StreamCom QoS Reservation Architecture: Design of Network Elements

*

Matthias Scheidegger

November 20, 2001

Abstract

This document describes the part of the StreamCom architecture concerned with quality of service provisioning and reservations. It details design problems of various related aspects, such as the issues arising when combining differentiated services, measurement based admission control and multicast, or the elimination of undesirable reservations for unauthorized receivers, in section 1. Section 2 gives an overview of the developped design and the involved entities and roles, whereas the rest of the document contains separate design descriptions for each of those entities.

1 General Considerations

1.1 The Choice of QoS Architecture

Many networking experts believe that the current best effort model of the Internet is not sufficiently suited for supporting new quality of service (QoS) sensitive applications like real time audio and video streaming, which require strict guarantees about the service the network provides. Best effort packets are not guaranteed to reach their destinations, nor to reach them in an exactly predictable amount of time. While traditional, mostly TCP based applications like mail or FTP work well under these conditions, an additional mechanism is needed to provide better packet delivery guarantees to the mostly UDP based real time applications. The two most prominent solutions to this problem are the so called integrated services [5] (with its main implementation RSVP [6]) and differentiated services [3] architectures (also called IntServ and DiffServ for short).

IntServ's basic idea is to 'pin' data flows to a path determined at the beginning of a transmission and then reserve bandwidth for the flow in every router along that path. Path pinning not only determines the involved routers, but also removes the possibility for packets to take arbitrary routes, a major source of variations in packet delivery time. Although experience shows that this approach is viable for small to medium sized networks, the fact that every router has to keep track of every passing flow causes severe scalability problems in large scale networks like the Internet. The signalling overhead aggravates the problem even more.

DiffServ aims to lessen these problems by moving complexity away from the backbone to the network edge — a traditional approach within the Internet community. Upon entering a backbone network, packets are given an identifier called the Differentiated Services Class Point (DSCP [11]) by the ingress router, assigning them to a certain forwarding behaviour in the backbone. The forwarding behaviour for a given DSCP is termed Per Hop Behaviour (PHB) and has to be the same in the whole DiffServ domain, which means that every router in the domain must treat any two packets with the same DSCP equally. That way, many flows can be grouped into a single behavioural aggregate. Core routers only have to distinguish between a small number of PHBs and are thus freed of the responsibility to store per-flow information. Figure 1 shows the effect of these behaviour aggregates (BAs) on routing complexity: While first-hop routers only have to know about the flows from their local network, ingress routers to backbone networks can simply distinguish between source networks.

*DRAFT VERSION

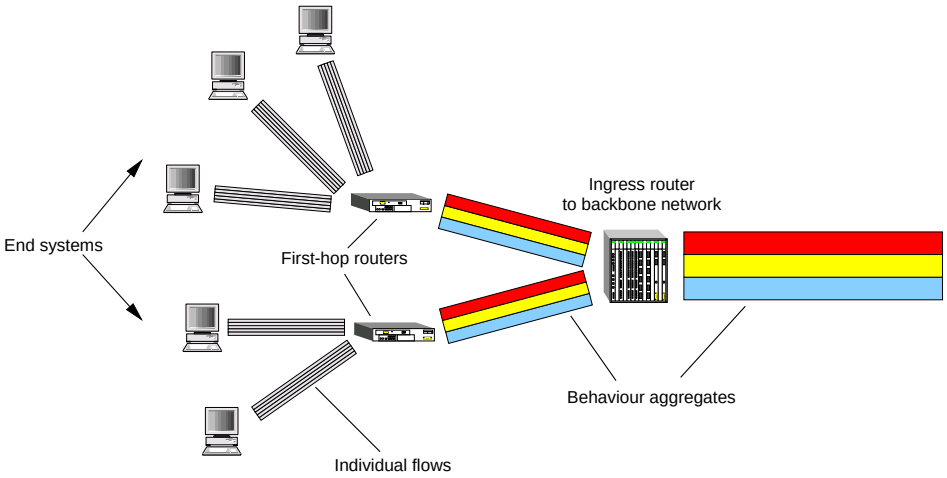


Figure 1: Flow aggregation using BAs

For the reasons stated above we decided to use DiffServ in the StreamCom architecture. The following sections focus on various issues in this context.

1.2 Admission Control

1.2.1 Bandwidth Brokers

In IntServ every router along a path can admit or reject a new reservation, depending on its available resources. A new flow is admitted if every router along its path admits it.
There is no standard way to do this *admission control* decision in basic DiffServ. Although a great part of the network resources can be provisioned statically — organizations usually determine their budgets for communications on a quarterly or annual basis and reserve bandwidth for similar periods — a smaller fraction of the available bandwidth should be open to dynamic allocation. RFC 2638 [12] introduces a possible solution that covers both reservation styles, using so called *bandwidth brokers* (BBs). Every BB controls one *trust region* (usually there is one trust region per domain). A BB's tasks are to configure the routers in its trust region and to negotiate dynamic allocations with members of its trust region or adjacent BBs. Agreements between trust regions are purely bilateral. Therefore, to allocate bandwidth for a flow spanning multiple trust regions, requests have to be forwarded by each BB along the path to its next-hop successor.

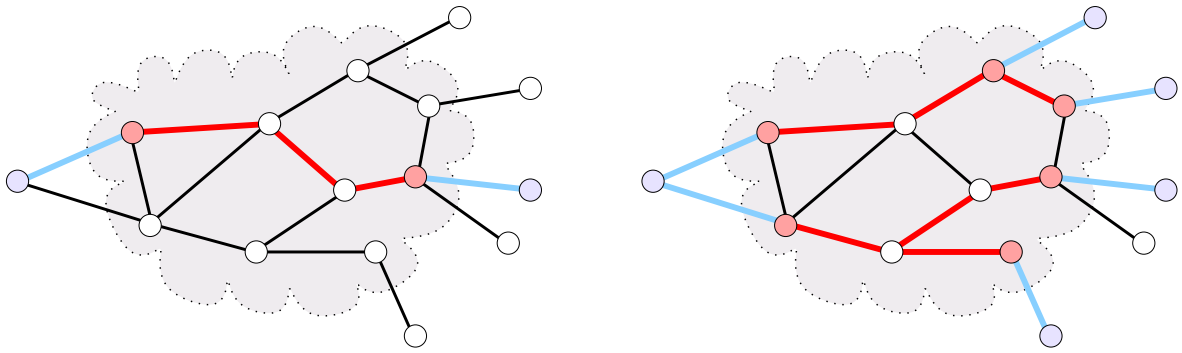
1.2.2 Measurement Based Admission Control

In recent scientific work, a new approach has been given increasing attention. It is usually called *endpoint admission control* (EAC) but other names have been used for the various variants. Two examples are Distributed Admission Control [9] and Egress Admission Control [8].
The basic idea of endpoint admission control conforms to the philosophy of the Internet to push complexity to the network edge. Instead of having central instances like bandwidth brokers, endpoints send a stream of probe packets and decide depending on the observed behaviour of these packets whether to admit a flow. Details of the network structure, like queueing disciplines and MPLS domains remain transparent to the endpoints, adding greatly to the simplicity of the approach.
In this document we don't consider architectural issues of EAC but rather examine the idea of *measurement based admission control* (MBAC), which doesn't imply endpoint control and can be integrated into other architectures, e.g. the bandwidth broker architecture.

Breslau et al. [7] introduced a classification for these measuring methods using two criteria. First, probe packets can be sent in the same traffic class as the already established flows or they can be assigned to a special distinct traffic class, which is called in-band and out-of-band measuring, respectively. While introducing more complexity to the core network, the latter method can be said to perform generally better. It requires significantly less time to attain the same level of accuracy as the in-band method.
Second, when a probe packet exceeds the admissible region, it can either be dropped or its ECN[1] bit can be set. Again, the latter approach yields better results at the cost of additional complexity inside the network.

1.2.3 DiffServ Reservations for IP Multicast

Reservations in the differentiated services architecture can be used for unicast and multicast flows alike. However, the effect of both kinds of flows on backbone resources differ significantly. Unicast flows normally only affect a single path inside the domain with fixed ingress and egress routers, whereas multicast flows can have multiple ingress and even more egress routers (Figure 2). Furthermore, the structure of the multicast tree is rarely known in advance .



