DiffRes: A Light Weight Reservation Protocol for the Differentiated Services Environment

Dorgham Sisalem, Srikanth Krishnamurthy, Son Dao
HRL Laboratories, LLC.
Malibu, CA
{dor,krish,son}@wins.hrl.com

Abstract— **In lieu with current efforts for building QoS models for the Internet, DiffRes provides a light weight QoS signaling protocol that allows for per-flow reservation establishment without the need for maintaining state information in the core routers. In this paper, we describe the functionality and design of the DiffRes protocol and possible interaction with other QoS signaling protocols.**

I. Introduction and Background

The combination of a simple networking architecture and congestion control mechanisms at the end systems were the prevalent reasons for the success of the Internet. In an environment dominated by time insensitive applications such as web surfing and media streaming QoS can be simply measured as the average throughput of an application. However, the QoS of recently proposed real-time multimedia services such as IP-telephony and group communication will be measured additionally by temporal loss and bandwidth values as well as delays suffered by the data packets. Guarantees for loss, delay and bandwidth can only be provided by introducing QoS control mechanisms that only admit data flows to the network if enough resources are available for supporting the QoS requested by those flows.

The integrated service (IntServ) model [1] provides a flexible architecture in which each end system can ask for a specific service with specific parameters for its data flows. In case of resource availability, the flow is admitted to the network. Each traversed router on the path connecting the sender to the receivers maintains per-flow state information about the flow and its requested resources. Data packets are then classified and scheduled in accordance with the state information.

With the resource reservation protocol (RSVP) Braden et al. [1] have proposed a QoS signaling protocol, in which the sender informs the receivers about the traffic shape of the data flow. In reaction to that, the receivers send reservation requests indicating the amount of required resources to support the flow.

Each network node traversed by the control messages maintains information about the requested resources for each flow. Reservation requests by different receivers for the same data flow are merged together at the network nodes. For each flow that has made a reservation, the network nodes periodically send control messages to their neighboring nodes to indicate the state of this flow. As all the nodes need to maintain state information for each flow and send and receive control messages to and from neighboring nodes to detect the state of each flow, RSVP incurs a significant processing overhead on the network nodes. In addition, various problems resulting from issues like merging of reservations and reservation updates need to be considered with RSVP.

Pang and Schulzrinne propose a sender based reservation scheme called YESSIR [2] in which the sender issues the reservation request. Each traversed router processes the reservation request, decides to accept or reject it and establishes some state information for the signaling flow in case of acceptance. After receiving a reservation message from the sender, the receivers issue acknowledgment packets indicating the result of the reservation. After establishing a reservation, the sender periodically transmits control messages to refresh its state at the network nodes. This approach reduces the complexity of RSVP as it does not require the routers to exchange per-flow refresh messages. However, it still requires per-flow state information at the routers.

The need for maintaining per-flow states, periodically refreshing these information as well as the overhead of per packet classification and processing results in scaling problems for the IntServ model and RSVP when handling a large number of flows. The differentiated service (DiffServ) model [3] was designed to avoid this problem by using service level agreements (SLA) between the network providers and the users. These SLAs describe the QoS level the aggregated traffic of a user can expect from the provider. Traffic sent in conformance with the established SLA is marked as belonging to a specific QoS level. At the core routers data packets are serviced differently based on their marked QoS level. As all packets with the same marks are treated equally, the core routers need only to maintain information describing the resources allocated for the supported QoS levels.

As the SLAs are used for traffic aggregates, there is no need for per-flow states. Additionally, with the DiffServ model only the edge routers need to police incoming traffic and take admission control procedures. While such an approach avoids the scalability problems of the integrated service model it is rather rigid. SLA are currently mainly thought to be established in a static manner or to be changed only infrequently in the order of days or weeks. Hence, the DiffServ model does not allow the user to increase or decrease the amount of its reserved resources in accordance with its traffic requirements. In addition to the static nature of the SLAs specifying a QoS level for an aggregate of flows can result in unfair distribution of resources among the flows belonging to the same aggregate due to the aggressiveness of some flows, difference in round trip delays and taken paths[4]. Further, the core routers are expected to be dimensioned large enough in a manner as to provide the user with the agreed upon QoS level on any path taken by the user at any time in accordance with the SLA. As the exact paths taken by a user's traffic is not known in advance the network needs to be highly over provisioned to account for all possible cases. This is even more pronounced for the case of multicast as the user's traffic might take different paths in the providers network and hence enough resources need to be available on all paths simultaneously.

In this work, we present a reservation protocol (DiffRes) for the differentiated service model that combines the flexibility of the IntServ model and the simplicity of the DiffServ model. That is, the basic architecture allows the users to make per-flow reservations on the one side but keeps the core routers simple on the other. Issues of policing traffic and collecting per-flow information are only handled at the edge systems which are not expected to handle a large number of flows. QoS differentiation is realized by marking data packets of different flows differently in accordance with the amount of resources allocated for those flows.

The main concept in DiffRes is to use only temporary per-flow state information at the core routers during reservation setup and tear-down operations combined with per-hop acknowledgments of those operations between the core routers. By having the network informing the end systems about the progress of a reservation setup or tear-down operation, DiffRes avoids the problems that might arise with the loss of control messages.

End systems maintain information describing the reserved QoS and participating end systems and collect information describing the path taken by the data packets. By periodically updating these information end systems can detect cases of path changes or node failures.

Sec. II presents a brief overview of the problems DiffRes intends to solve and its architecture. Protocol messages and the flow of messages are described in Sec. III and Sec. IV. Issues of admission control and setting of protocol parameters are discussed in Sec. V and Sec. VI. Sec. VII describes some implementation options and Sec. VIII presents an interoperability scenario of DiffRes and RSVP. A summary and a look at future work are finally presented in Sec. IX.

