Use of COPS for Intserv operations over Diffserv: Architectural issues, Protocol design and Test-bed implementation

Roberto Mameli, Stefano Salsano (*)

(*) CoRiTeL
Consorzio di Ricerca sulle Telecomunicazioni
Via di Tor Vergata, 135
00133 Roma (Italy)

Abstract

This paper describes a solution for the Intserv operations over Diffserv network, based on the use of a Bandwidth Broker for the resource allocation.
The Architectural scenario is described, which is based on the current IETF work in this area. For the communication of the admission control request to the Bandwidth Broker, the use of COPS protocol is proposed and the detailed protocol specification have been given. In particular, a new client type for the COPS protocol ids proposed to support dynamic DiffServ admission control. The Policy Decision Point (PDP) acts as a "Bandwidth Broker" for the Policy Enforcement Point (PEP) which is requesting resources. The use of the defined mechanism is suited for (but it is not limited to) the Integrated Services operation over Diffserv networks.
The proposed model has been implemented in a test-bed, where both the control plane and the data plane are realized according to the specification.

1. Introduction

The two traditional approaches to QoS provisioning in IP networks, namely the Integrated Services (Intserv) and the Differentiated Services (Diffserv) Architectures are characterized by opposite choices. In fact, the former is stateful and per flow-based, while the latter is stateless and manages aggregates. As a consequence, they obtain opposite advantages and disadvantages. A possible solution that tries to conjugate benefits of both the Intserv and the Diffserv approaches is based on a proper combination of them, using Intserv in the access and Diffserv in the core. In fact, it achieves scalability due to the Diffserv aggregation in the core, while keeping the advantages of end-to-end signaling. [1] describes such a solution, but leaves some open issues. One of them is related to resource management within the Diffserv domain. At least three options exist:

- Statically provisioned resources
- Dynamically provisioned resources by means of RSVP
- Dynamically provisioned resources by means other than RSVP

In this work we focus on the third solution, where Admission Control in the DiffServ network is based on a centralized device called Bandwidth Broker (BB). The use of a server to admit and reject traffic within a Diffserv domain has been considered since the very beginning of the discussion about the Diffserv architecture [2]. The reference architectural scenario for our work is described in Section 2. In the scenario described the routers at the boundary between the Intserv and the Diffserv networks plays a key role. In fact they are in charge of performing Admission Control with the help of the BB. An intra-domain scenario is assumed, where the BB is in charge of controlling resource for a network in a single administrative domain.
The Edge Router (ER) communication with the Bandwidth Broker is realized by means of an extension to the COPS protocol. The COPS is used to exchange resource allocation requests/responses. The Edge Router contains the PEP – Policy Enforcement Point (client side of the COPS protocol), while the Bandwidth Broker plays the role of the PDP – Policy Decision Point (server side of the COPS protocol). Sections 3 and 4 provide information on the proposed COPS extensions, while the detailed specification can be found in [3].
The interworking of RSVP and COPS protocols in the Edge Router is described in section 5, while the test-bed implementation is described in section 6.

2. Intserv over Diffserv: the role of the Edge Router

The Intserv over Diffserv scenario described in [1] is reported in Figure 1:

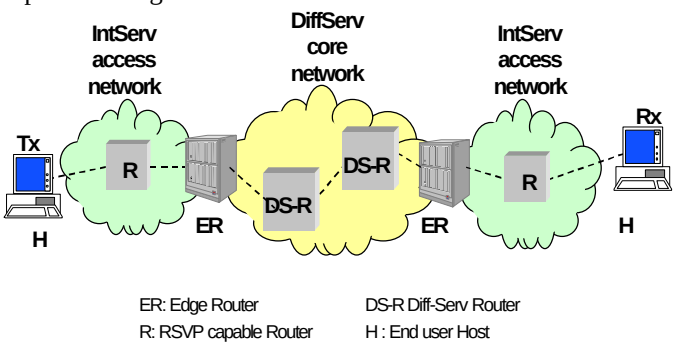

Figure 1 – IntServ/DiffServ interoperation scenario

As previously mentioned, end-to-end QoS is achieved exploiting Intserv in the access networks and Diffserv in the

core. This allows scalability, due to the Diffserv aggregation, while keeping the advantages of end-to-end signaling by means of the RSVP protocol. Let us describe the sequence of operations involved in a single end-to-end reservation in order to clarify how Intserv and Diffserv interact:

- The sender application on Tx generates an RSVP PATH message carrying the flow's characteristics (i.e. the Tspec). The message is carried towards the receiver Rx; in the IntServ access regions it is subject to standard RSVP/IntServ processing, while in the DiffServ core it is forwarded transparently.