(a) Unicast flow

(b) Multicast flow

Figure 2: Unicast and multicast flows crossing a domain

A simple way to reserve bandwidth for multicast flows would be to presume broadcast behaviour and therefore allocate bandwidth on every link inside the domain. That obviously leads to a big waste of bandwidth when the number of actually affected links is small.
Alternatively, bandwidth reservation could be restricted to the affected links, eliminating bandwidth waste. To do this we have to learn the exact structure of the multicast tree — or at least the part inside our domain. The way to do this depends on the scenario — in our case the StreamCom scenario, which is discussed further below.
The dynamic nature of IP multicast also causes a problem called the *Neglected Reservation Subtree Problem (NRS problem)* [4]. Since multicast trees can grow and shrink dynamically, additional subtrees without any QoS reservations can be created, leading to one of two undesirable situations. The first occurs when the *connecting node* of the subtree is an egress router: Replicated packets carry the same DSCP as the original one, even the ones that go to the unallocated subtree. Due to the missing reservation in the ingress node of the next DiffServ domain, packets will be dropped. Unfortunately, that also affects packets of correctly reserved flows because the ingress node only knows about one aggregate flow.
The second situation occurs when the connecting node is an interior router. Multicast packets flowing down the new subtree will not be dropped in that case, but they will steal bandwidth from lower class traffic. Figure 3 shows both cases graphically.

[1] ECN - Explicit Congestion Notification [13]

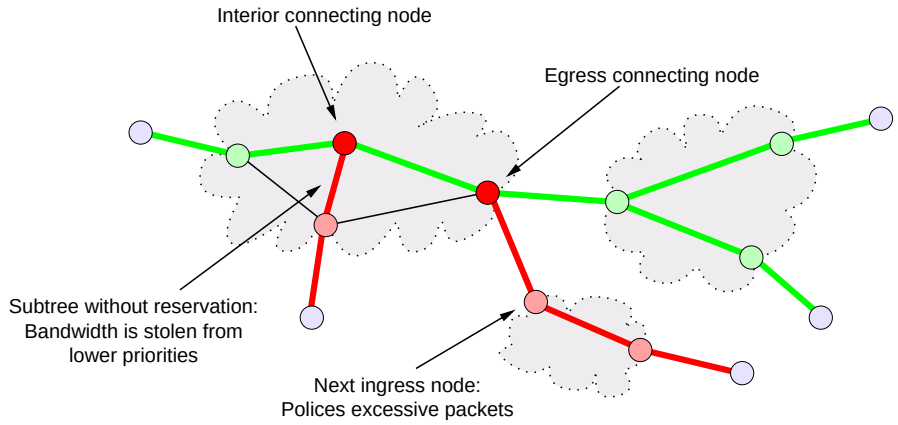


Figure 3: The Neglected Reservation Subtree Problem

The solution presented in [4] is the following:
A DSCP field per entry is added to the multicast routing table in every router. When replicating packets, this value must be used instead of the original DSCP from the incoming packet. In the case of best effort traffic nothing happens; the DSCP value remains the same. In the case of prioritized traffic however, the DSCP is preserved only for reserved subtrees. Packets going out in any other direction receive the DSCP value of a worse than best effort service class. Thus, the service provided to existing flows is not impaired, and the remaining multicast packets suffice to support multicast tree management functionality. This approach even allows for heterogenous multicast groups (i.e. groups where different subtress receive different service levels).

For the above solution to be possible we need a centralized control entity, possibly integrated in a bandwidth broker, keeping track of active multicast sessions with QoS reservations and configuring routers accordingly. Such an entity necessitates awareness of the multicast tree's current structure, which can be gained through a broker oriented and a router oriented approach: One is to have the bandwidth broker gather information from every router inside its trust region on a regular basis. While this can result in significant overhead as an isolated method, it adds little complexity if the broker already gathers unicast routing tables in a similar fashion.
The other solution requires the multicast routers to send a join notification to the broker whenever a new subtree is attached to them, a task usually performed by the routing daemon. That way, the broker needs to update its topology information only when there is a structural change. However, the reduced overhead comes at the cost of additional complexity in the core routers.

The architecture in this document uses a simplified version of the approach introduced above. Bandwidth brokers control the ISP networks and their QoS reservations, and the multicast topology information inside the brokers gets updated using the router based approach. However, the concept of a lower than best effort service class was left out for reasons of simplicity and flexibility. Packets are retagged to best effort instead.

1.2.4 Admission Control for IP Multicast

The business model used in StreamCom is based on the assumption that most clients buy tickets in advance to receive a later transmission. Also, a smaller number of clients should be able to join dynamically during the transmission. This splits the problem of admission control into two parts: scheduled reservations for a group of known receivers and dynamic reservations for any clients wanting to join during the transmission. The following sections focus on those problems in the above order.

Scheduled Reservations

In order to be able to reserve bandwidth for a multicast tree in advance we need the following information:

1. Characteristics of the video transmission, like scheduled time, duration and peak bandwidth.
2. The source address of the transmission, i.e. of the Content Server.
3. A list of all receiver addresses known in advance.

Using that information we can identify the involved ISP networks. All of these ISPs must then determine their ingress and egress nodes that will be part of the multicast tree. Where possible, the topology of the tree inside an ISP network should also be calculated in advance. All this allows us to reserve bandwidth on all involved networks for the scheduled time and duration. Note that the reservation can only be called successful if all ISPs have sent a confirmation message.

There is one restriction to the QoS architecture required to be able to provide ex-ante reservations: All ad-hoc reservations must be confined to a certain finite time interval. Absence of this requirement can lead to a situation where a previously scheduled reservation can't be granted because of unexpected ad-hoc reservations occupying the available bandwidth resources.
Assume this requirement as granted. Then, if the maximum duration of a ad-hoc reservation is Δt and the scheduled start of the transmission is T, the ISP must not allow ad-hoc reservations inside the video transmission band starting at $T - \Delta t$, thus letting any ad-hoc reservations time out before the scheduled transmission starts (see Figure 4). Obviously, QoS reservations can only be done efficiently in advance if there is an entity controlling the reservations on a network. Thus, the requirements of the StreamCom project can be best fulfilled using a bandwidth broker architecture.

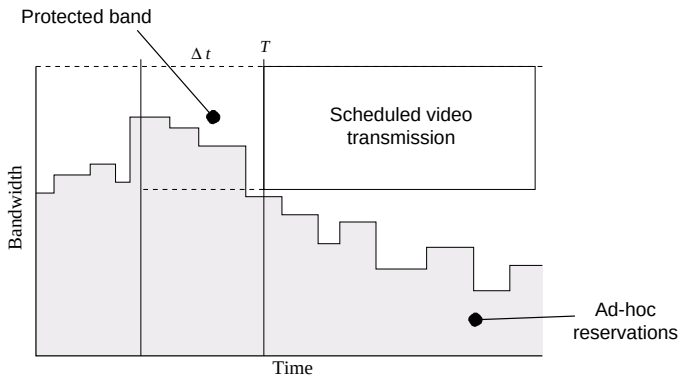


Figure 4: Bandwidth protection for scheduled reservations

Dynamic Reservations

In addition to scheduling QoS reservations in advance, the StreamCom QoS architecture should be able to reserve bandwidth for clients joining dynamically. Obviously, this has to be done using ad-hoc reservations, which raises the question of how to perform admission control in such a case, in particular since the additional reservation only affects a new branch of the multicast tree, not the whole route from source to destination.
The first question arising is how to identify the additional branch in the multicast tree, restricted to the network of concern. Luckily, the approach from [4] described above gives us just that. When a new client joins the multicast group, the connecting router and each router of the new branch will inform the bandwidth broker about the event. Using that information the bandwidth broker can calculate the exact path of the branch between the branching and the egress node where the branch leaves the ISP network.
The next question is: How should admission control be done? The StreamCom project specification [10] requires the approach to be measurement based. Various approaches to measurement based admission control have been proposed, usually in the form of a whole architecture, but unfortunately multicasting has only been unsufficiently considered in these designs. However, the measuring methods used in these works can often be applied or adapted to multicasting.

An important difference between unicast and multicast sessions is the ability of multicast sessions to grow and shrink. When the tree grows, there is only one new branch — without any branchings — originating from

a connecting router and ending at the joining client, possibly going through multiple DiffServ domains. In such a case, admission control can be done for all domains at the same time if there is a statical reservation for the aggregate of all flows using measurement based admission control. Since the connecting router may be an interior node, the architecture must support admission control between arbitrary source nodes inside the domain and the last-hop router. Measuring *agents* deployed on every DiffServ node can provide this functionality. The agents on the connecting node and on the last-hop node measure the available service between them. They are called *upstream* and *downstream agent*, respectively. Figure 5 shows the scenario.