II. Basic Architecture and Design Goals

The basic objective of DiffRes is to provide the end users with the means for signaling dynamic reservations that are suitable for their applications' needs. These reservations are made on an end-to-end basis without keeping state information at the core routers of the network. For the network provider, DiffRes is designed to allows for a simple network architecture that supports QoS differentiation. As resources are allocated dynamically when they are needed the network provider can use its available resources more efficiently and provide predictable QoS levels without having to over-dimension his network.

A. *Problems to be Solved*

The basic model for a dynamic reservation protocol is for an end system to send a control packet requesting a certain amount of resources. At each traversed hop, the routers either accept, reject or modify the request based on their current load state. After arriving at the receiving end system an acknowledgment of the reservation success or failure is generated and sent back to the sender.

Without any state information at the core routers there are different problems to be addressed in the context of such dynamical bandwidth reservations scenario:

• **Duplicate reservations:** After sending a reservation request the reserving entity expects a confirmation of the success or failure of the reservation. If after some time period no confirmation was received the reserving entity might retransmit the reservation packet. With no internal state information the core routers can not decide solely on the basis of the retransmitted packet if a reservation was already made for this flow or not. This problem is even more pronounced for the case of multicast reservations, as in this case receivers might join a session at different time points. Hence, for each new receiver resources need to be reserved on the path connecting the sender to this receiver. As multiple receivers are very likely to share parts of the multicast distribution tree they should share the same reserved resources as well. However, without any knowledge about which sessions have already reserved resources, the routers would probably end up establishing a new reservation for each new receiver. Both the multicast case and loss of control packets lead to an inefficient usage of resources. The same problem applies for the case of a reservation deletion request. In this case, one needs to avoid deleting resources more than once when a connection is terminated.

- **Path changes:** Internet measurements conducted by Paxon [5] suggest that while not very likely to happen often, it is nevertheless possible that after establishing a reservation for some flow on a specific path, the network changes the path over which the data packets of the flow are sent. As a result, the data packets of the flow would be carried over a path for which it made no reservation. As the packets of the flow would still be marked by the edge devices as having some specific QoS level, they would not be simply treated as best effort packets but as prioritized packets and compete for the resources reserved for the packets of other flows of the same priority. This means that the QoS level which this flow belongs to, might become congested and the reserved QoS is no longer assured.
- **Admission control:** To avoid overload situations, routers should only admit new flows if the amount of requested resources in addition to the already allocated resources are below some threshold. With RSVP, a core router can determine the amount of reserved resources based on the flows' state information. Additionally, a reservation for a specific flow is only maintained while the routers receive refresh messages indicating that the flow is alive. This allows the core routers to update their estimations of the amount of reserved resources without explicitly requiring a reservation tear-down. DiffRes routers, however, do not keep information about the amount of resources reserved for specific flows. Hence, in case of a node failure or path change a router will not be able to determine that the number of flows it needs to support has changed and hence it should actually change the amount of reserved resources. Therefore some method is required to keep the routers updated on the actual amount of resources consumed by the active flows.

B. Protocol Architecture

DiffRes is mainly an extension of the DiffServ model. That is, only edge devices need to maintain detailed per-flow state information describing the amount of consumed resources, QoS level and flow identity. Admission control, traffic policing and shaping is mainly realized at the edge devices. Data packets that obey their service profile, i.e, are sent in accordance with the reserved resources for their flow, are marked as belonging to a specific service class. Packets that do not obey the service class are either dropped [6], marked with a lower service class [7] or sent as best effort traffic. The marking of the packets can either be done by the end systems which have the knowledge about the importance of the single packets or by the edge devices, which have the knowledge about the aggregate rate of the traffic a user is generating. In either case, the edge devices need to implement some policing or shaping mechanisms to assure the conformity of the flows to their reserved resources [3].

The current approach for establishing a service profile is usually based on static service level agreements (SLA) between the service provider and the user. Thereby, the provider assures the user that the aggregated traffic generated by the user and conforming to the established SLA will receive the agreed upon QoS level.

DiffRes enhances this service model by allowing for dynamic resource reservation while maintaining a stateless core network. That is, inside the network reserved packets receive differentiated levels of service based on special markings in the packet header.

To enable the users to dynamically signal their requirements and adjust the dedicated resources for them in accordance with their actual needs, DiffRes extends the DiffServ model with an end-to-end as well as a hop-by-hop signaling scheme:

- **End-To-End signaling:** To make a reservation, the sender starts by transmitting a reservation message to a network edge device. The edge device registers the reservation and forwards the request to the core routers. Based on the service required and available resources the core routers might reject, accept or modify the received request and forward it to the next hop along the path to the receiver. The receiver sends then a feedback message to the sender indicating the result of the reservation request. If no feedback message was received after a period of time the sender reinitiates its reservation request. After establishing the reservation, the sender periodically transmits control messages to collect information about the resource availability in the network and the path taken by the flow. This allows the end systems to detect changes in the network load state and possibly adjust their resource consumption based on these information. Additionally, the end systems can detect cases of path changes and initiate a new reservation for the changed path.
- **Hop-by-hop signaling:** As control messages might get dropped in the network, end systems and core routers need to determine up to which router a reservation request was processed before getting discarded. Simply reissuing a reservation request in response to the loss of a previous one, would lead to duplicate reservations at nodes where the first request was already successfully processed. To avoid this case of duplicate reservations, a combination of

