PROTOCOLS FOR COMMUNICATION BETWEEN QOS AGENTS: COPS AND SDP

Daniel Zinca[1], Virgil Dobrota[1], Cristian-Mihai Vancea[1], Gabriel Lazar[1]
Department of Communications
Technical University of Cluj-Napoca
Cluj-Napoca, Romania

Abstract – One solution to ensure QoS needed by multimedia applications is to start the requests directly from the host. In this case, a QoS management architecture can be used to avoid that applications are requesting parameters that cannot be sustained by the network. This paper considers two protocols that can be used for the communication between the QoS agents and a QoS management tool in a architecture we developed. The first protocol is COPS that was designed to exchange policy information, including QoS parameters, either the IntServ or DiffServ scheme. The second protocol is SDP, designed for exchanging multimedia session information between participants. The element of novelty in this paper is that we defined several modifiers that enable the use of SDP in combination with SIP for exchanging QoS policy information between QoS agents and the management tool. We investigate the two protocols regarding the suitability to the task and we found that using SDP with the proposed extensions is a feasible solution that has several advantages over the solution of using COPS.

I. INTRODUCTION

Different multimedia applications need different QoS (Quality of Service) parameters. IETF defined two categories for implementing QoS: IntServ (Integrated Services) [4] and DiffServ (Differentiated Services) [5]. Depending on how QoS is implemented by different applications, there are three situations: no QoS at all, IntServ (for example Microsoft NetMeeting) or DiffServ. If a multimedia application is QoS enabled, it can make requests to the network. One disadvantage of using QoS enabled applications is that the same type of multimedia application can implement different QoS techniques if produced by different manufacturers and the network does not necessary implement that type of QoS. Other disadvantage is that applications tend to request for example bandwidth as much as possible, without taking into account of other applications running on the same network.

In order to eliminate these disadvantages we proposed the following architecture [8], discussed in details within Section II. It consists of a QoS management tool running on a neutral station belonging to the Internet Service Provider and of QoS agents enabled on each terminal that run multimedia applications.

The protocol used for communication between the QoS agents and the QoS management tool is of great importance. In this paper we compare two existing protocols regarding the suitability to the task. The first protocol is COPS *(Common Open Policy Service)* [1] and the second one is SDP *(Session Description Protocol)* [2] used in combination with SIP [3]. COPS was designed as a management protocol in order to exchange information between policy servers and networking equipment. It has specific extensions for IntServ and DiffServ.

[1] *Technical University of Cluj-Napoca, Department of Communications, 26-28 Baritiu Street, 3400 Cluj-Napoca, Romania, Phones: +40-264-195699/ext.208, +40-264-413038, Fax: +40-264-197083, E-mail:{Daniel.Zinca, Virgil.Dobrota, Mihai.Vancea}@com.utcluj.ro, gabi_l@email.ro*

We found that the client side of this protocol can be implemented by the QoS agent. We used Intel's COPS SDK [9] for developing client applications. In our architecture, the network node (COPS specification) is QoS agent and the policy server (COPS specification) is QoS tool. Section III discusses this protocol and the results we obtained.

The second approach is our proposal of adapting SDP to be used. There are several application-defined modifiers in SDP and Section IV discusses some of them. We developed a SIP/SDP parser and we included into the VoIP (*Voice over IP*) application.

Section V draws the conclusions of the work done regarding the use of these two protocols for exchanging QoS information between QoS agents and the management tool.

II. QOS MANAGEMENT TOOL ARCHITECTURE

We proposed a QoS management architecture consisting of QoS agents running on end stations and QoS Management tool running on a neutral station (see Figure 1).