- When the receiving host Rx receives the PATH it generates the corresponding RSVP RESV message, that is carried back towards the sending host. As before, the RESV message undergoes standard RSVP/IntServ processing in the IntServ access regions and is transparently tunneled in the DiffServ core. During his travel backwards, assuming that it is not rejected before for resource unavailability, it reaches the ingress Edge Router ER1.

- In ER1, the RESV message is processed. The ingress Edge Router ER1 is a key element in the sequence of operations described. A detailed description of its tasks is reported below: among them it performs Admission Control, i.e. based on the information carried in the RESV message and on its view on the utilization of network resources it decides to admit or reject the request.

- If ER1 approves the request, the RESV message is admitted and is allowed to continue upstream towards the sender. If it rejects the request, the RESV is not forwarded and the appropriate RSVP error messages are sent.

Differently from core routers, the ER is RSVP aware and stores per-flow states; the ER is capable of managing packets both on a micro-flow basis and on an aggregate basis. The choice between the two possibilities depends on the role of the Edge Router. When it acts as ingress ER, i.e. for packets from the originating Intserv network to the Diffserv core it forwards packets in an aggregate fashion on the outgoing interface, while it can handle micro-flows on the incoming interface. Instead, when it behaves as egress ER, i.e. for packets from the Diffserv core to the destination Intserv network, it is able to distinguish micro-flows on the outgoing interface. Note, however, that the distinction between ingress and egress edge router depends only on the direction of the data stream. This means that the same ER may be ingress ER for a flow and egress ER for another one in the opposite direction.

Let us focus on the ingress Edge Router, since it provides some important functionality. Among them we cite:

- Classification: it performs per micro-flow classification, i.e. it is able to distinguish the different Intserv flows.
- Mapping: it performs mapping of IntServ service classes into DiffServ PHBs; it is also in charge of aggregating IntServ micro-flows into DiffServ aggregates.
- Marking: it marks (or remarks) the DS field of incoming packets according to the target PHB, that in turn results from the mapping operation explained above.
- ADSPEC update: for GS flows the exported terms in the ADSPEC (i.e. C and D) must be properly updated with a value depending upon the topology and the characteristics of the DiffServ core.
- Admission Control: this is one of the main tasks performed by the ingress Edge Router. The Admission Control is applied to the virtual hop represented by the DiffServ core network. The purpose of this procedure is to ensure that Diffserv resources are available in the Diffserv domain to support the requested Intserv flow. In a "pure" DiffServ network per flow Admission Control is not needed, as simpler "aggregate" policing at ingress points based on provisioning can be used. As explained in [1], the purpose of per-flow admission control is to increase network utilization and/or to support tighter end-to-end QoS guarantees (at the expense of increased complexity).

There are basically two distinct approaches to the realization of Admission Control in the ingress Edge Router:

- Distributed: in this case Admission Control is based on information locally available in the ingress Edge Router.
- Centralized: differently from the previous choice, Admission Control is realized by means of a Bandwidth Broker (BB), i.e. logically centralized entity acting as an Admission Control server. The BB could use global knowledge of both the network topology and the resource allocation (see Figure 2) to take Admission Control decisions. The definition of mechanisms and algorithms used for the BB operation is outside the scope of this document and will not be further detailed here. Related work can be found in [4].

The distributed solution is quite simple to implement, but it is also rather inaccurate, since each ER exploits information with a local scope, without the overall vision of the network status. In contrast, the second approach raises some non-trivial issues in terms of complexity and scalability, but allows better resource utilization within the DiffServ cloud. An architectural definition and scalability analysis of the centralized scenario can be found in [5].