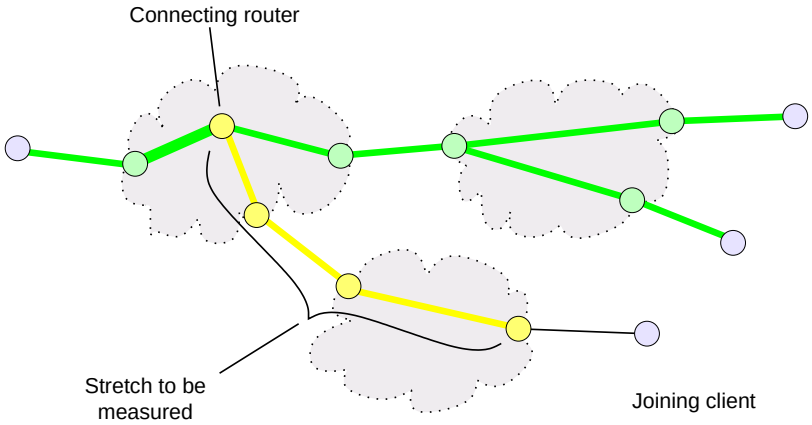
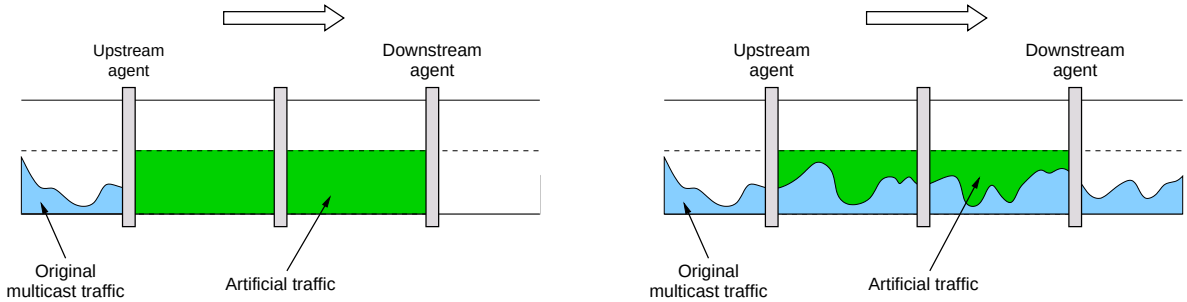


Figure 5: Admission control for additional branches

Since the measurement flow ends on the last-hop, the effects caused by that node can't be measured. The errors resulting from the missing hop must therefore be estimated and corrected by the downstream agent.

Furthermore, due to the architecture chosen in section 1.2.3, the session's multicast traffic will already be flowing down the new branch, marked as best effort traffic by the connecting router. This interferes with the usual measurement methods, which are based on measuring *before* the actual traffic begins to flow. A straightforward approach would be to block the multicast stream at the connecting router for the duration of the measurement (Figure 6(a)). The way clients authenticate themselves in the StreamCom architecture — discussed later in this document — depends on the original video data, thus making that approach unsuitable, however.

The approach used in the StreamCom architecture uses a more sophisticated technique: The upstream agent permanently measures the current bandwidth of the video stream and and generates supplemental probe packets to obtain a flow with a constant bandwith equal to the expected peak bandwidth. The downstream agent then measures the behaviour of this aggregate flow and absorbs the supplemental packets (Figure 6(b)).



(a) Blocking

(b) Supplementing

Figure 6: Two approaches to straighten out the burstiness of video traffic

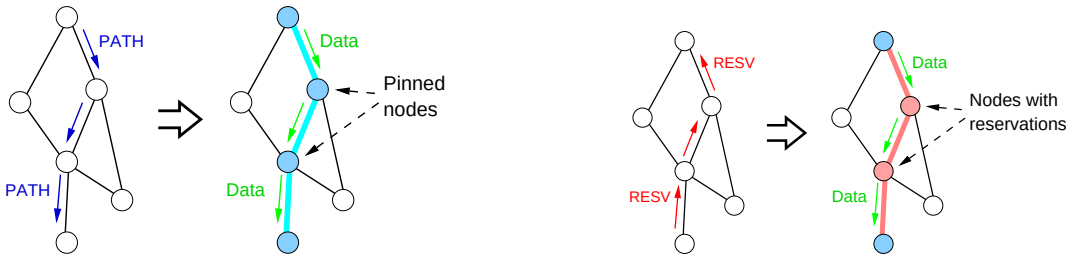
1.2.5 Authorization for QoS Reservations

An important requirement for the StreamCom QoS architecture is the ability to distinguish between authorized and unauthorized clients and to provide QoS reservation only to the former. Thus, a mechanism to deliver and validate user authentifications is needed. This document only focuses on the delivery architecture on the QoS reservation side and its interfaces to the client and to the validation part described in [2].

ISPs reserve bandwidth independently. Therefore, authentication information must be distributed to each ISP network involved in the video transmission. Further, every subtree, and therefore every single branch, leading to an authorized client has to be authenticated. The signalling part of RSVP is well suited for this task, especially when using the per-domain approach described in [1] to make it scalable. For that reason the authentication part of the StreamCom architecture uses RSVP to distribute authentication information along the multicast tree. The following part of this document presents the related properties of RSVP and the resulting issues to be solved.

Properties of RSVP

RSVP 'pins' a path to a route by sending Path messages downstream along the route. Every RSVP node along that path receiving such a message stores the address of the previous hop and then forwards it along the route[2]. After that, any packets belonging to the flow in question must be forwarded to the same next-hop node as the Path message (Figure 7(a)).



(a) Path phase

(b) Resv phase

Figure 7: The two phases of RSVP signalling

Once the path has been set up, the receiving endpoint sends a Resv message in the opposite direction, which the RSVP nodes forward along the previously pinned path. Thus, the Resv message reaches every RSVP router on the path. Combined with the ability of RSVP Resv messages to carry additional data objects, this yields a way for clients to distribute their authenticating information upstream to the nodes of concern (Figure 7(b)).
Applying it to a multicast scenario requires some additional functionality. Although only one reservation is needed for every branch of the tree, all the Resv messages of the leaf nodes converge towards the tree root. RSVP's solution for that is the ability of RSVP nodes to merge related Resv messages into one.

Integration into StreamCom

The RSVP part of StreamCom uses an approach similar to [1] by viewing a whole DiffServ domain as a single RSVP node. That can be done by building RSVP only into the edge routers and leaving core routers without any RSVP functionality. In our case however, Resv messages must be processed when they enter the network (i.e. by the flow's egress router), as opposed to when they leave it, like in the original proposal. This is due to the fact that in the multicast case the egress node isn't known to the ingress node, in contrast to the unicast case, where it could be derived from the Path messages processed earlier. Since reservations are done per domain, validation

[2]Note that there may be multiple non-RSVP hops between two RSVP nodes

of authentication information contained in the Resv messages should also be possible on a per domain basis. The StreamCom architecture complements the domain's bandwidth broker with a validation facility called *policy server*. Egress nodes send the information extracted from Resv messages through the broker to that server and receive either a positive or a negative validation report. Dependening on that report they can then either forward the Resv message upstream or send an ResvErr message back downstream [3].

As we have seen above, the number of Resv messages near the tree root can be reduced by merging these messages in intermediate nodes. However, messages carrying invalid authentication information mustn't be forwarded or merged with other messages. Since this requires knowledge about the validity of messages, merging is done in the egress routers.

2 Overview

In the remaining part of the document we describe the architecture in detail, starting with a presentation of the entities and their roles in the different scenarios from session setup to termination. Further sections specify the single entities' designs, algorithms and states.

2.1 Entities

The StreamCom architecture includes several different entities. Among those involved in the QoS reservation part the **bandwidth broker** is the central one. The bandwidth broker implementation from the University of Bern [14] being the starting point, we altered and added functionality of the original design to fit the requirements. This new functionality consist of support for multicast topology management, measurement based admission control and client/subtree authentication using the policy server. Further details can be found in section 3.
Authentification is done by the **policy server**, which is described in [2]. The bandwidth broker communicates with it to validate authentication information supplied to it by the egress routers.

The **content server** [16] is the source of the various multicast groups of a session. In order to make RSVP signalling possible, it needs to send Path messages to these groups on a regular basis.

Routers can represent one of multiple specialized roles, but all of them have a common set of responsibilities, which is that of a **Core router**: They must support differentiated services and multicast routing, and they must include an **MBAC agent**. Again, edge routers may take different roles, **ingress router** or **egress router**. Both must support RSVP signalling and forward RSVP messages like Path and Resv along the multicast tree. Authentication information contained in Resv messages must only be processed if the message comes from outside the domain, which is only the case in the multicast tree's egress routers. In all other cases they should simply be forwarded.

Finally, **Clients** have must have a basic RSVP implementation to send their authentication keys using Resv messages, which requires prior reception of Path messages. This functionality also has to be included in the reception software described in [16], of course.

