

Resource allocation in the new fixed and mobile Internet generation

By Guy Pujolle,* Ulf Körner and Harry Perros

In this paper we study the scalability issue in the design of a centralized policy server controlling resources in the future IP-based telecom network generation. The policy servers are in charge of controlling and managing QoS, security and mobility in a centralized way in future IP-based telecom networks. Our study demonstrates that the policy servers can be designed in such a manner that they scale with increase in network capacity. Copyright © 2003 John Wiley & Sons, Ltd.

Introduction

In the world of networks, control techniques are crucial. Packet switching networks are like motorways; if there are too many packets travelling down then none of them get through. Both the network and the data flows must be controlled. Flow control is a preventive measure, it limits the data flows to the physical medium's transfer capacity.

We can define the terms flow control more precisely: flow control is an agreement between two entities (the source and the destination) to limit the service's transmission throughput in accordance with the resources available in the network.

When looking at Internet flow control technique, a lot of effort has been through the slow start and congestion avoidance algorithm, the RSVP protocol, the DiffServ and IntServ schemes, etc. However, none of these proposals have been accepted by the Telcos for their IP networks. The only solution they employ is the oversizing solution.

However, in the near future there will be millions of fixed terminals and millions of mobile terminals to handle. The users waiting for a quality of service, (VoIP, real-time VoD, etc.) will need to support a control of the network that could decline in different QoS bearers.

For that reason, a large group of companies are proposing a novel solution that is supported by a policy-based control using a signalling system.

In the following section we describe the new control-based networking architecture. Then, we present a queuing model to size the policy center and some results that seem of interest on the scalability of this architecture. Finally we conclude with some future extensions of this work.

The Telecom Internet

The Internet Telecom was born from the aims of the UMTS manufacturers. The core network of UMTS has to transport all kind of information with a high quality of service. The first generation

Guy Pujolle works in the Laboratoire LIP6, University of Paris, France, Ulf Körner teaches at Luwn University, Sweden, and Harry Perros at NCSU, USA.

*Correspondence to: Guy Pujolle, Laboratoire LIP6, Université de Paris 6, 4 Place Jussieu, 75252 Paris Cedex, France.

for this core network is based on ATM using the AAL-2 and AAL-5 protocols. This is due to the fact that ATM has built-in mechanisms that permit different quality of service to different types of traffic. Since the UMTS architecture is more and more dependent on IP applications and in order to avoid duplication it was decided that the second generation of the core network of the UMTS has to be IP-based. A look at the different solutions to provide a quality of service in IP networks shows that neither IntServ, DiffServ, RSVP nor native IP using Gigarouters or Terarouters are ideal solutions. The proposal is to come back to an old solution which is quite classical in the telecommunication area. Specifically, they propose to use a signaling scheme in order to allocate the necessary resources to guarantee a quality of service.

The proposal is described in Figure 1. This architecture is described mainly in the RFC 2748¹ and RFC 2753.² Extensions are given in references.³⁻¹⁰

Figure 1 illustrates the layout of various policy components in a typical new Internet architecture. The architecture is composed of two planes: the user plane that could be MPLS or native IP with, for example, IP over DWDM links. The signaling protocol COPS is used to communicate policy information between a Policy Enforcement Point (PEP) and a remote Policy Decision Point (PDP) within the context of a particular type of client. The device to make local policy decisions in the absence of a PDP can use the optional Local Policy Decision Point (LPDP). In the sequel, the PDP is also named Policy Server. The PEP may communicate with a policy server to obtain policy decisions or directives. The PEP is responsible for initiating a persistent TCP connection to a PDP. The PEP uses this TCP connection to send requests to and receive decisions from the remote PDP. Communication between the PEP and remote PDP is mainly in the form of a stateful request/decision exchange, though the remote PDP may occasionally send unsolicited decisions to the PEP to force changes in previously approved request states. The PEP also has the capacity to report to the remote PDP that it has successfully completed performing the PDP's decision locally, useful for accounting and monitoring purposes. The PEP is responsible for notifying the PDP when a request state has changed on the PEP. Finally, the PEP is responsible for the deletion of any state that is

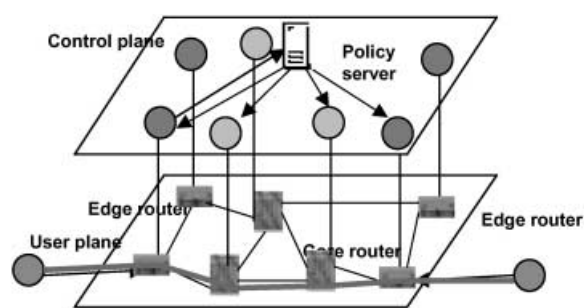


Figure 1. The global architecture

no longer applicable due to events at the client or decisions issued by the server. When the PEP sends a configuration request, it expects the PDP to continuously send named units of configuration data to the PEP via decision messages as applicable for the configuration request.