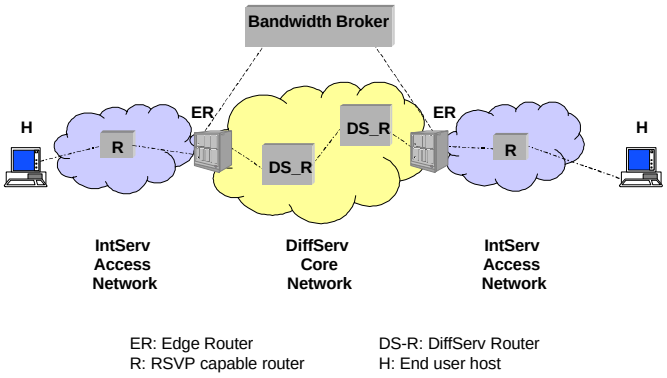

Figure 2 – Bandwidth Broker

In this document the centralized solution of Figure 2 is analyzed. A protocol for the communication between the ER and the centralized server is needed. The use of the COPS [6] is considered in this work. The COPS protocol can be extended to support new Client types. Using COPS for Admission Control in a Diffserv network has been already discussed. For example in [7] such a scenario is described, but the definition of the extensions to the COPS protocol is not provided.

In this work we describe the use of COPS for outsourcing the allocation of resources in a Diffserv network. The new client-type is called “COPS-ODRA” (COPS- Outsourcing Diffserv Resource Allocation) [3]. The COPS-ODRA client relies on the Outsourcing model as it will explained in the next section. The COPS-ODRA client type is especially suited to support the Intserv operation on Diffserv network, but can also support other scenarios for Diffserv resource allocation, without using Intserv in the Access network.

Note that the scenario depicted in Figure 2 refers to a single DiffServ core domain (intra-domain case). In a multi-domain scenario, one could simply use of RSVP as end-to-end signaling (raising scalability concerns) or more complex communication mechanisms between BBs of different domains could be defined. These inter-domain aspects are outside the scope of this paper.

3. Outsourcing and provisioning models for resource allocation

The COPS (Common Open Policy Service) protocol is a simple query and response protocol that allows policy servers (PDPs) to communicate policy decisions to network devices (PEP). Two main models are supported by the COPS protocol: outsourcing model and provisioning model.

The Outsourcing model is used when there are "Trigger events" in the PEP that require a policy decision (e.g. a dynamic request to admit a new flow). The PEP delegates this decision to an external policy server (PDP). It sends a query message to the PDP, typically waiting for the response decision before admitting the new flow (see Figure 3). The Outsourcing model is used for the RSVP client type defined in [10].

On the other hand, the Provisioning model [8] foresees that the PDP proactively configures the PEP so that it knows how to run its QoS mechanisms. The mechanisms to exchange the configuration information and to store this information is based on the definition of a "Policy Information Base", see [11].