end-to-end and hop-by-hop signaling is used. This is realized by having each router include its address to an address list carried by the reservation message. After receiving a reservation request, the core routers maintain state information indicating the identity of the flow, the requested resources and the address list of the routers already traversed by the request carried as in the request as well as a timer indicating when these data become obsolete. In periods of T_{router}, the core routers inform their upstream neighbor routers about all reservation requests they received from them. Upon receiving a confirmation that the downstream router received a reservation request, the router deletes the entry for that request from its own list. If the timer associated with a request (Q_i) which was received from router (R_{l}) expired at router ($R_{\text{l+1}}$), $R_{\text{l+1}}$ includes in its periodic messages to R_{l} a negative acknowledgment with the address list and QoS information of Q_i. The negative acknowledgment is then forwarded with periodic the hop-by-hop router messages along the routers indicated in the reversed address list of request Q_i. Hence, the sender of Q_i would finally receive a message indicating the nodes for which a reservation was successfully made. It can retransmit the reservation message and include in it the list of addresses from the previously lost request. Routers that find their address in this list in the retransmitted request do not need to reestablish a new reservation for this request or maintain state information for it.

A successful reservation is finally indicated by a positive acknowledgment by the receiver. This acknowledgment needs to be intercepted by the edge device in order to initiate the policing and shaping functionalities. The amount and level of reserved resources is translated to a rate of marking of a specific QoS level. Traffic belonging to a flow with a reservation gets then marked based on the requested QoS level and its compliance with that level.

C. Scalability

IntServ capable routers need to maintain state information for each flow for the purposes of classifying the flow to a certain QoS level, scheduling the flow with a specific QoS in addition to information indicating the state of the flow and the amount of resources dedicated to it.

Restricting the QoS levels to only a few and using marks in the packet headers to indicate the level to which a packet belongs to reduces the task of classifying packets to a matter of reading the markings and assigning the packet to the appropriate QoS level. This eliminates the need for classification information. Stoica et al. [8] present an approach in which the scheduling information required for the case of guaranteed bandwidth and delay services can be eliminated as well by carrying some additional information in the packet headers.

With DiffRes the core routers need only to maintain per-flow state information for a short period of time during the reservation establishment or adjustment. As it is not to be expected that thousands of reservation requests would be issued at the same time, the amount of saved states would most probably be acceptable.

However, in contrast to RSVP for example, DiffRes does not support merging of receiver replies at intermediate routers. This means that each reservation request or update would trigger an acknowledgment by the receivers. Depending on how often control messages are sent, the size of the multicast group and its distribution several options can be applied to reduce the control traffic[9]:

- **Suppression:** For the case of a multicast tree with lots of receivers that are attached to a small number of nodes the acknowledgments would carry similar contents. To avoid transmitting several similar acknowledgments, each receiver schedules the transmission of its acknowledgment packet after some randomly chosen time period ($T : 0 < T < T_{\text{feedback}}$). If an acknowledgment form another receiver with similar content was seen during this period the receiver suppresses the transmission of its own report [10].
- **Unicast transmission:** For the case of multicast groups with few receivers distributed over a wide area using multicast for distributing the acknowledgments would only generate unnecessary overhead. A more efficient solution would be to unicast the acknowledgments form the receivers to the sender directly.

In addition to the end-to-end control messages used by DiffRes, the routers exchange hop-by-hop messages among each other. However, as each router only periodically transmits a message to its neighboring routers the overhead is small and does not change with an increased number of flows.

III. Protocol Messages

In this work, we will be mainly concentrating on the issues of signaling and maintaining reservation requests and not on the actual realization of service differentiation at the routers. Therefore, we will refer to the sum of requested resources simply as bandwidth and will not detail how the users should describe their QoS requirements. Depending on the wished granularity and scheduling schemes used at the routers, different implementations of DiffRes might use different representations for a QoS level such as mean bandwidth, maximum bandwidth, burst length [1] or simply the flow type[2]. In any case, any QoS description needs to be translated into bandwidth and buffer requirements at the routers. As in general, buffer costs are negligible compared to bandwidth costs [11] we will be only using the term bandwidth for describing the user's requirements in this study.

DiffRes control messages can be divided into three generic types:

- **Request messages:** Those are messages sent by the sender. The basic request message contains fields for describing the type of the message, a session identification (I_{d}), a timestamp, fields for describing the requested QoS, a timestamp and a field for collecting timing information from the network (T_{delay}). Request messages can be further subdivided into following types:
  1. **QUERY:** This message is used to collect information about the resource availability in the network as well as some timing information.
  2. **RESV:** The sender indicates with the RESV message the actual amount of resources he requires. In addition to the fields of the basic request message, RESV messages include a field for collecting the addresses of the routers traversed towards the receiver (L_{router}) as well as a field carrying the list of addresses traversed by a previously lost reservation ($L_{\mathrm{router\_old}}$). To simplify the processing of those variable length lists, RESV messages include also counters indicating the length of those lists (N_{router} and $N_{\mathrm{router\_old}}$).
  3. **DELETE:** Used to delete already established reservations and has the same format as RESV messages. In case of retransmitting a DELETE message in response to the loss of a previous one, $L_{\mathrm{router\_old}}$ indicates the list of routers where the session still maintains a reservation .
  4. **REFRESH:** To keep the end systems updated about the available resources and the path taken by the data packets, the sender needs to periodically issue request messages with the type field set to REFRESH. REFRESH messages have the same format as the basic request messages in addition to a L_{router} field for collecting path information and a field for indicating the amount of resources allocated for the session (B_{resv}).
- **Acknowledgment messages (ACK):** ACK messages are sent by the receivers in reply to received request messages. In addition to the fields of the basic request messages, ACK messages carry a field indicating the routing information collected by the request messages and a flag for indicating a change in path.
- **Router messages (RACK):** As will be described later, see Sec. IV, neighboring routers supporting the DiffRes protocol exchange periodic messages among each other. The RACK messages indicate the identities of the received or failed reservation or deletion requests in the periods between sending RACK messages as well some timing information.