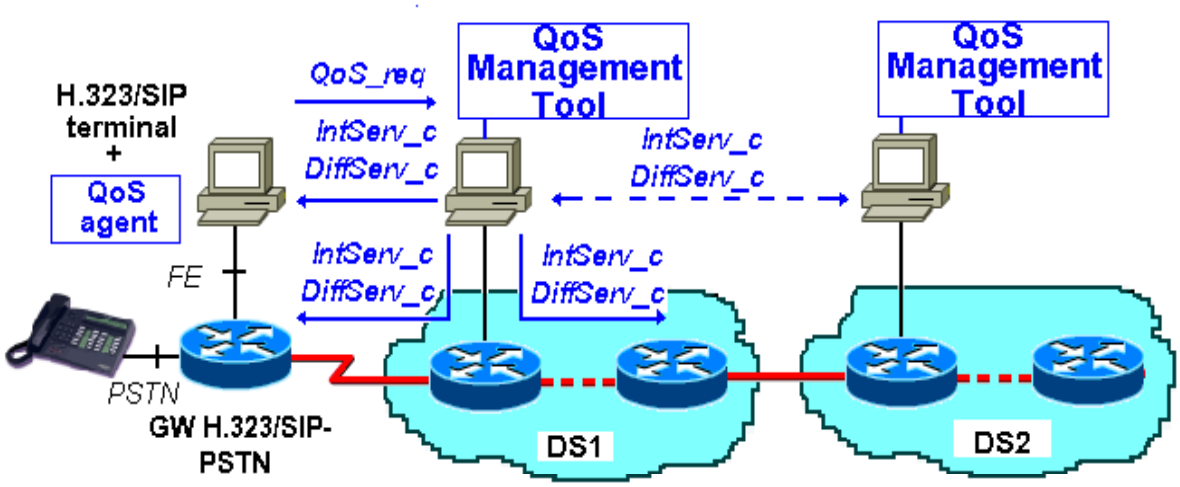

Figure 1. The QoS Management tool and QoS agents [8]

The QoS agents are sending *QoS_req* to the management tool in order to obtain the proper scheme to be used. Depending on the application type and specific QoS requirements, the QoS tool determines the proper 'treatment', consisting of IntServ/DiffServ configuration of routers in the Autonomous Domain. Also, the QoS agents receive the session parameters (if using IntServ) or DSCP value (if using DiffServ). The QoS agent is responsible of changing the scheme used on the terminal. Depending on the type of multimedia application used, the QoS agent actions can be different regarding the detection of the multimedia application characteristics and requirements.

If the QoS Agent must implement RSVP, it will use Microsoft Generic QoS API (*GQoS*), an extension to Windows Sockets. On the other hand, if the QoS Agent must implement DiffServ, a NDIS (*Network Driver Interface Specification*) Intermediate Driver is used, that marks certain packets sent over one network interface of the terminal according to predefined rules, for example the destination IP address, source/destination port, DSCP (*DiffServ CodePoint*) value. Either implementation is independent of multimedia application used.

Figure 2 presents the QoS agent actions in the case of a VoIP application.

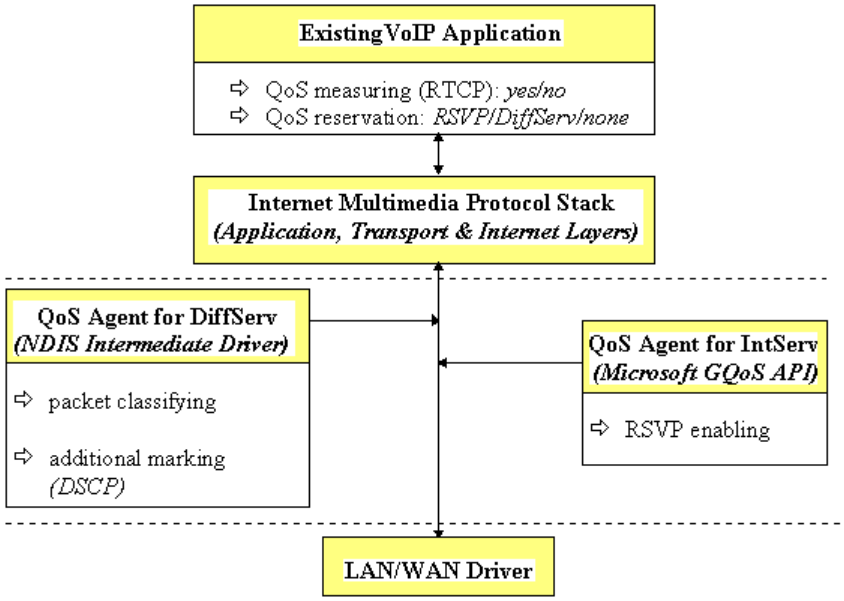

Figure 2. QoS agent actions

III. USING COPS AS THE COMMUNICATION PROTOCOL BETWEEN QOS AGENTS

The first choice for the communication protocol between QoS agents and the management tool we developed was COPS. Although the clients are usually embedded in routers, we found that these also can be implemented in QoS agents. For testing the suitability of the solution proposed, we used Intel COPS SDK.

The policy server, i.e. the PDP-Policy Decision Point in the COPS specification, can be either embedded in the QoS management tool, either the tool could dynamically change the PDP rules. The clients, i.e. the PEP-Policy Enforcement Points, are embedded in the QoS agents. The communication between PDP and PEP is initiated by the agent (by sending a *Client-Open* message) when it determines that a multimedia application is running. The policy configuration is determined by the management tool and is sent by the PDP (the *Client-Accept* message of *Client-Close* if a QoS scheme cannot be implemented in a particular case). If the network conditions are changing and the management tool determines that the QoS scheme must be modified, the PDP can send the new policy to the agent, in order to replace the previous one. The agent must send *Keep-Alive* messages to the PDP. The transport protocol used is TCP and the port number on which the policy server listens is 3288 [1]. The location of the management tool (IP address) is configured in the agent.