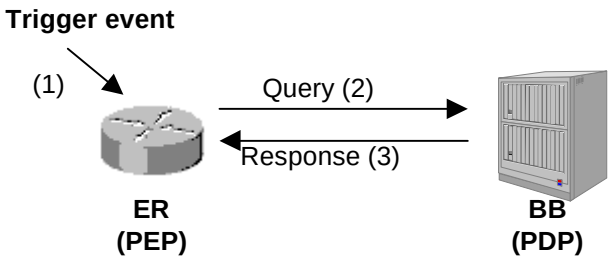

Figure 3 – COPS-ODRA Outsourcing Model

In the Provisioning model, either there are no "Trigger events" at the PEP (i.e. only packet classification, marking, scheduling, etc. is performed at the PEP) or these events must be handled using local information (i.e. mapped in the available resources provisioned by the PDP). The two models should be compared against scalability, flexibility, efficiency in resource allocation and implementation complexity.
The Provisioning model is very well suited when there are no such dynamic requests coming to the PEP. In other scenarios, like for example the RSVP/Diffserv interworking the dynamic requests are a fundamental feature in the PEP/Edge Router. In this case a possible solution is to fully rely on the Outsourcing model, so that a very simple PEP can be defined. The drawback is the need of relatively frequent communication with the PDP, but there are scenarios where this solution can be cost-effective.
Note that the Outsourcing model lends itself to more sophisticated solutions if scalability concerns arise. In fact the Outsourcing model can be dynamically paced by the PEP in real-time. A straightforward option is to pre-reserve some amount of bandwidth and to make Admission Control Request with a coarser bandwidth granularity to limit the pace of requests.
In general, a combination of Outsourcing and Provisioning model could be used to provide a flexible and general solution for QoS in IP networks. As a first step, the solution described in this paper (and realized in the test-bed) focuses on the pure Outsourcing case

In the Outsourcing model used by the COPS-ODRA client the PEP explicitly asks the PDP/BB for a given amount of resources, from an ingress point to an egress point. Note that per-flow state is not stored in the PDP/BB. Instead,

resource allocation requests are properly aggregated and only aggregate state information is kept in the PDP/BB, allowing for higher scalability. The RSVP client type defined in [10] has a different behavior, keeping a separate state in the PDP for each RSVP flow.

4. Use of the COPS protocol for Outsourcing Diffserv Admission Control

In order to be flexible, the COPS protocol has been designed to support multiple types of policy clients. Each client-type will be described in a different usage draft. The detailed definition of the COPS-ODRA client type is provided in [3]. The purpose of this section is to provide an overview on the operation of the proposed client type.

- Start of operations

In order to start operations, the PEP must open the dialogue connection with its PDP/BB. First, a TCP connection is established between the client and server and the PEP sends a Client-Open message with the Client-Type = COPS-ODRA. If the PDP supports this client type, it responds with a Client-Accept (CAT) message. If the client type is not supported, a Client-Close (CC) message is returned by the PDP to the PEP. After receiving the CAT message, the PEP can send requests to the server.

- Common operations

Following the terminology in [9], the PEP will send "Resource Allocation Requests" (RAR) to the PDP/BB once the connection is established.
A Resource Allocation Request will contain:

- the topological information (e.g. ingress point and egress point in the Diffserv domain) which allows the PDP/BB to aggregate different Resource Requests
- the type of the requested resource (e.g. the Diffserv Per Hop Behavior or Behavior Aggregates)
- the amount of the requested resource (e.g. the specification of the bandwidth)

The resource allocation request is used by the PDP to perform the Admission Control procedure. According to the requirements of the requested service, the PDP/BB will properly map the requested edge-to-edge resources into network resources and will assure that such resources are available throughout the Diffserv cloud. The discussion of the Admission Control algorithms in the PDP/BB and of the mechanisms used by the PDP/BB to get topological/routing information from the Diffserv domain are outside the scope of this document. Related work can be found in [4], where the use of OSPF and SNMP for the communication between the BB and the Diffserv routers is proposed.

In response to the Resource Allocation request, the PDP sends a reply where it accepts or rejects the RAR. On receiving the answer, the PEP activates its local QoS mechanisms as needed.

- State information in PEP and PDP/BB

The COPS protocol is stateful in the sense that Request/Decision state is shared between PEP and PDP. Depending on the COPS client type, one or multiple states can be installed in the context of a single PEP/PDP relationship. The "handle" object uniquely identifies a single installed state at the PEP and at the PDP side. In case of RSVP client type [10], a different state is installed for each RSVP flow (actually one PATH state and one RESV state). This implies that a lot of state information is duplicated in the PEP and in the server.
In COPS-ODRA the state information is aggregated: the state represents the set of resources allocated by the PDP/BB to a PEP. Therefore a unique value for the handle object is used in the context of a single PEP/PDP relationship. The handle is inserted by the PEP in the first request and then it is used in every message by the PEP and by the PDP/BB.
The state information in the PDP/BB is represented by the set of the triples (resource type, ingress point, egress point) and the corresponding amount of allocated resources. The PDP/BB keeps this state information separately for each different COPS-ODRA client (i.e. for each connected PEP). The requests for the same resource type, ingress point, egress point coming from the same PEP are properly composed by the PDP/BB so that only the aggregate information is stored.
This aggregate state information stored in PDP/BB is logically shared by the PEP in the sense that it is the result of the sequence of Request messages sent and of the Decision messages received. There is no need in the PEP to evaluate and store the aggregate state information: in the simplest case, the client side (PEP) stores a set of "per flow" reservation information. In more advanced scenarios, the PEP client can evaluate and store aggregate information.
Temporary state information per each request must be stored in the PEP and in the PDP/BB in order to correlate requests with decisions. To this purpose the notion of request ID is needed and a corresponding client specific protocol object is defined (see section 3). This temporary information is deleted in the PDP/BB when the Decision message is sent and in the PEP when this message is received.