IV. Message Sequences with DiffRes

In this section, we describe the interaction between the end systems and the network in order to establish, maintain and delete a reservation.

The basic approach is for the sender to transmit request messages to an edge device which forwards them into the network into another edge device and finally to the receiver. The receivers reply to incoming request messages by sending acknowledgment messages that contain the results of the request operation. As depicted in Fig. 1 the edge devices need to receive and process all control messages whereas the core routers need only to react to the reservation or deletion messages.

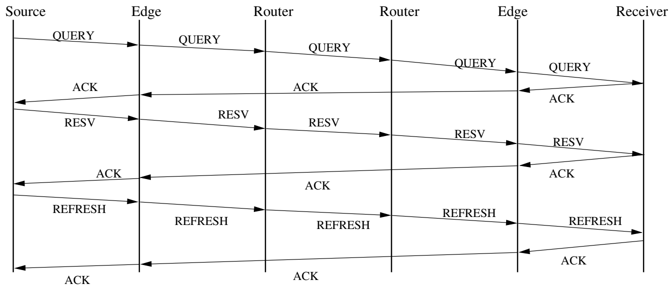

Fig. 1. End-to-end reservation message flow

A. The Query Phase

As depicted in Fig. 1 the sender starts the reservation procedure by issuing a QUERY packet. With the QUERY packet the sender indicates the QoS level it is requesting and the amount of resources to reserve.

Each traversed router controls the reservation requests indicated in the QUERY message and might change the bandwidth values depending on the requested service and its load situation. Additionally, each router adds the maximum time value (T_{rack}) it might take to send an acknowledgment packet for this request to the router from which this request was received, see Sec. VI.

When arriving at the receiver, the QUERY message indicates the current resource availability in the network which should aid the end systems in deciding on the amount of resources to request. While naturally, these values might change before the sender issues the actual reservation request, they should be used as an indication of the resource availability in the network.

B. The Reservation Phase

During the reservation phase the sender requests a specific amount of resources from the network. Note that we do not differentiate between new reservations or updates of an already established one. A sender can at any time increase the amount of allocated resources to him by issuing a new reservation request indicating the wished increase in resources. This is then treated as a new reservation at the core routers. Edge devices need, however, to update their state information accordingly.

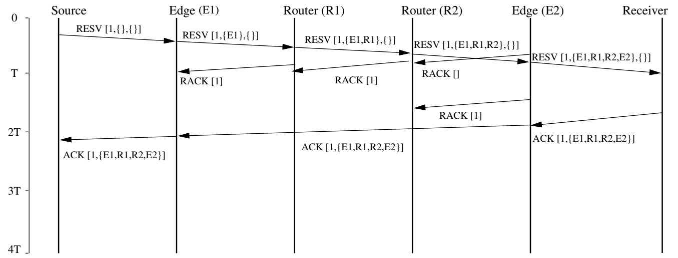

Fig. 2. Example of a successful reservation with T indicating the time between two RACK messages

• **Sender behavior:** After receiving an ACK message for its query request the sender starts the actual reservation procedure by issuing a reservation packet (RESV) indicating the requested QoS resources to reserve. The bandwidth value determined during the query phase should be taken as guidelines to the actual resource availability in the network. Additionally, the sender schedules a timer to expire after (T_s) seconds.

• **Sender-side edge device:** The RESV message is first sent to the edge device which checks if it has enough resources to admit the flow. In the case of acceptance it keeps state information for identifying the flow and the requested QoS parameters.

As already mentioned, service differentiation is realized by marking packets as belonging to different QoS level. This can either be achieved by marking the packets at the sender or at the edge device. In either cases, the edge device is responsible for ensuring the conformity of the entering flows to their reserved resources. This entails using policing or shaping mechanisms that drop or reduce the priority of packets that are sent in excess of the reserved resources.

By intercepting acknowledgment messages and noting the information about the resource availability in the network the edge device can avoid the case of *killer reservations*. This form of killer reservations indicates the case where a user requests a large amount of resources which gets granted in part of the network. As a result new reservations are rejected on that part of the network. However, the large reservation gets rejected at some other part of the network. This means that the resources reserved for that request are wasted and that the other requests were unnecessarily rejected. By rejecting requests that are not conformant with the query results the effects of this situation can be reduced considerably.

Further, the edge device schedules a timer to expire after T_{RE}, includes its address in the address list (L_{route}), increments N_{router} by one and forwards the message towards the next router on the path to the receiver.

• **Core router behavior:** Core routers temporarily maintain state information describing reservation requests. That is, after receiving a RESV message the router maintains a copy of the identification, QoS and routing information carried in the RESV message. These information, however, are only maintained for a period of maximally T_{Ri} seconds. If during this period of time, the router does not receive a RACK message acknowledging the reception of this request at the next router towards the receiver, the router considers this request as lost. This request is then added to the failed requests list in the next RACK message destined towards the router from which the request was originally received as indicated by the router list of this request. In case a RACK message indicating that this request was received by the next router arrived before T_{Ri} expires the state information is deleted. Before deleting the information, the router needs to make sure that it has acknowledged the reception of this request.

The routers maintain state information describing the total amount of reserved resources for each QoS level. In case the router can accommodate the new request, it needs to update the state information describing allocated resources for the requested QoS level. In case the requested amount can not be granted the router might reset the bandwidth field down to zero to indicate a reservation failure. Depending on the requested service, the router might also adjust the bandwidth field in accordance with its available resources.

