-speed communication networks, and space communication via satellites.