- Synchronization

Synchronization procedures are foreseen in COPS specification to cover failure situations. The basic idea is that the PDP/BB can "reset" the state and ask the PEP to rebuild it by sending proper resource allocation Request messages. If the PEP has only stored "per-flow" state, it will send one Request message for each active reservation. If the PEP has stored aggregate states, it can send "aggregate" Resource Allocation requests.
Selective re-submissions (i.e. for resource type, ingress or egress point) can be supported.

4.1. Message types

The COPS protocol provides for different COPS clients to define their own "named", i.e. client-specific, information for various messages. This section describes the messages exchanged between a COPS server (PDP) and COPS ODRA clients (PEP) that carry client-specific data objects.

- Request (REQ) PEP →PDP

The REQ message is sent by COPS-ODRA clients to issue a 'Resource Allocation Request' to the PDP. It can be used to request new resources, to modify a previous reservation or to release a reservation. Each REQ message contains a single request. The PDP responds to the resource allocation request with a DEC message containing the answer to the query. Note that resource allocation request messages can be generated and sent to the PDP in response to the receipt of a Synchronize State Request (SSQ) message.

- Decision (DEC) PDP → PEP

The DEC message is sent from the PDP to a COPS-ODRA client in response to the REQ message received from the PEP. Unsolicited DEC messages cannot be sent for this client type. Each DEC message contains a single decision. The Decision Flags object will contain the answer in the Command-code field according to the COPS specifications. In particular the Command-code will be "Install" to mean a positive answer and "Remove" to mean a negative answer. No report is sent by the PEP to confirm the reception of a Decision message. Only in case of specific errors, the PEP will send back a Report State message to the PDP/BB.

- Report State (RPT) PEP → PDP

For COPS-ODRA client type, the Report State message is sent by the PEP to the PDP in case of problems with a received Decision message. More specifically it is used to communicate that the Decision contains a Request identifier which cannot be correlated to a previous request. This event is the manifestation of abnormal behavior. On reception of a Report State message the PDP could start a Synchronization procedure.

- Synchronize State Request (SSQ) PDP → PEP

The Synchronize State Request message is sent by the PDP to the PEP to "reset" the state information. It requests the PEP to send the set of resource allocation REQ messages needed to rebuild the state. The SSQ can apply to the whole set of PEP active reservations PEP, or to a specific resource type and ingress-egress couple, depending on the information contained in the Client SI object.

- Synchronize State Complete (SSC) PEP → PDP

The Synchronize State Complete message is sent by the PEP to the PDP to inform that all the REQ messages needed to rebuild the state have been sent.

5. RSVP/COPS interaction

From now on the architecture represented in Figure 2 will be assumed as the reference scenario. As previously explained, the Edge Router comprises most of the functionality needed for interworking, including admission control on a micro-flow basis. Obviously, for proper operation, RSVP and COPS signaling should properly interact. To clarify this mechanism let us consider the sequence of operations involved in an end-to-end reservation, that is described in detail in [1] and reported here in a simplified form. We are considering unicast reservations.

As described above, queries from the ingress ER to the BB are triggered whenever the former receives a RESV message from the downstream DiffServ domain. Note that in this situation RSVP and COPS are synchronized. A possible choice in the implementation of RSVP/COPS interaction would be to keep this synchronization. This would imply a blocking behavior; whenever the RSVP daemon triggers a query it blocks indefinitely waiting for a response. This raises the obvious problem of managing situations in which a response from the PDP/BB is not temporarily available. In fact, in such cases, the blocking behavior should be avoided, in order to be able to manage other events and to avoid soft state expirations.