• **Receiver edge device behavior:** Before arriving at the receiver, the control messages may pass another edge device that connects the DiffRes network to either another network or the final receiver. The edge device needs to maintain state information describing the flow's identity, QoS information as well as the routing information as described in the received RESV message.

• **Receiver behavior:** At the receiver site, the receiver acknowledges the reception of the RESV message by issuing an ACK message indicating the result of the reservation request.

B.1 The lossy case

In case either the acknowledgment packet (ACK) or the reservation packets (RESV) themselves are dropped the sender needs to determine the set of routers where it has already established a reservation in order to avoid duplicate reservations.

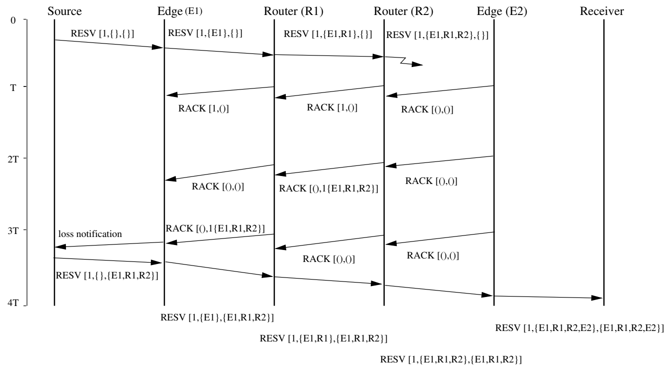

Fig. 3. Example of a failed reservation with T indicating the time between two RACK messages

• **Loss of RESV packets:** Due to congestion or link errors the reservation packets might be dropped at some core router (R_n). Each router (i) keeps the state information for a request for a time period of T_{Ri} before considering that request to be lost at a router further downstream towards the receiver. Fig. 3 describes the case of a lost reservation request. The reservation messages are depicted in the form of (RESV (flow id, L_{router}, $L_{\text{router\_old}}$)) and the RACK messages are depicted in the form of (RACK ((I_{d}) (I_{d} , L_{router}))) with the first field indicating the identities of new requests and the seconds indicating the identities and routing list of failed requests. The reservation request for the flow of identity 1, passes the edge device (E1) and the first router (R1) successfully and is dropped at router (R2). Each traversed router adds its address to the routing list (L_{router}) and maintains state information about this request for a period of time (T_{Ri}). Each T seconds the routers transmit a RACK message indicating the received and failed requests. In case no requests were received the request lists of the RACK message would be empty, e.g., the RACK messages of E2. In case a request was successfully received, the RACK messages indicate the identity of the flows in the request list field, e.g., the RACK messages of R1 and R2. Assume that the timer of R2 (T_{R2}) for the request of the flow of identity 1 expires prior to $2T$. Hence, R2 includes in its RACK messages the identity of the failed request and the contents of the routing list of that request as seen at R2. R1 forwards this information at time $3T$ to the edge device which sends a loss notification message to the sender indicating the identity of the lost request and addresses of the routers successfully traversed by that request.

In response to the failed request the sender issues a new reservation request RESV with the QoS information taken form the loss notification received from the edge device in response to the loss. While L_{router} and N_{router} are empty in the new request, $L_{\text{router\_old}}$ and $N_{\text{router\_old}}$ have the values of routing list and length indicated in the loss notification. Each traversed router checks if its own address was included in the $L_{\text{router\_old}}$ list attached to the RESV message. If that is the case, it can be assumed that a reservation was already made for this request and the router needs only to forward the request to the next router on the path towards the receiver. Routers that do not find their address in the list need to take the same procedure as for the lossless case, see Sec. IV-B.

In case the flow's path changed in between the transmission of the first reservation request and the retransmission of that request, the resources reserved on routers that are no longer traversed are lost. This situation needs to be accommodated in the calculation of the amount of reserved resources for the admission control procedure.

• **Loss of ACK packets:** If the sender receives no answer to its reservation request after a timeout period of T_{s} it issues a reservation check request (RCHECK) with L_{route} set to empty and the QoS information and timestamp as were used in the last RESV message. Each traversed router includes its address to the address list (L_{route}) and forwards the packet towards the receiver. In case the original request reached the receiver, the receiver would have maintained an entry for the request and the routing information for it. Similarity of the routing lists as well as the timestamps of the last successful reservation operation as observed at the receiver and the RCHECK message, indicates the success of the last reservation message and loss of the acknowledgment for it. In this case, the receiver issues an ACK message with the routing list included in the RCHECK message. The similarity of the timestamps and difference of the routing lists indicates a change in the path between the last reservation operation and the current checking operation. Hence, the receiver issues an ACK message with the path change flag set on and the routing list of the last successful reservation operation. In reaction to that, the sender issues a reservation request with $L_{\text{router\_old}}$ set to the routing list indicated in the ACK message. Only the traversed routers that are not included in the $L_{\text{router\_old}}$ list need to establish a new reservation for the session.

In case the receiver did not receive the RESV message in the first place, i.e., the timestamps of the of the last seen RESV message at the receiver and the timestamp included in the RCHECK message differ, the receiver sends an ACK message with the path change flag set off and an empty routing list. This indicates to the sender that it needs to reissue a new reservation request.

C. The Update Phase

To keep the end systems informed about the resource availability and path changes and avoid using hard state in the routers for estimating the amount of established reservations, the end systems need to periodically send REFRESH messages. Each traversed router includes its address in the routing list, possibly updates the bandwidth information and forwards the message towards the next router on the path to the receiver.

The receiver includes the bandwidth and routing information seen in the RESV message and issues an ACK message.