The software implementation of the PEP can be done using socket calls. The source code in the COPS SDK is an example of such implementation (see the architecture presented in Figure 3).

The Portability Layer allows the extension of the SDK to other operating environments in addition to Microsoft Windows family, for example to Linux or Solaris. The base COPS extension performs the interaction between the client and the server according to COPS specification. The RSVP extension provide the mechanism to exchange RSVP policy control decisions between clients and the Policy Server. The COPS-PR (COPS for Policy Provisioning) extension is used to communicate PIBs between PDPs and PEPs. The DiffServ extensions enable the support of QoS differentiation between aggregated flows, in order to classify and mark the traffic entering the network.

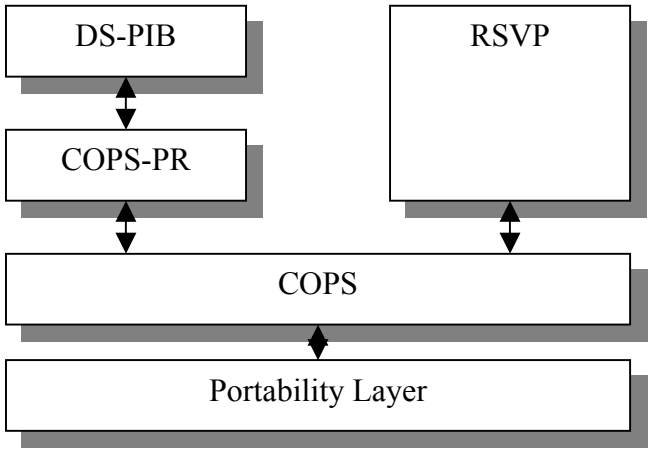

Figure 3. Intel COPS Client SDK architecture [9]

The RSVP extension provide the mechanism to exchange RSVP policy control decisions between clients and the Policy Server. The COPS-PR (*COPS for Policy Provisioning*) extension is used to communicate PIBs between PDPs and PEPs. The DiffServ extensions enable the support of QoS differentiation between aggregated flows, in order to classify and mark the traffic entering the network.

We implemented the PEP in the agent and we used the PDP binary distributed with the SDK. As the development tool we used Microsoft Visual C++ 6.0. Figure 4 presents a COPS *Client-Open* message sent by the agent and captured by Ethereal software package.

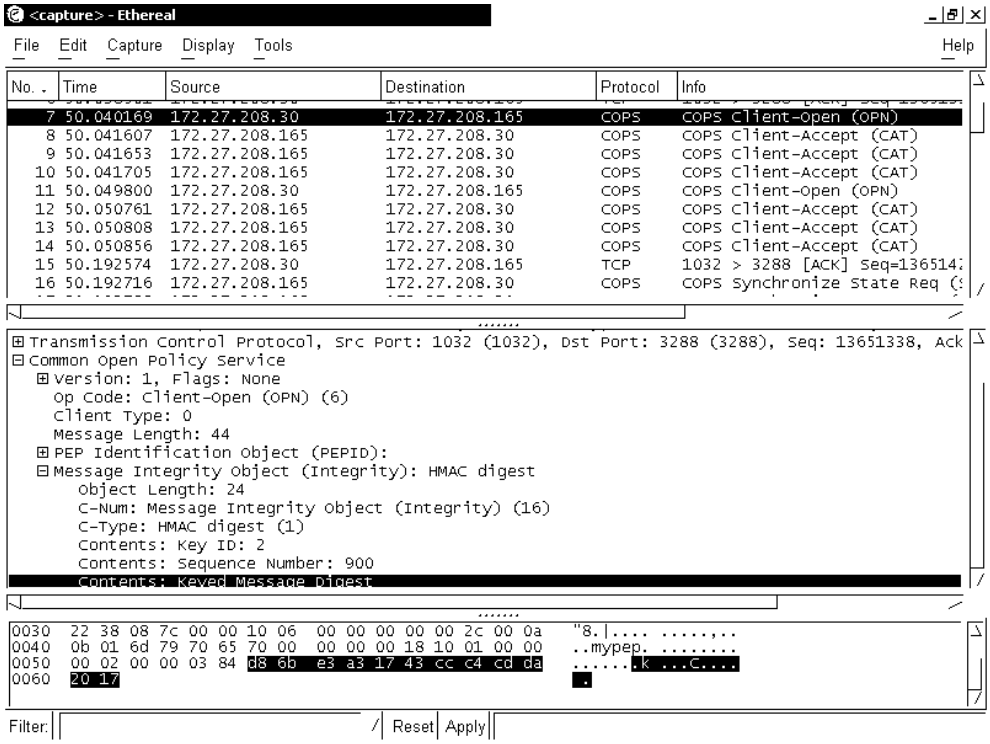

Figure 4. COPS *Client-Open* message

In Figure 4, the PDP IP address is 172.27.208.165, the QoS agent address is 172.27.208.30 and the *Client-type* is RSVP. The agent must send requests for both clients, RSVP and DiffServ respectively.