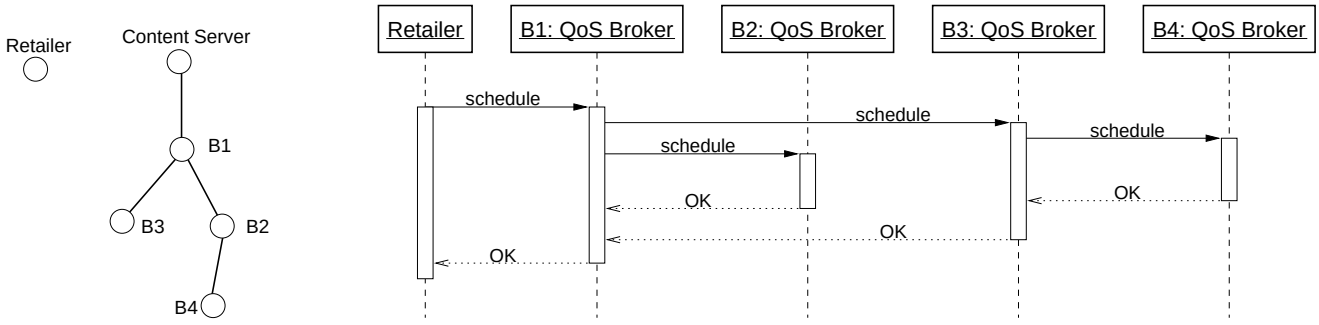
2.2 Scenarios

The following scenarios show typical situations of the StreamCom QoS architecture in action. They should cover all important cases and should suffice to gain a good understanding of the system. Technical details can be found in the subsequent sections.

[3] Since the message gets sent to the multicast address, every member of the subtree will get a copy. The enclosed copy of the original reservation object must be used by clients to determine if the message was directed to them

2.2.1 Reservation Scheduling

Before the subscription period of the retailer component begins, it contacts the 'nearest' QoS broker to schedule a bandwidth reservation for the broadcasting period. Enclosed in the request are time, duration, service level requirements for every multicast group, the content server's IP address, the multicast addresses used and a list of anticipated receiver's IP addresses covering the planned area of the transmission. After receiving such a request the broker decides whether it can grant the reservation or not. If it can't, it sends back a negative reply. Otherwise, it determines the downstream DiffServ domains and forwards the request to their respective QoS brokers. The reply is positive only if each of these brokers replied positively also. The procedure continues recursively down through the anticipated multicast tree. Figure 8 shows an example for a simple case.



(a) Topology

(b) Sequence Diagram

Figure 8: Reservation scheduling procedure

2.2.2 Session Setup

Setting up the multicast session is done in two phases. In **Phase A**, the QoS brokers activate the reservations scheduled beforehand and the content server begins to send to the multicast groups. During the setup phase of the multicast trees the routers send `JOIN_INDICATION` messages to their respective QoS brokers, who in turn reply with `SET_CODEPOINT` commands, carrying the DiffServ codepoint specified in the scheduled reservations.
In **Phase B**, RSVP Path messages, generated by the content server, begin to flow down the multicast trees along with the video data, thus enabling the clients to send their authentication keys by replying with Resv messages. When a Resv message reaches an egress router, it extracts the authentication object and sends it to its QoS broker as part of an `auth` request, which further indicates the domain the Resv message came from. The QoS broker forwards the keys to the policy server for authentication and informs the egress router of the outcome. If the authentication was successful, the router merges the Resv message with any other successful reservation messages it received and forwards it upstream. Otherwise, it sends a ResvErr message back to the client.
Meanwhile, the broker updates its topology database, depending on the outcome of the previous authentication procedure. If any of the Resv messages carried illegal authentication keys, the resulting unauthorized subtrees must be identified and the connecting routers' multicast routing table must be updated using `SET_CODEPOINT` messages. Section 3 describes the algorithm used to identify those subtrees.
A special case occurs if an egress router didn't receive a Resv message from a subtree for a certain time. In that case, a timeout occurs in the QoS broker, which has the same effect as an unsuccessful `auth` request. The timeout clock for a subtree is restarted after every successful Resv message. Therefore, in order to keep the reservation running, a client must send Resv messages in regular intervals.
The reservations for the detected multicast tree excluding any detected unauthorized subtrees, along with the larger scheduled reservation used in the setup procedure, serves as the basis to calculate the effectively charged costs. Clients joining or leaving later further increase or decrease these costs.

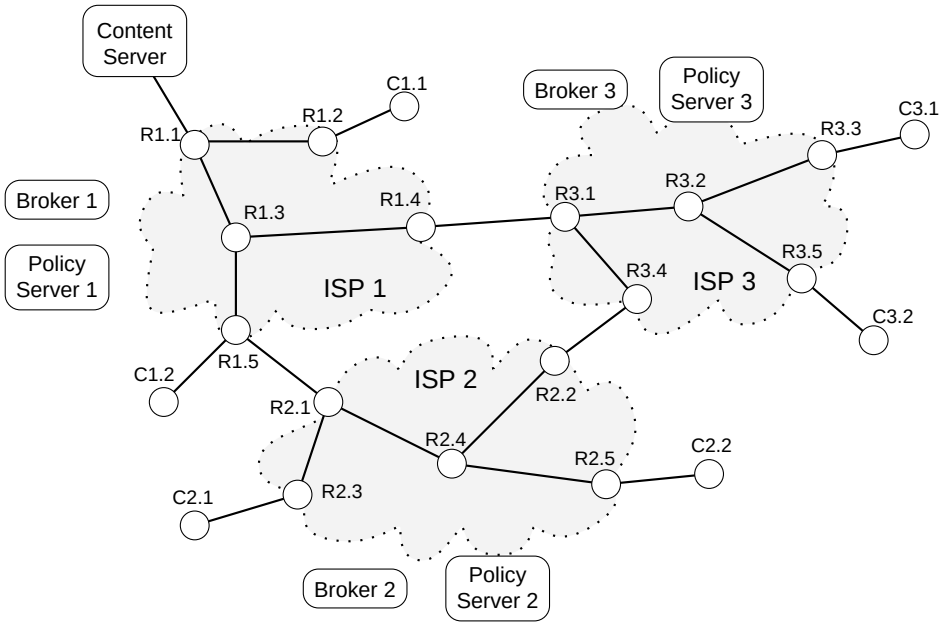


Figure 9: Situation before session setup

The following example scenario is based on the situation shown in Figure 9. Nodes with names of the form $Rd.n$ are routers and nodes with names of the form $Cd.n$ are clients, where d designates the domain and n designates the entity number. Further, the scenario only uses one multicast group for simplicity.

To examine the events in *Phase A* we now focus on ISP A. At the scheduled time the QoS broker sends reservation commands (`add_flow`) to all routers in its domain and the content server begins to send the video stream to the multicast group. Clients can now join the group, causing routers to generate `JOIN_INDICATION` messages. In our example — from the perspective of ISP 1 — the client $C1.2$ and the ISPs 2 and 3 join the multicast group[4]. Consequently, all routers except $R1.2$ send a `JOIN_INDICATION` message to Broker 1, which in turn sends `SET_CODEPOINT` messages to each router (see Figure 10), concluding Phase A of the session setup procedure. When the clients have joined the multicast group and have received at least one Path message from the content server, they enter *Phase B* by sending a Resv message up the multicast tree. Figure 11 shows the multicast tree after Phase A and the flow of the Resv messages. Communications with the QoS broker and policy server have been merged to single two-way arrows to reduce complexity. As described above the egress routers authenticate the Resv messages and merge them when appropriate. Two special cases are included in the diagram: $C2.2$ sends an illegal authentication request and receives a ResvErr message, whereas $C3.1$ doesn't send any Resv message at all, causing a timeout in Broker 3. Finally, Figure 12 shows the reaction of the brokers and the state of the system when the session is set up. Since $R3.2$ and $R2.1$ are the connecting routers to the newly detected unauthorized subtrees, the brokers send appropriate `SET_CODEPOINT` messages to them in order to cancel reservations. With the resulting topology session setup comes to an end.

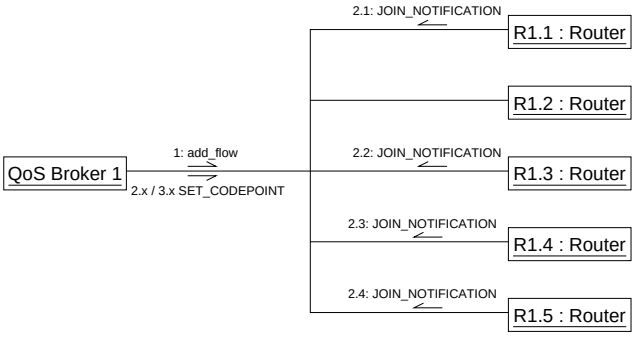


Figure 10: Initial topology updates

[4]The ISPs themselves don't join the group, of course, but rather clients connected to them. However, this is transparent to ISP 1.