After receiving an ACK message with a routing list different from the list received in previous control messages, the sender issues a reservation request with $L_{\text{router\_old}}$ including the addresses that were common to its old list and the new one received in the ACK message. Only the new routers not included in $L_{\text{router\_old}}$ need to establish a new reservation.

D. The Deletion Phase

To release resources allocated to a flow, the sender should issue a DELETE message before ending the session. To ensure that resources are only deleted at routers for which this flow actually made a reservation, the sender includes the routing list of the flow in the $L_{\text{router\_old}}$. Each traversed router that finds its address in the $L_{\text{router\_old}}$ list reduces the value of the allocated resources by the amount indicated in the delete message and forwards the message to the next router towards the receiver. DELETE messages are treated at the routers similar to RESV messages. That is, they are acknowledged with the RACK messages. In case a DELETE message is lost, the sender receives a loss notification indicating up to which point the deletion operation was successful. In this case it retransmits another DELETE message. For the case of reservation messages, $L_{\text{router\_old}}$ indicates the list of routers where a reservation has already been established. For the case of deletion, $L_{\text{router\_old}}$ indicates the routers where the deletion operation still has not been processed. Hence, $L_{\text{router\_old}}$ indicates here the list of routers included in the path traversed by the flow and not included in the loss notification.

Using source-based routing [12] in conjunction with the delete operation can be of great value in reducing the waste of resources due to changes in paths. That is, after a change in path, the sender needs to make a new reservation on all newly traversed nodes and the resources reserved on the parts of the old path that are no longer traversed are wasted. Using source-based routing, the sender can send a delete packet with routing list (L_{router}) set to the old path and $L_{\text{router\_old}}$ including the addresses of the routers that are no longer traversed. Each router would then forward the DELETE packet not based on its current routing table but using the routing information in L_{router}. Each router that finds its address in the $L_{\text{router\_old}}$ deletes the amount of resources indicated by the DELETE request.

E. Multicast Operation

In contrast to the unicast case, for the case of multicast, receivers can join a session much later than the sender. In this case, the new receiver waits for a REFRESH message and sends an ACK message with the path change flag set on and the routing information indicated in the REFRESH message.

After receiving the ACK message from the new receiver the sender issues a RESV message towards that receiver. To avoid duplicate reservations, the sender maintains a two dimensional overall list with the one axis having the address of all the routers traversed by the multicast packets and the other axis listing all receivers. Each entry of this list, indicates then whether receiver X is traversing router Y. Such a list can be composed by the routing entries of the acknowledgment messages.

When a new receiver joins the session it waits until receiving a REFRESH message and sends an ACK message with the QoS and routing information indicated in the REFRESH message. The sender checks the routing list towards that receiver as indicated in the ACK message and determines which routers are new to its overall list. A reservation is then only required for those routers. That is, the sender issues a RESV message with $L_{\text{router\_old}}$ listing the routers on the path towards this receiver already included in the paths to other receivers.

In case the sender does not get an ACK message from some receiver over a period of time, the receiver is considered as having left the session and the sender deletes the resources reserved for this receiver.

While maintaining the routing table describing all the receivers and traversed routers introduces an additional overload on the sender, such a table can also be used for the distribution of costs of the consumed resources among all receivers [13].

In a multicast communication scenario the paths connecting the receivers to the senders might vary in their capacity. To accommodate this heterogeneity the sender might enforce a homogeneous reservation for all receivers that mirrors the capacities of a large part of the receivers or just the worst one. Another option might be to use layered data transmission [14]. That is, the user divides its data into n different layers. Based on the resource availability on the path from the sender to the receiver, the receiver can join up to n layers. The number of layers and resources allocated to the different layers can then be adjusted based on the heterogeneity of the receivers and network [15].

V. Admission Control

In the DiffRes architecture, core routers need only to maintain information about the sum of the already reserved resources for all the flows. While maintaining the exact sum of the reserved resources could be achieved using the information in the RESV and DELETE messages, relying on explicit deletion messages might lead to inconsistency problems. Some problems might arise due to loss of DELETE messages, node failures, path changes or incomplete DiffRes implementations that might not generate DELETE messages. Therefore, the REFRESH messages are used for estimating the amount of reserved resources. The senders transmit a REFRESH message every T_o seconds indicating the amount of reserved resources for their flows (B_{resv_i}). Hence, the sum of all values of B_{resv_i} of all flows in an interval of $(t, t+T_o]$ would represent the total amount of reserved resources (R_{resv}) during this time period.

$$R_{\text{resv}} = \sum_i B_{\text{resv}_i} \tag{1}$$

However, as Fig. 4 shows, due to the jitter in the delays the packets might face, a router might not receive REFRESH packets from all flows in the interval of $(t, t+T_o]$. Consider in Fig. 4 the case of flow $F1$. The first REFRESH message of this flow is received just outside the observation interval of $(t, t+T_o]$. The second instance sent exactly T_o seconds later is received ($T_o + T_{\text{j}1} : T_{\text{j}1} < T_j$) seconds later and hence falls in the interval $(t+T_0, t+2T_o]$. To accommodate this situation, the routers need to take late arriving REFRESH messages into consideration as well. Setting the maximum possible delay jitter to (T_j), the routers need to determine R_{resv} during an interval of $(t, t+T_o+T_j)$.