When a policy is successfully installed on the PEP, the PEP should send a report message to the PDP confirming the installation. The server may then update or remove the configuration information via a new decision message. When the PDP sends a decision to remove a configuration from the PEP, the PEP will delete the specified configuration and send a report message to the PDP as confirmation. The policy protocol is designed to communicate self-identifying objects which contain the data necessary for identifying request states, establishing the context for a request, identifying the type of request, referencing previously installed requests, relaying policy decisions, reporting errors, providing message integrity, and transferring client specific/namespace information. To distinguish between different kinds of clients, the type of client is identified in each message. Different types of clients may have different client specific data and may require different kinds of policy decisions. It is expected that each new client-type will have a corresponding usage draft specifying the specifics of its interaction with this policy protocol.

The context of each request corresponds to the type of event that triggered it. The COPS context object identifies the type of request and message (if applicable) that triggered a policy event via its message type and request type fields. COPS identifies three types of outsourcing events: (1) the arrival of an incoming message (2) allocation of

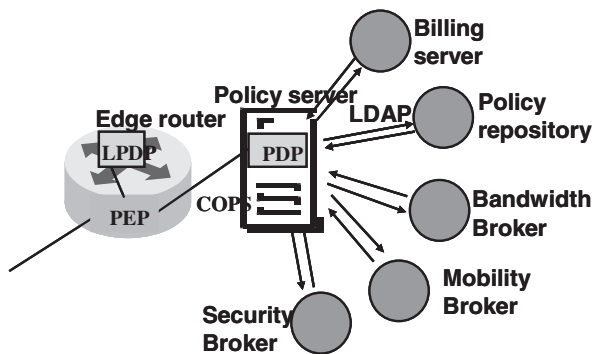


Figure 2. The PDP

local resources, and (3) the forwarding of an outgoing message. Each of these events may require different decisions to be made. The content of a COPS request/decision message depends on the context. A fourth type of request is useful for types of clients that wish to receive configuration information from the PDP. This allows a PEP to issue a configuration request for a specific named device or module that requires configuration information to be installed. The PEP may also have the capability to make a local policy decision via its Local Policy Decision Point (LPDP), however, the PDP remains the authoritative decision point at all times. This means that the relevant local decision information must be relayed to the PDP. That is, the PDP must be granted access to all relevant information to make a final policy decision. To facilitate this functionality, the PEP must send its local decision information to the remote PDP via an LPDP decision object. We describe in Figure 2 the PDP actions.

The PDP is in relation with different servers that have to decide about the demand of the user: First, a repository server accessed through LDAP verifies the access rights of the user. The Bandwidth Broker, the Mobility Broker and the Security Broker provide the user with the resource necessary to get the quality of service in demand.

The main problem concerns the size of the domain under the responsibility of a policy server.

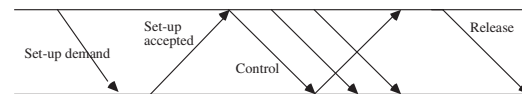


Figure 3. The life of a connection

The Model and the Performance

The main problem concerns the size of the domain under the responsibility of a policy server. The problem is to understand how many customers just one policy server could manage. When several domains have to be traversed to open a connection between two users, the policy servers communicate through the COPS protocol.

We assume that best effort services do not ask for a reservation and therefore do not use the signaling mechanism.

To open a connection a signaling command has to be sent. This is the first action to be performed by a new customer. During the life of the connection, the policy server has to make sure that the quality of service of the connection is maintained. The load of the policy server depends on the total number of flows that are accepted by the network.

The policy server and the different brokers are each modeled by a queue. Let the current number of connections that the policy server is in charge of be c . The life of a connection is depicted in Figure 3.

A user that wants to establish a connection to a destination user issues a set-up request. This permits the user to negotiate via the policy server a quality of service (QoS) between the two ends of the connection. When the set-up request is accepted, an ACK message is sent to the user. From time to time a control packet is transmitted to the policy server for maintenance or to request a change in the policy which may result to a new configuration in the routers. Therefore, during the duration of a connection, the policy server is busy with the management of the connection.

We assume that c connections can be managed simultaneously and we would like to determine the maximum value for c , that is, the maximum number of requests that may circulate in the policy server and its different brokers. Depending on the

implementation of the protocol, different kinds of management of the broker could be taken into account. As a first approach, we assume that the requests have to be served following a well-defined path, visiting first the policy server, then the different brokers in series. All the requests follow the same path. One possible model is given in Figure 4, where N is the total number of brokers. The requests ask for services which are exponentially distributed with parameters $\gamma_1, \gamma_2, \dots, \gamma_N$. The first queue represents the policy server. This queue is composed of c parallel independent servers. Each server corresponds to a connection that is under control of the policy server. When c connections have been accepted the number of server of the policy queue is c . A set-up request is automatically accepted, which corresponds to one more server in the policy queue. When a customer enters the policy-server queue, the server will be busy until the end of the lifetime of the connection. This lifetime is decomposed in service times and

two types of waiting times. Indeed, the service time of the policy-server corresponds to the treatment of the request. When a customer leaves the policy server, he enters the path going through the N brokers. Then, at the end of the last broker it enters with probability $1-p$ into a waiting time queue that represents the time that the connection has to wait before the next control action.