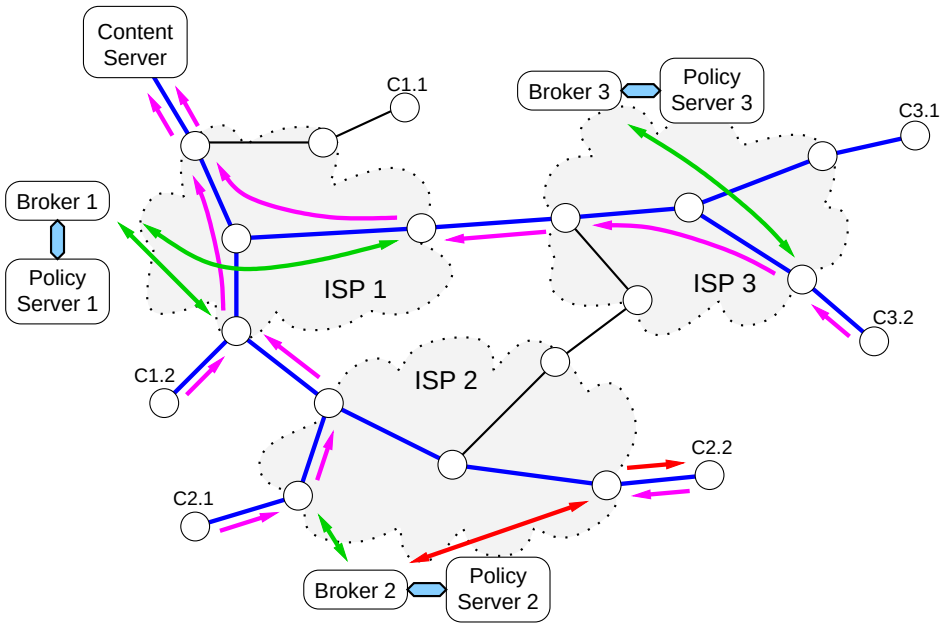


Figure 11: Flow of Resv messages

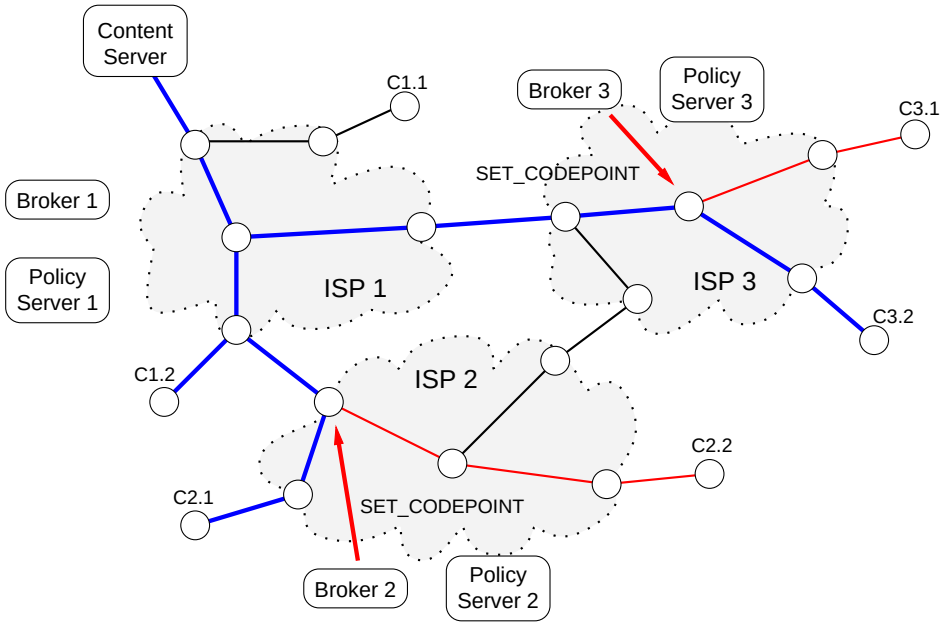
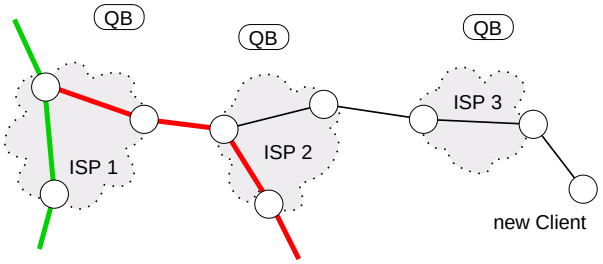


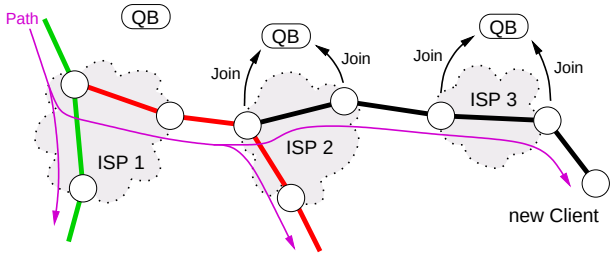
Figure 12: Pruning of unauthorized subtrees

2.2.3 Dynamically Joining Client

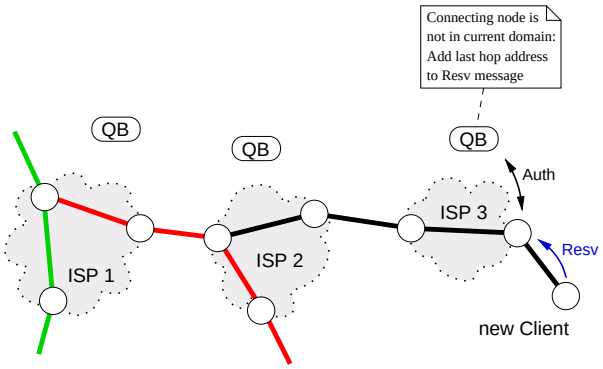
Once the video broadcast has been started there is the possibility of additional clients wanting to join the multicast group dynamically. However, since the reservations have been reduced to include only the actual set of participating clients, the statically configured reservations controlled by measurement based admission control must be used, as described on page 5.
However, due to the many possible states of the system, the dynamic joining procedure is rather complex. Figure 13, which will now be discussed, shows that procedure in a typical, successful case. Where appropriate, we mention differences between the example and other cases.



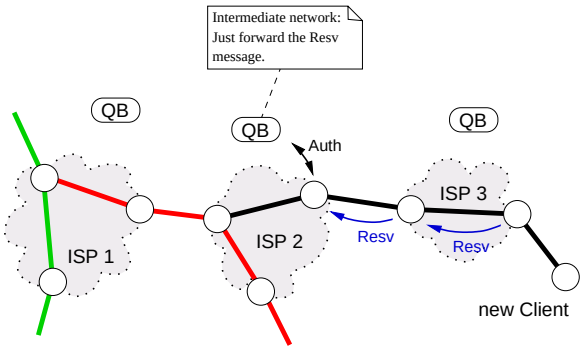
(a) Initial situation



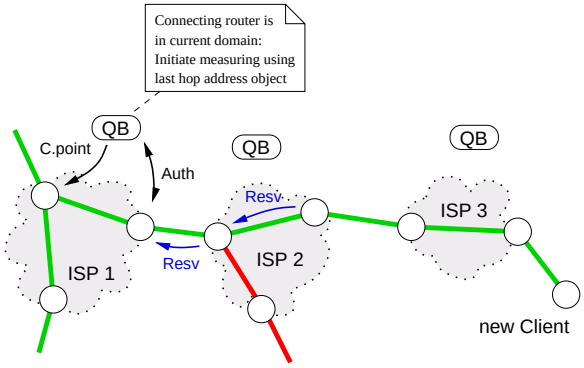
(b) Client joins the multicast tree



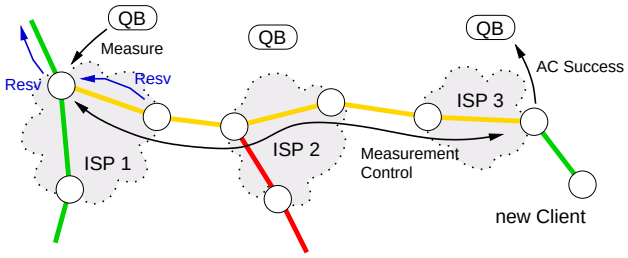
(c) Client sends Resv message



(d) Propagation of Resv message



(e) Resv reaches connecting domain



(f) Measurement to control admission

Figure 13: Dynamic reservation setup

The initial situation (Figure 13(a)) shows part of the multicast tree after the session setup procedure has completed. Thick connection lines stand for links which are part of the multicast tree. Green, red and black ones signify links which have been correctly authorized, failed to be authorized, or have not been decided upon at all, respectively. The entities labelled QB are conglomerates of a QoS broker and a policy server, one for every domain.
When the new client joins it causes all multicast routers who haven't previously been part of the tree to send join notifications to their brokers (Figure 13(b)). The brokers in turn update their internal topology databases. Now video traffic an Path messages begin to flow towards the new client.
After the client has joined the multicast group and begins to receive Path messages, it sends a Resv message upstream (Figure 13(c)). The last-hop router (as seen from the tree's perspective) intercepts the message and tries to authenticate the client. If that fails, it sends back a ResvErr message and the procedure ends. Otherwise, determines if the connecting point of the unreserved subtree is inside its own domain. Since that is not the case it tells the last-hop router to add its IP address as an RSVP object to the Resv message and send it on.
When the message reaches the next domain, a similar procedure takes place (Figure 13(d)). In contrast, the Resv message simply gets forwarded upstream, since neither the connecting router nor the last-hop router are part of the domain.
In Figure 13(e) the Resv messages reaches the ISP domain containing the unreserved subtree's connecting router. After successfully verifying the authentication information, the broker sets the new class point for the subtree on the connecting router and initiates admission control (Figure 13(f)). It sends a message to the agent on the connecting router, telling it to start a measuring session with the last-hop router — which in turn is known because of the additional RSVP object in the Resv message. At the same time, the original Resv message (without the special address object) is sent further up the multicast tree, which usually results in a merging operation in the next domain. The notified agent takes the role of upstream agent and opens a control connection to the last-hop router. After a brief exchange of parameters, an in-band measurement is performed using the technique described on page 6. When the measurement is done, the downstream agent decides whether the results are acceptable or not. In the latter case it sends a ResvErr back to the client and notifies its broker with an `ac_failure` message. Otherwise, it only sends a `ac_success` message. The reservations done earlier time out after a while, thus restoring the system to a clean state.
The procedure presented here assumes reasonable behaviour from the client side. In particular it assumes that the client doesn't send additional Resv messages shortly after receiving a ResvErr message. In cases where that assumption doesn't hold, such undesired messages can be blocked by denying authentication.