R_{resv} determined by summing all B_{resv_i} during an interval of $(t, t+T_o+T_j)$ is actually an upper bound on the already reserved resources in the interval of $(t, t+T_o]$. As REFRESH messages are sent in periods of T_o, the routers need to start a new observation period every T_o as well. However, as the observation periods need to be (T_o+T_j) long in order to consider late arrivals, R_{resv} might include duplicate REFRESH messages from the same flow. As Fig. 4 shows, during the interval $(t+T_o, t+2T_o]$ the REFRESH message of $F1$ is counted twice. While this surely leads to underutilizing the routers it gives an upper limit on the reserved resources and prevents routers from admitting new flows unless there is definitely enough resources to accommodate their needs. Actually, with such an approach, the average utilization level (U) of a router will be around

$$U = 1 - \frac{T_j}{T_o} \tag{2}$$

In addition to the delay jitter problem, REFRESH messages might get dropped and hence the bandwidth share consumed by these flows is not accounted for in the determination of R_{resv}. An approach similar to that described in [16] can be utilized to reduce the effects of losses. The routers need to take the maximum seen R_{resv} during a window of n observation intervals with each interval (T_o+T_j) long. Further, the core routers should not allocate all of their available resources (R) but reject new flows for the case of ($R_{\text{resv}} > R \times \sigma : \sigma < 1$).

With such an approach, the core routers start every T_o seconds an observation interval of the length of (T_o+T_j). The routers keep a variable describing the maximum reservation value observed in the last window of n observation intervals ($R_{\text{resv}_{old}}$) and a variable describing the maximum count of reserved resources measured in any of the observation intervals in the current observation window ($R_{\text{resv}_{current}}$). After receiving new reservation or deletion requests the routers need to increase or decrease $R_{\text{resv}_{old}}$ and $R_{\text{resv}_{current}}$ by the requested amount.

Flows requesting B resources are then only admitted for the case of

$$\max(R_{\text{resv}_{old}}, R_{\text{resv}_{current}}) + B < R \times \sigma \tag{3}$$

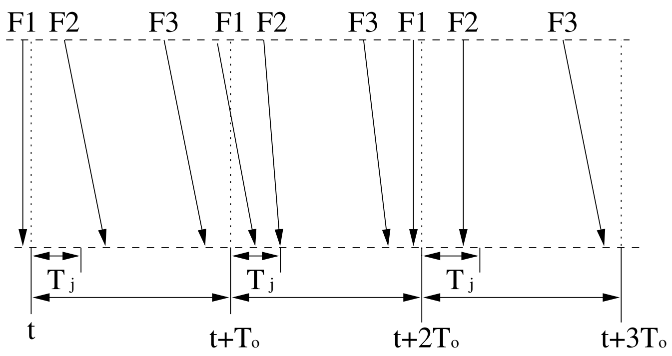

Fig. 4. Scenario for determining the amount of reserved resources

VI. Timeout Settings in DiffRes

As described in Sec. IV the sender and network routers use timeouts for determining the loss of control packets. That is, the sender waits T_s seconds before concluding that either the reservation request or the acknowledgment for it was lost. The edge device and core routers consider a reservation packet to be lost if it was not acknowledged by the next router downstream after T_{R_i} seconds.

When setting the values for the different timers one needs to make sure that the sender's (T_{s}) timer does not expire before any of the routers' timers (T_{R_i}) or while a loss notification is being sent towards the sender.

During the query and refresh operations, each traversed router (i) adds to the T_{delay} field of the request messages a value of T_{rack_i}. T_{rack_i} indicates the maximum time that can pass at router (i) between receiving a RESV or DELETE message and receiving an acknowledgment for it from its neighboring router. This delay (T_{rack_i}) consists of the buffering delays at this router and the neighboring one, the round trip propagation delay between the two routers as well the time between sending two RACK messages at the routers. The buffering delays can be assumed to be known at the routers. Through the exchange of the RACK messages, routers can determine a worst case estimation of T_{rack}. After traversing all routers towards the receiver, T_{delay} indicates the worst case delay between losing a request packet in the network and notifying the sender of the loss.

The sender sets its retransmission timeout and the value of the T_{delay} field in the RESV and DELETE messages to this value as well. Each traversed router i sets its timer for indicating the validity of the reservation or deletion request to ($T_{\text{R}_i} = T_{\text{delete}} - T_{\text{rack}_i}$) and sets the value of T_{delete} in the request message to T_{R_i}.

VII. Implementation Issues

To alert routers to more closely examine the contents of the request messages, the DiffRes messages can be carried in IP packets with the IP router alert option [17] similar to RSVP.

To differentiate between data and control traffic it suffices for the case of unicast to use a different port number. For the case of multicast, receivers should not be allowed to receive the multicasted data flow before establishing a reservation for it. Otherwise, packets marked with a specific QoS level are carried towards the receiver without having allocated the needed resources for them. This could lead to a degradation of the QoS offered by this level and lower levels. Hence, as the receivers need to wait for a REFRESH packet before sending an acknowledgment, the multicasted control data should be completely isolated from the data traffic it is controlling. This could lead to problems in case the network establishes different multicast trees for the control and data traffic. As both multicast sessions have the same senders and receivers this situation is only expected to occur rarely. The probability, effects and solutions of such a situation need to be further investigated.

In determining the maximum time a negative acknowledgment might arrive at a receiver we assumed lossless inter-router communication. To realize such a situation, RACK messages need to be transmitted at the highest possible priority in the network. Another option would be for the routers to retransmit the received flow identities for n times in their RACK messages. In this case, the calculations of T_{rack} needs to take the delay introduced by the retransmission into account.

VIII. DiffRes in Heterogeneous Environments

The description of the operation of DiffRes in Sec. IV assumed a homogeneous environment with all senders, receivers and network routers using the DiffRes protocol. In a more realistic environment we need to subdivide the network into clouds connected with each other through gateways. Hereby, a cloud indicates a homogeneous portion of the network deploying a specific QoS architecture such as DiffRes, RSVP or a differentiated services bandwidth broker [3].