As a consequence the RSVP should be properly adapted in order to manage requests and responses asynchronously. After a query and before getting the response, processing of other events should be possible. A possible way to realize this behavior could be the introduction of pending reservation states in the RSVP daemon. Consider as an example the reception of a RESV message by the ingress Edge Router ER1, that in turn triggers an admission control query towards the PDP/BB. In this case the RSVP daemon could set up a reservation state before retrieving the response, marking it as pending. No further operations related to this state should be performed until it remains pending, i.e. until the response from the PDP/BB is available; this means, for example, that the RESV message should not be forwarded upstream. In the case of positive response (i.e. reservation request accepted), the corresponding reservation should be instantiated on the egress interface of ER1, and the RESV message should be forwarded upstream. In the case of rejection, e.g. due to unavailability of resources, a RESVERR message should be propagated downstream towards Rx, according to standard RSVP behavior. The introduction of pending reservation states could allow the RSVP daemon to perform normal processing related to states other than the one considered. This obviously includes refresh operations triggered by timeout expirations. Note that a timeout could also be defined for pending states. In this case, when the timeout of a pending reservation state expires, and no response has been received in the meanwhile, the RSVP could either reject the request or revert to local decision based on LPDP

functionality in the ER.

Figure 4 and Figure 5 show the typical sequence of operations involved in a reservation, both in the case of acceptance and in the case of rejection. As stated above, the interface between the PEP in the ER and the PDP/BB is realized by means of the COPS protocol extension described in [3], while the interface between the RSVP daemon and the PEP within the ER is described in a [13].

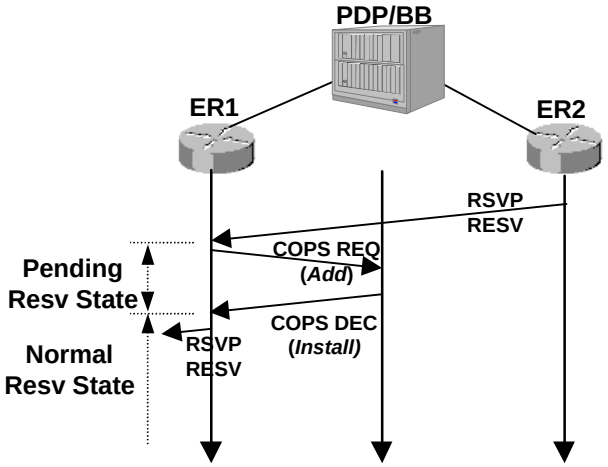

Figure 4 – Reservation successfully accepted

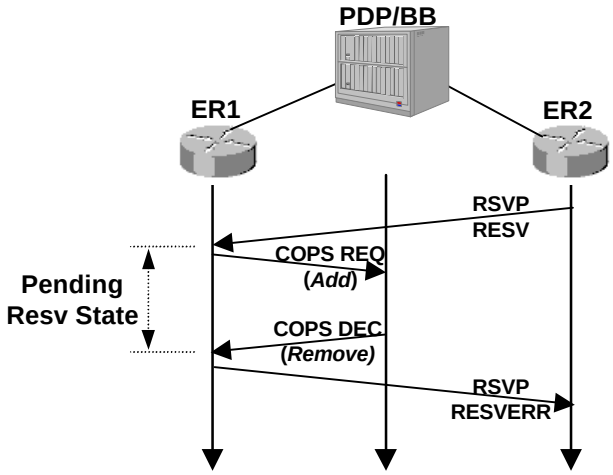

Figure 5 – Reservation rejected

Finally, Figure 6 shows the sequence of events involved in the release of resources. When a RESV TEAR is received a COPS-ODRA REQ message is sent to the PDP/BB, in order to let it keep track of resource usage. After the reception of the corresponding DEC message the PEP notifies the RSVP, which in turn releases the reservation and forward upstream the RESV TEAR. Note however that the same sequence of operations happens in the case of PATH TEAR or in the case of timeout expiration.