2.2.4 Leaving Client

Receivers of the video transmission can also leave the session while it is still running. Luckily, we need no additional functionality to handle this. When a client wants to stop receiving the video he or she simply leaves the multicast group and stops refreshing the reservation. Consequently, the multicast routers notify the brokers of the event by sending them `LEAVE_NOTIFICATION` messages, which in turn can reduce the scheduled reservations to fit the pruned tree and adjust any special codepoint settings due to unauthorized subtrees. They can also leave out the first step if the client didn't join dynamically in the first place.

2.2.5 Session Termination

Unlike session setup, session termination is a trivial procedure. At the end of the video transmission the content server simply stops sending and the QoS brokers cancel the scheduled reservations. The information stored in the brokers can now be used to calculate the effective costs of the reservations, both scheduled and dynamic. Any other state information in the network is now useless and can be destroyed. It should be noted here that the clients are informed about the end of the transmission by the content server.

3 QoS Broker

A central part of the QoS reservation architecture is the role of the QoS Broker. Every DiffServ domain must have its own broker as a central point for resource managing, admission control and authentication services. The bandwidth broker design used in StreamCom is based on the one described in [14], whose original field of application is QoS management for unicast connections. This section presents the design extensions needed to provide QoS for multicast groups in the context of the StreamCom model.

3.1 Internal Extensions

Inside the QoS broker, a new data structure must be created, containing the topology information of all controlled multicast trees, including supplemental information such as authentication state, timers and multicast routing tables with their DSCP extension. The basis for this structure is a new MCRouter class, which contains per-node information and pointers to links (graph edges) going to adjacent nodes, and which can be based on the original router class used in the broker. Thus, the structure enables algorithms to view multicast trees as subgraphs of the hardware based topology of the domain. Section 3.3 describes some problem specific algorithms using this structure. The information inside the structure gets updated through communication with external entities. The next section further describes the related procedures and protocols, with the exception of the reservation scheduling interface and broker to policy server interaction, which are covered in [2].

3.2 Interfaces to other Entities

3.2.1 Multicast Routing Daemons

In order to keep the QoS broker informed about the multicast topology inside its domain and to support different DiffServ class points for different multicast subtrees, the interaction between the broker and the multicast routing daemons located on the routers must be enhanced with the messages proposed in [4]. There are two messages used to update routing information, `JOIN_NOTIFICATION` and `LEAVE_NOTIFICATION`. Both messages carry two fields, the joining or leaving multicast group address and the address of the next hop. When the multicast tree's topology changes, the routing daemons automatically issue the appropriate messages to the broker. Although simple, this approach suffices to maintain the topology information inside the broker in a scalable way.
A third kind of message, `SET_CODEPOINT`, enables the broker to assign arbitrary DSCPs to subtrees. The required fields in the message are the multicast and next-hop addresses and the DSCP value to be inserted into replicated packets going into that direction. In contrast to [4], there is also a "magical" DSCP value preventing the routing daemons to change the DSCP fields of a multicast group. The broker only sends `SET_CODEPOINT` messages when the necessity arises, i.e. when the authentication period of a subtree expires without a successful authentication or before and after a MBAC procedure.

3.2.2 RSVP Daemons

When RSVP Resv messages from clients or downstream domains are received on the network borders, it is the task of the RSVP daemons located on the edge nodes to communicate with the QoS broker and let it verify the authentication keys. The daemons do this by sending `auth` requests to the broker, which include the reservation objects as arguments. Figure 14 shows the actions taken inside the broker once a request has been received. The three possible values a broker can return to the requesting RSVP daemon are `OK`, `OK_AND_ANCHOR` and `FAIL`. The second tells the daemon to accept and forward the Resv message like a normal `OK`, but to include its own address in a special address object, thus 'anchoring' an MBAC procedure spanning multiple domains. After that procedure has completed, the downstream agent communicates the result back to the broker who initiated it as described in section 3.2.3.

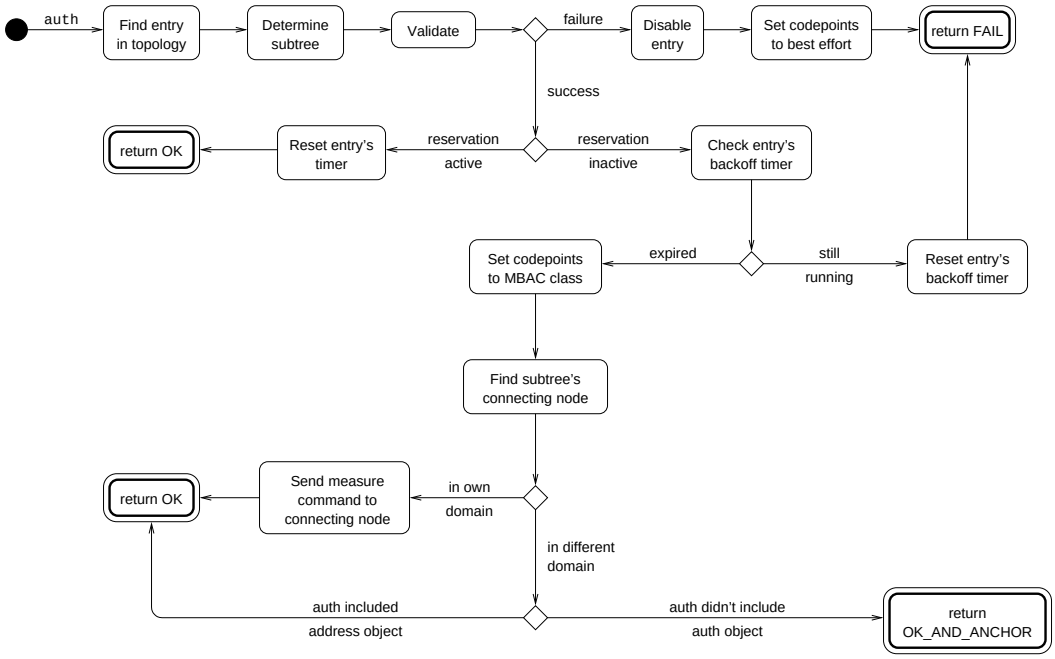


Figure 14: QoS broker algorithm for `auth` requests

3.2.3 Measuring Agents

There are two interfaces between QoS brokers and measuring agents, one to start an MBAC procedure as used in Figure 14 and one to gather the results. The former is done by issuing a `measure` command to the upstream agent using the argument format from section 4.3.1. The latter happens through asynchronous messages (`ac_success` and `ac_failure`) sent by the downstream agent. Figure 15 shows how the broker reacts to these messages. This approach was chosen because the starting of MBAC sessions and the gathering of results needs to be done on different brokers if the session spans multiple domains.

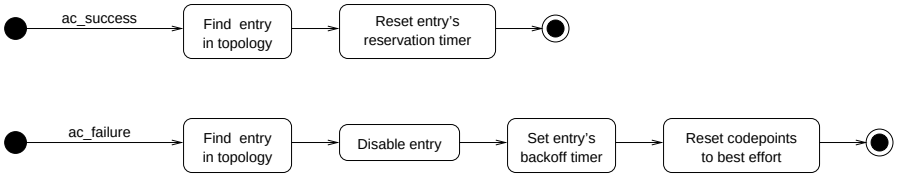


Figure 15: QoS broker actions for `ac_*` messages

3.3 Algorithms

The responsibilities imposed on the QoS broker require a few specialized algorithms. Here we show two of them, beginning with an algorithm identifying the authorized and unauthorized subtrees inside a domain, using a graph coloring approach (Figure 16). The first step is to create a tree structure from the topology database stored in the broker. Each node is marked as uncolored, except for the egress nodes who are marked according to their authentication status — i.e. *authorized*, *unauthorized*, *connecting*, where the lattes signifies a connecting point of authorized and unauthorized trees. Starting from there, all nodes of the authorized subtree get marked by tracing the multicast tree back from the egress nodes to the ingress node. The remaining part of the tree is then marked as *unauthorized*, or *connecting* if the node is on the border between both parts.