Let μ^{-1} be the mean service time of the policy server. Let α^{-1} the time the server waits for a new control demand concerning the same connection. After the passage through the N brokers, with probability p the connection is released. If a new cycle is necessary, this corresponds to a new control action.

The quantity N is the number of brokers the demand has to go through. We assume that the service times requested in the different servers are exponentially distributed with parameters $\gamma_1, \gamma_2, \dots, \gamma_N$ depending on the type of service the broker has to run. The bottleneck of the system can be either the policy server or one of the brokers.

Indeed, there is no queue in the policy server and the waiting time for a new action since the number of servers may become infinite. The model is a BCMP model and may be solved as a product form solution. However, in this paper we are interested in the maximum number of connections that such a system may support. If we assume that the system is running under a finite value of c that we can obtain by replacing a customer leaving the system by a customer entering the system, the model may be replaced by a new one, where input and output were removed.

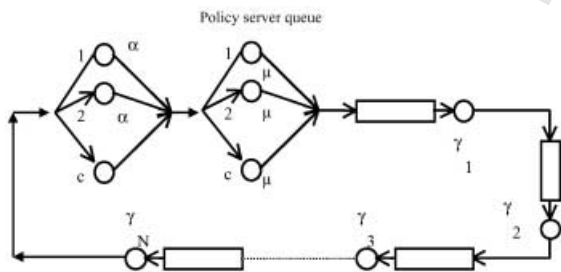


Figure 4. The model of a policy server

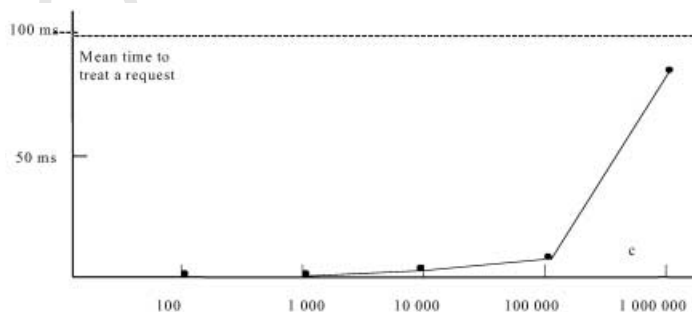


Figure 5. Maximum throughput vs c , for $N = 3$

As the number of customers is constant and equal to c , this model is a BCMP model that can be solved as a product form solution. Then, it is easy to compute the mean time to get an answer to a request or an intermediate control to the policy server. This time corresponds to go through the policy server queue and the different brokers. To be conformed to the classical performance required by a signaling request, this mean response time has to be under 100ms. This corresponds also to a real time process that should be supported by the policy server.

The mean response time of the system depends upon the number c and the number of brokers. If we assume $\gamma_i = 10^{-6}$, $i = 1, 2, 3, \dots, N$, we obtain the results depicted in Figure 5.

The result is not strongly dependent on N . It depends on the service rate of the brokers. When the rate is very high, it is possible that more than one million connections can be handled simultaneously. This gives an idea of the size of a domain.

Conclusions

We believe that in the future the enforcement points could not be edge routers except complicating the way to enforce the policy on these machines.

To get a direct negotiation with the PDP, a draft proposal has been issued¹⁰ proposing to use a COPS protocol for supporting a Service Level Specification negotiation. This solution provides the network with a flexible protocol that may

support multiple client-types. The COPS-SLS protocol needs only the corresponding new client-type (COPS-SLS). The client-handle object defined by COPS protocol gives a mechanism for handling various requests in a single PEP. This capability will be used to handle several SLS negotiations from a single PEP.

Finally, we have also proposed¹¹ to place a PEP in the smart card that will be included in mobile terminals.

References

1. RFC 2748—The COPS (Common Open Policy Service) Protocol, January 2000.
2. RFC 2749—COPS usage for RSVP, January 2000.
3. RFC 2750—RSVP Extensions for Policy Control, January 2000.
4. RFC 2751—Signaled Preemption Priority Policy Element, January 2000.
5. RFC 2752—Identity Representation for RSVP, January 2000.
6. RFC 2753—A Framework for Policy-based Admission Control, January 2000.
7. RFC 2872—Application and Sub Application Identity Policy Element for Use with RSVP, April 2000.
8. RFC 3084—COPS Usage for Policy Provisioning (COPS-PR), March 2001.
9. Verma DC. Policy-based Networking: Architecture and Algorithms: New Riders; 2001; 19–23.
10. Nguyen TMT, Boukhatem N, El Mghazli Y, Pujolle G. COPS Usage for SLS negotiation (COPS-SLS), <draft-nguyen-rap-cops-sls-01.txt>, November 2001.
11. Urien P, Pujolle G, IP benefits for smartcards use in wireless networks. ETSI, July 2001. ■