IV. USING SDP AS THE COMMUNICATION PROTOCOL BETWEEN QOS AGENTS

In this paper we propose the SDP protocol to be used for communication between QoS agents. SDP was developed in order to communicate the relevant conference setup information (the session description) to participants in a multimedia conference [2]. SDP can use different transport protocols, SIP being one of the most used. Different multimedia applications are implementing SIP/SDP parsers. Although there is no QoS-related information, in [2] is stated that the application can define several modifiers for specific purposes. One example is [10] where a RTCP (RTP Control Protocol) attribute is introduced.

We propose to use the *b= (bandwidth)* modifier to communicate specific information regarding the QoS parameters. For example, for communicating the bandwidth requirements, the *b=AT:<bandwidth requirement in kbps>* modifier will be used, because the value is application-specific. The *b=X-D:<delay in miliseconds>* is needed for exchanging the delay requirements. In case of DSCP value, *b=X-DSCP:<DSCP value>* specificator have to be used, whilst the scheme information may envolve *b=X-S:<INTSERV or DIFFSERV>*. According to SDP specifications, existing SDP parsers will ignore these fields, but the new ones can use them to exchange the QoS information. Figure 5 presents an example of a SDP message including the proposed extensions for DiffServ scheme. The message is sent by the QoS management tool after determining the proper treatment, and is encapsulated in the body of a SIP INVITE message.

```
INVITE sip:Agent1@172.27.208.161 SIP/2.0
Via: SIP/2.0/UDP 172.27.208.165
From: tool <sip:tool@172.27.208.165>
To: Agent1 <sip:Agent1@172.27.208.161>
Call-ID: 4037248562@172.27.208.165
CSeq: 1 INVITE
Content-type: application/sdp
Content-Length: 136

v=0
o=tool 2345 3345 IN IP4 172.27.208.165
s=QoSManagementTool
c=IN IP4 172.27.208.165
m=audio 2410 RTP/AVP 0
b=X-S:DIFFSERV
b=X-DSCP:46
```

Figure 5. Structure of a SIP/SDP message for exchanging DiffServ information

It is also possible to send an INVITE from the QoS agent containing no specificators and to have the management tool to send the SIP OK message containing the QoS specificators.

V. CONCLUSIONS

Our paper investigates two protocols to be used for communication between QoS agents in the QoS management architecture we proposed in [8], COPS and SDP respectively. The element of novelty is the use of optional fields in SDP to fit the purpose of communicating QoS parameters.

Several issues must be addresses in order to make SDP a complete solution for exchanging QoS information and we intend to continue the investigations in order to find solutions. Of great importance is the security issue, that can be solved using IPSec to encapsulate the SIP/SDP message. There are also other QoS parameters that need to be added as modifiers.

The SDP solution has the following advantages over the COPS solution:

1. A SIP/SDP parser is already implemented in several multimedia applications and the support for the modifiers can be easily added

2. COPS is a separate protocol that runs over TCP. SDP/SIP can run over UDP therefore is no need to implement TCP in dedicated applications (for example in VoIP telephones) because UDP is simpler to implement than TCP and multimedia applications are using usually RTP over UDP.

References

[1] D. Durham et al, "The COPS (Common Open Policy Service) Protocol", *RFC 2748*, January 2000
[2] M. Handley et al, "SDP: Session Description Protocol", *RFC 2327*, April 1998
[3] J. Rosenberg et al, "SIP: Session Initiation Protocol", *RFC 3261*, June 2002
[4] R. Braden et al, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", *RFC 2205*, September 1997
[5] S. Blake et al, "An Architecture for Differentiated Services", *RFC 2475*, December 1998
[6] V. Fineberg "A Practical Architecture for Implementing End-to-End QoS in an IP Network", *IEEE Communications Magazine*, January 2002, pp. 122-130
[7] V. Dobrota, "Digital Networks in Telecommunications. Volume 3: OSI and TCP/IP", Mediamira Science Publishers, Cluj-Napoca 2002
[8] D. Zinca, V. Dobrota, C.M. Vancea, G. Lazar, "A Practical Evaluation of QoS for Voice over IP", *12th IEEE Workshop on Local and Metropolitan Area Networks LANMAN 2002*, Stockholm, Sweden, August 2002, pp.65-69
[9] Intel COPS SDK 3.1, *http://developer.intel.com*, March 2002
[10] C. Huitema, "RTCP attribute in SDP", <*draft-ietf-mmusic-sdp4nat-02.txt*>, Work in progress, February 2002