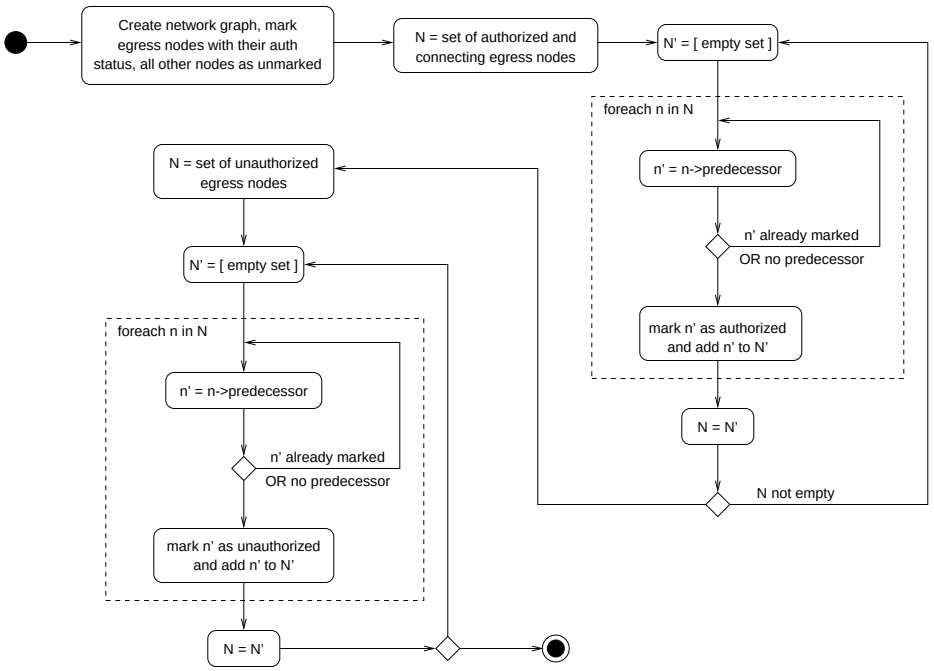


Figure 16: “Find unauthorized subtrees” algorithm

Finding the branch between an egress node and a (a priori unknown) connecting node is a frequently occuring task when doing measurement based admission control. It is solved with the algorithm shown in Figure 17. Its principle is very simple: Starting at the egress node, follow the multicast tree backwards until a node belonging to the authorized subtree is found. The result is a list of nodes, starting at the connecting node and ending at the egress node.

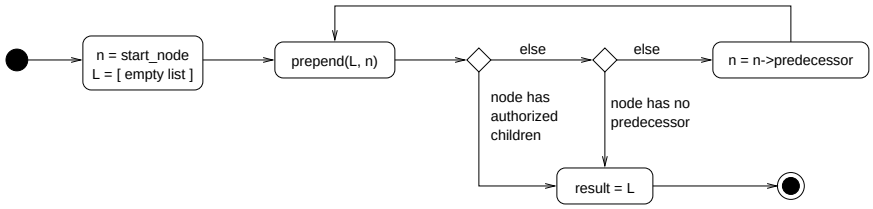


Figure 17: “Find connecting branch” algorithm

4 Routers

StreamCom is based on DiffServ functionality. Consequently, we presume that every participating ISP network is fully DiffServ capable and supports a minimal common set of PHBs (i.e., best effort service for common traffic and two prioritized traffic classes for scheduled and probed reservations, respectively). Multicast support is also necessary, and edge nodes must support RSVP signalling. Measuring agents on each router add the ability to enhance the QoS reservations to include dynamically joining additional clients.

4.1 Multicast Routing Daemons

There are multiple variants of multicast routing daemons in use today, each supporting a different routing style. This poses a problem for the StreamCom architecture, since modifications to all of them are necessary unless we restrict

the set of supported daemons. Fortunately at least, those modifications can be done to daemons of all known routing styles. The modifications that need to be done have been proposed in [4]. In short, they consist of the following:

- Multicast routing table entries are extended with a DSCP field, which is applied to all replicated packets. By default, a special value is used, preventing the daemon from changing the DSCP field of passing packets.
- The QoS broker can change this additional field by sending a `SET_CODEPOINT` message.
- When a new subtree joins or leaves, the daemon notifies the QoS broker by sending `JOIN_INDICATION` or `LEAVE_INDICATION` messages, respectively. These events must be defined separately for every routing style.

Additionally, there must be a mechanism for a measurement agent to control the forwarding behaviour of packets when a measurement session is active. How this can be done depends on the router's operating system. Possible solutions would be to either preempt the multicast routing daemon, thus controlling which packets reach it, or to define an interface between the routing daemon and the measurement agent in order to delegate the forwarding decision to the latter. The modified multicast routing daemons will have to be deployed on every multicasting router in the planned area where StreamCom services will be offered.

4.2 Authentication with RSVP

To allow authentication information to propagate to through the network, specific nodes have to support RSVP signalling or at least a subset thereof. These are the content server, the edge nodes of the participating ISP domains and the endpoints. Their roles can be seen in context in the examples of section 2. This section focuses only on the responsibilities of the edge nodes.
RSVP messages are handled differently depending on the link on which a node received them. If they come from the direction of the multicast source or from the interior of the own network they are simply forwarded along the multicast tree. Otherwise, they trigger an authentication procedure[5].
This procedure is initiated by sending a `auth` request to the broker, including all reservation objects contained in the Resv message. What happens in the broker remains transparent to the RSVP daemon, it only sees the outcome, which can be one of the following: First, the broker can reject the authentication (`FAIL`), in which case the daemon must generate a ResvErr message and send it back down the multicast tree. Second, it can accept the authentication (`OK`). The daemon must then either forward the Resv message upstream along the multicast tree or merge it if appropriate. Third, the broker can accept the authentication but require a MBAC procedure (`OK_AND_ANCHOR`), causing the daemon to add an additional reservation object containing its address to the Resv message before forwarding it.

4.3 Measuring Agents

Measuring Agents (MAs) are necessary because clients should be able to dynamically join a session as explained in section 2.2.3. Their purpose is to perform admission control based on the measurement of probing streams sent over the network between two agents. Since the network route to be measured can lie between two arbitrary nodes, every node of every DiffServ domain in the broadcast area must be equipped with an agent. MAs act only upon request from the broker or from another MA. They can take one of two roles at a time: The *traffic generator* (aka *upstream agent*) and the *traffic sink* (aka *downstream agent*).
At the beginning of a measuring session the broker contacts the upstream agent, which then takes control of the measuring procedure. Included in the request are the downstream agent's address, the multicast address to use, the desired maximum packet loss rate, the peak rate of the video stream and its DiffServ class point. After receiving a request the upstream agent contacts the designated downstream agent and begins to send probe packets, after a brief exchange of measuring parameters. Following the method on page 6, it adapts the bandwidth of its probe packets to account for the current bandwidth of the video stream. When the number of sent packets reaches the level required

[5] RSVP messages coming from a client or from a downstream domain are always Resv messages.

to guarantee the desired loss rate, it stops sending and notifies the downstream agent of the event, which ends the procedure simply by confirming. Finally, the downstream agent indicates the outcome of the admission decision to its QoS broker by sending an `ac_success` or `ac_failure` message. Figures 18 and 19 give a graphical view of this procedure.

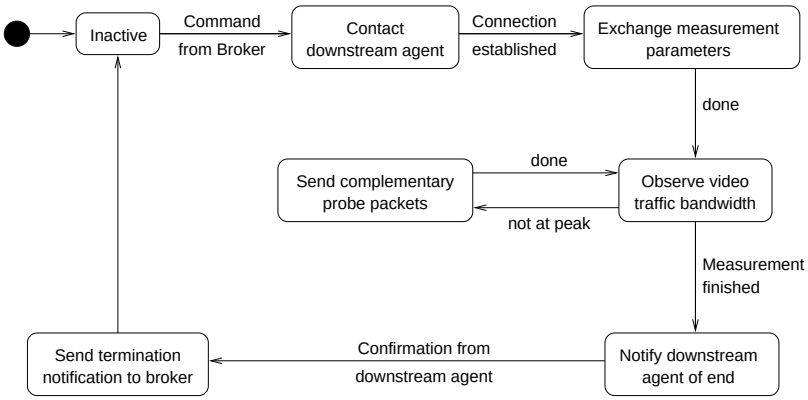


Figure 18: Upstream measuring agent state diagram

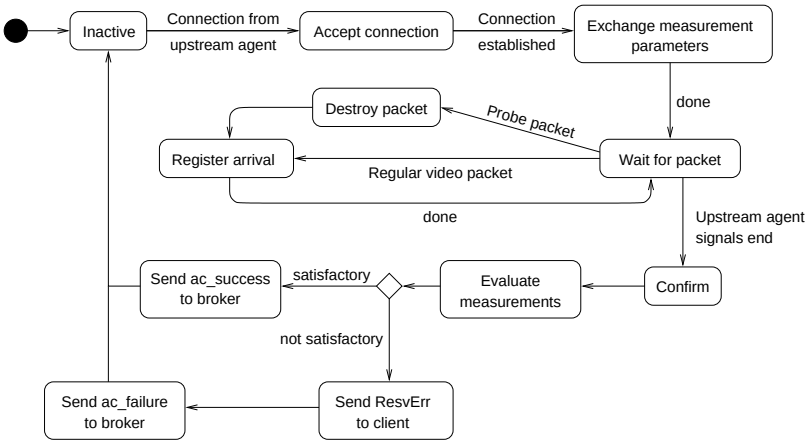


Figure 19: Downstream measuring agent state diagram

The outcome of the measurement is known to the downstream agent and its QoS broker only. This is due to the definition of the dynamic joining procedure from section 2.2.3 and might be changed if experience shows that more control becomes necessary.