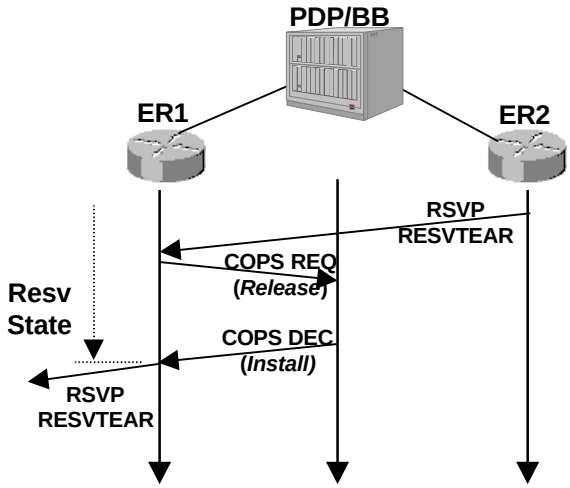

Figure 6 – Resource Release

6. Testbed implementation and future work

This paragraph describes the prototype implementation realized within CoRiTeL lab. The starting point was the testbed configuration described in [12] and reported in Figure 7.

The testbed of Figure 7 is composed of 5 Personal Computers running the Linux operating system (Red Hat 6.1 distribution, kernel version 2.2.12). The PC are equipped with 100 Mb/s ethernet cards. All the functional elements represented in Figure 1 are represented here. The core network is represented by the central PC, that is configured as a Diffserv core router. Traffic control functionality provided by Linux kernel is exploited to realize the Intserv and Diffserv mechanisms (policing, scheduling). In particular, EF and AF traffic classes are supported in the Diffserv network. The two adjacent PCs are configured as Edge Routers; they run a version of the ISI RSVP daemon (RSVPd release 4.2a4) which we have adapted. A complete description of the preliminary changes done in the RSVPd is given in [12]. In the first step these changes were substantially related to the traffic control level (i.e. mapping, marking and micro-flow aggregation), with minor interest in the admission control functionality.

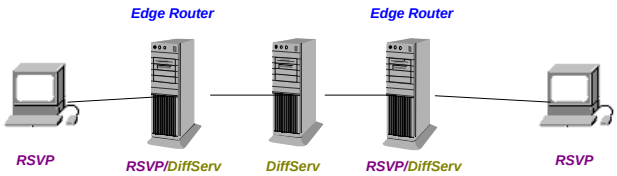

Figure 7 – Original Testbed configuration

The subsequent step has been the enhancement of the test-bed of Figure 7 with the introduction of the Bandwidth Broker (see Figure 8). Currently the test bed includes 4 Edge Routers and 2 Core routers. The COPS-ODRA protocol has been implemented on the client-side and on the server side.

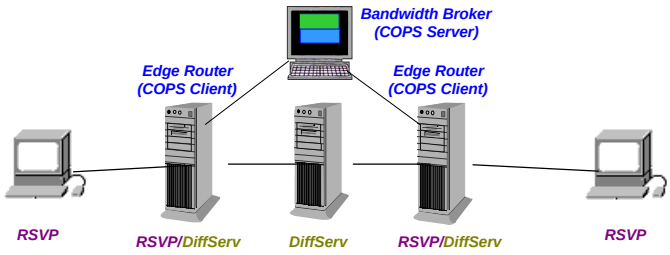

Figure 8 - Testbed Configuration with the BB

The modules that have been developed are shown in Figure 9. The COPS-ODRA on the client side interacts with the RSVPd or with a test application that can generate Resource Allocation requests using the CCAPI. In the current version, the COPS-ODRA server is configured by reading a static table, where the set of available bandwidth for each pair of Edge Routers and for each class is provided. This means that the Decision module and the resource & Topology model are too simplistic. We are working on the second version of the server, which should include topological information on the Diffserv network, in order to map the Edge-to-edge resource requests into the allocation of internal network resources.