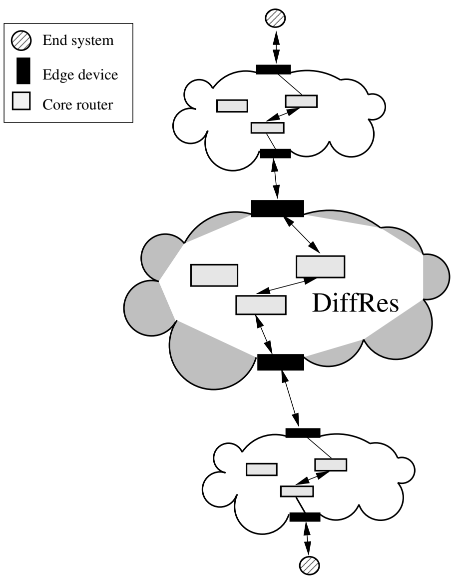

Fig. 5. End-To-End reservation example

For a simple interoperability scenario we consider the case of DiffRes cloud connecting two RSVP clouds. In this scenario, the RSVP messages are transported unaltered in the DiffRes cloud and are considered to be ordinary data packets. Additionally, the gateways need to assume the functionalities of the DiffRes senders and receivers. One possible mapping scenario would be:

1. When the ingress DiffRes edge device receives a PATH message from the RSVP cloud it initiates a query operation.
2. The egress edge device forwards the PATH message into the second RSVP cloud and delays the acknowledgment of the QUERY packet until it receives a reservation packet from the RSVP cloud.
3. The egress device generates the ACK message indicating the success or failure of the reservation operation in the RSVP cloud.

4. In case the reservation failed in the RSVP cloud, the ingress device forwards the failure notification to the first RSVP router directly.
5. Otherwise, the ingress device initiates a reservation operation in the DiffRes cloud, maps the result into an RSVP reservation message and forwards that message into the RSVP cloud.

Note also, that while it is essential that edge devices of a DiffRes cloud comply with the DiffRes specification, not all routers need to understand and process DiffRes messages. While QoS can not be guaranteed over paths traversing non-DiffRes routers, the overall performance of a network is still expected to improve, as the DiffRes routers will prevent overload situations in the parts of the network where they are deployed.

IX. Summary and Future Work

In this work, we presented a light weight signaling protocol for delivering and maintaining per-flow reservation requests without having to maintain per-flow state information in the network core routers.

With DiffRes we provide solutions to the issues of avoiding duplicate reservations due to loss of control messages, path changes and multicast communication.

The efficiency of DiffRes in terms of stability, scalability and bandwidth utilization, has to be investigated under simulative and experimental situations. Additionally, we need to further test the integration of DiffRes networks with other networks using RSVP for their QoS signaling.

Further, we still need to deeper investigate the issue of multicast reservations for heterogeneous environments.

References

[1] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," Tech. Rep. RFC 1633, Internet Engineering Task Force, June 1994.

[2] Ping P. Pan and Henning Schulzrinne, "YESSIR: A simple reservation mechanism for the Internet," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998, also IBM Research Technical Report TC20967.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," Request for Comments (Informational) 2475, Internet Engineering Task Force, Dec. 1998.

[4] Biswajit Nandy, Nabil Seddigh, and Peter Pieda, "Diffserv's assured forwarding PHB: what assurance does the customer have?," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, New Jersey, June 1999.

[5] Vern Paxon, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, California, Apr. 1997.

[6] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," Request for Comments (Proposed Standard) 2598, Internet Engineering Task Force, June 1999.

[7] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," Request for Comments (Proposed Standard) 2597, Internet Engineering Task Force, June 1999.

[8] Ion Stoica and Hui Zhang, "Providing guaranteed services without per flow management," in *Special Interest Group on Data Communication (SIGCOMM'99)*, Cambridge, USA, Aug. 1999.

[9] M Hoffman, J. Nonnenmacher, J. Rosenberg, and H. Schulzrinne, "A taxonomy of feedback for multicast," Internet Draft, Internet Engineering Task Force, June 1999, Work in progress.

[10] J. Nonnenmacher and Ernst W. Biersack, "Optimal multicast feedback," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, USA, Mar. 1998.

[11] Martin Karsten, Jens Schmitt, Lars Wolf, and Ralf Steinmetz, "Cost and price calculation for internet integrated services," in *11. Fachtagung ITG/VDI Fachtagung Kommunikation in Verteilten Systemen (KiVS'99)*, Darmstadt, Germany, 1999.

[12] Carl A. Sunshine, "Source routing in computer networks," *ACM Computer Communication Review*, vol. 7, no. 1, pp. 29–33, Jan. 1977.

[13] Shai Herzog, Scott Shenker, and Deborah Estrin, "Sharing the cost of multicast trees: An axiomatic analysis," in *SIGCOMM Symposium on Communications Architectures and Protocols*. ACM, Aug. 1995.

[14] Steven McCanne, Martin Vetterli, , and Van Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 6, Aug. 1997.

[15] Dorgham Sisalem and Frank Emanuel, "QoS control using adaptive layered data transmission," in *IEEE Multimedia Systems Conference'98*, Austin, TX, USA, June 1998.

[16] Sugih Jamin, *A measurement-based admission control algorithm for integrated services packet networks*, Ph.D. thesis, Dept. of Computer Science, University of Southern California, Los Angeles, California, Aug. 1996.

[17] D. Katz, "IP router alert option," Request for Comments (Proposed Standard) 2113, Internet Engineering Task Force, Feb. 1997.