Measuring agents interact across network borders. Thus, they need to verify the authenticity of the messages they receive and the authorization of the entity sending those messages. We propose to use an approach similar to the one used in secure shell (ssh, [15]) by encrypting and authenticating the connection with asymmetric keys. However, in order to make the approach more scalable, keys shouldn't be created for every agent but rather for every domain. Since the exact specification of such an algorithm would go beyond the scope of this document, we won't expatiate this topic any further here.

4.3.1 Broker-Agent Interaction

The interaction between QoS brokers and measuring agents is very simple: To start a measurement, a broker sends a command to a measuring agent, containing all required information to perform the admission control decision. These informations are encoded as follows:

DSAgentAddr: IPv4 address
Downstream agent's address, encoded as a 32 bit integer, wich is the address of the client's last-hop router.

MCAddr: IPv4 address
Address of the multicast group, encoded as a 32 bit integer. Probe packets must be sent to this address.

LossRate: Single precision float
The desired maximum packet loss rate, determining the measurement duration. Must be in the interval (0,1], where 0 stands for a perfect transmission and 1 for no guarantees at all, similar to best effort. Probably the possible values for this field will in practice be restricted to a very small set, to increase the reliablitiy of the measurements.

PeakRate: 32 bit unsigned integer
Expected peak rate of the video stream, in bytes/second. If the rate of the stream falls below this value during the measurement, the agent must generate additional probe packets to keep the total bandwidth on this level.

DSCP: DiffServ class point
The DSCP value to be used for any probe packets.

ResvObj: RSVP reservation object
The RSVP reservation object originally sent by the client requesting the reservation, needed to send back a ResvErr message if the admission control decision turns out negative.

This information gets handed over by remotely calling a function in the MA using the RPC mechanism. RPC was chosen because of its relative ease of use with regard to prototype implementation and because of current development work on a broker prototype based on RPC, but it can be substituted with any other protocol style. The function is declared as follows:

ErrorCode **measure**(DSAgentAddr, MCAddr, LossRate, PeakRate, DSCP, ResvObj)

A returned error code other than zero indicates failure. No special error codes have been defined at the time of writing.

4.3.2 Agent-Agent Interaction

Most of the communication during a measurement based admission control procedure takes place between the upstream and downstream agents, possibly across domain borders, thus making the protocol considerably more complex than in the above section. Since measuring agents must communicate across domain borders, the use of RPC can become a security hazard. It is therefore replaced by a line based TCP protocol.
When the upstream agents receives a `measure` command from the broker, it first opens a secure TCP connection to the downstream agent. After this handshake procedure, the relevant parameters are transmitted, which are the multicast group address, the desired loss rate, the expected peak rate and the original reservation object sent by the client. Then, the upstream agent announces the begin of the measurement and starts sending complementary probe packets. When the number of sent packets is sufficient to make an admission control decision, it stops sending and notifies the downstream agent, which confirms and closes the connection. Figure 20 shows an example.

4.3.3 Measuring Method

A central aspect of an MBAC architecture is the kind of measurement method used. Following the classification from [7], we can describe the method used in this architecture as in-band dropping. Since MBAC easily interferes with other allocation styles (e.g. bandwidth brokers), an additional isolated traffic class is allocated in which reservations can only be done by measuring first.
When the upstream agent announces the start of the measurement, the downstream agent starts intercepting incoming multicast packets. It keeps an account of all received packets of the multicast group in question and prevents

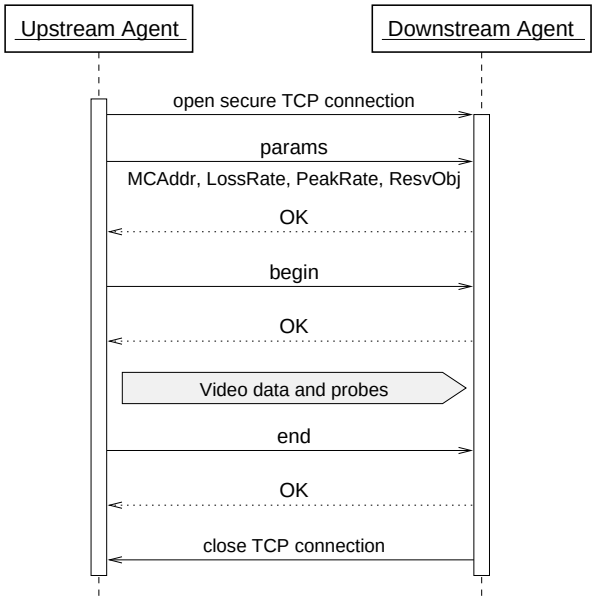


Figure 20: Agent-agent interaction

forwarding of any probe packets sent by the upstream agent. When the measurement terminates, the agent calculates the bandwidth of the received packets and the loss rate based on the expected peak rate. If it is lower or equal to the desired loss rate the admission is granted, otherwise it fails.

4.4 Distribution of Functionality

This section briefly summarizes, which of the tasks described above are assigned to which nodes inside a DiffServ domain. *Core nodes* have the least responsibility: They must support DiffServ and carry a modified multicast routing daemon and a MBAC agent. *Edge Nodes* also need an RSVP daemon. Depending on the multicast routing protocol used, intermediate nodes may lack a multicast routing daemon, although in this document we assume the presence of a daemon on every router.

5 Clients

Like other entities in the StreamCom architecture, clients are required to provide basic set of functionalities. Most of these or out of this document's scope, namely the details of subscribing to the service, of exchanging tickets for decryption keys and of how to generate reservation authentication keys. The functionalities that are of concern here are the multicast and RSVP capabilities.
There are two ways to join a session: First, a client can follow the standard procedure by subscribing during advertisement period and joining when the session starts. The advantage here is a guaranteed QoS reservation. Second, he or she can buy the tickets after the advertisement period and/or try to join the session after the initial setup has been completed. Either way, while the observed QoS may differ, the procedure stays almost the same. This procedure works as follows:

1. Join all needed multicast groups.
2. Wait until a RSVP Path message is received.
3. Calculate an authentication key.

4. Send RSVP Resv messages containing the previously calculated key for all joined multicast groups.

5. If a ResvErr message is received and the standard procedure is used, leave the session. Otherwise, if a ResvErr is received when joining after initial setup, either leave the session or back off for a random amount of time.

6. Video transmission is now working. Repeat the procedure starting at point 2 in regular intervals.

The details of authentication and reservation setup remain transparent to the client.

References

[1] R. Balmer, F. Baumgartner, and T. Braun. A concept for RSVP over diffserv. In *9th International Conference on Computer Communication and Network (ICCCN 2000), Las Vegas*, October 2000.

[2] Roland Balmer. Streamcom key validation infrastructure: Todo. TODO.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. IETF, December 1999.

[4] R. Bless and K. Wehrle. Ip multicast in differentiated services networks. IETF, September 1999.

[5] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. IETF, June 1994.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) – version 1 functional specification. IETF, September 1997.

[7] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *SIGCOMM 2000, Stockholm*, 2000.

[8] C. Cetinkaya and E. Knightly. Egress admission control. In *IEEE INFOCOM, Tel Aviv*, 2000.

[9] F. Kelly, P. Key, and S. Zachary. Distributed admission control. In *IEEE JSAC*, 2000.

[10] Dimitri Konstantas and Torsten Braun. Streamcom: Admission control for multicast streams over differentiated services networks. SNF, 2001. Project No. 5003-057755.

[11] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. IETF, December 1998.

[12] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. IETF, July 1999.

[13] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. IETF, January 1999.

[14] Günther Stattenberger. Qos broker: Todo. TODO.

[15] Secure shell. SSH (http://www.ssh.org).

[16] Unknown. The basic streamcom system: Todo. TODO.

Contents