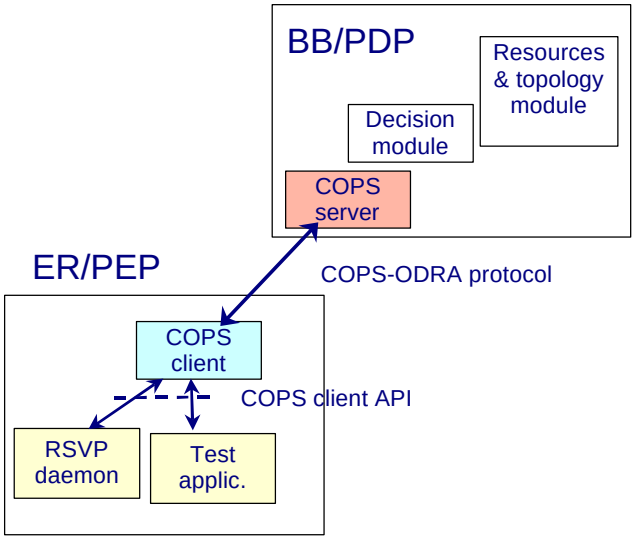

Figure 9 – Software modules

Work is also ongoing to measure the performance of the proposed solution, aiming to infer some conclusion about the possible scalability issues.

7. Acknowledgements

The authors would like to thank Andrew Smith from Extreme Networks who provided the source code of a generic COPS server, Andrea Ferraresi and Eleonora Manconi from CoRiTeL for their work on the specification and the implementation of the COPS-ODRA client-type.

8. References

[1] Bernet, Y., Yavatkar R., Ford, P., Baker, F., Zhang, L., Speer, M.,Braden, R., Wrocklaski, J., Felstaine, E., "A Framework for Integrated Services Operation Over Diffserv Networks", IETF <draft-ietf-issll-diffserv-rsvp-03.txt>, September 1999, Work in Progress.

[2] K. Nichols, V. Jacobson, L. Zhang " A Two-bit Differentiated Services Architecture for the Internet "RFC 2638, July 1999

[3] S. Salsano, “COPS Usage for Outsourcing Diffserv Resource Allocation”, <draft-salsano-issll-cops-odra-00.txt>, February 2000, Work in Progress

[4] O. Schelén, A. Nilsson, J. Norrgard,S. Pink: Performance of QoS Agents for Provisioning Network Resources. In Proceedings of IFIP Seventh International Workshop on Quality of Service (IWQoS'99), London, UK, June 1999.

[5] A. Detti, M.Listanti, S.Salsano, L.Veltri, "Supporting RSVP in a Differentiated Service Domain: an Architectural Framework and a Scalability Analysis", ICC '99, June 1999, Vancouver, Canada.

[6] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry "The COPS (Common Open Policy Service) Protocol", IETF RFC 2748, January 2000

[7] A. Terzis, J.Ogawa, S.Tsui, L.Wang, L.Zhang "A prototype Implementation of the Two-Tier Architecture for Differentiated Services" IEEE Workshop on QoS Support for Real-Time Internet Applications, June 2-4, 1999 Vancouver, Canada

[8] F. Reichmeyer, et al. "COPS Usage for Policy Provisioning", draft-ietf-rap-pr-01.txt, October, 1999, Work in Progress.

[9] R. Neilson, J. Wheeler, F. Reichmeyer, S. Hares (Editors), "A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment" Version 0.7

[10] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry, "COPS usage for RSVP ", IETF RFC 2749, January 2000

[11] M. Fine, K. McCloghrie, S. Hahn, K. Chan, A. Smith, "An Initial Quality of Service Policy Information Base for COPS-PR Clients and Servers", draft-mfine-cops-pib-02.txt, October 1999.

[12] W. Almesberger, S. Giordano, R. Mameli, S. Salsano, F. Salvatore, “A Prototype Implementation for the IntServ Operation over DiffServ Networks”, submitted to Globecom 2000

[13] R. Mameli, “The CCAPI (COPS Client Application Programming Interface)” <draft-mameli-issll-cops-api-00.txt>, February 2000, Work in Progress

[14] R. Mameli, S. Salsano, “Integrated services over DiffServ network using COPS-ODRA” <draft-mameli-issll-is-ds-cops-00.txt>, February 2000, Work in Progress