Buletinul Stiintific al Universitatii "Politehnica" din Timisoara

Seria ELECTRONICA si TELECOMUNICATII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 47(61), Fascicola 1-2, 2002

Developing QoS-aware Applications in LANs

Daniel Zinca[1], Virgil Dobrota[1], Cristian Mihai Vancea[1], Gabriel Lazar[1]

Abstract – One of the methods used towards End-to-end QoS (Quality of Service) is to develop QoS-aware applications. The purpose of this paper is to investigate how feasible this solution is with tools available today and to draw some conclusions from the testbed used.

Keywords: QoS, IntServ, RSVP, DiffServ

I. INTRODUCTION

In addition to data-only applications, LANs are now facing the convergence of video and voice applications. There is the need to ensure a minimum sustenable rate and low delay and delay variation for certain applications. One possible method for solving these requirements is overprovisioning. When overprovisioning is not possible, more efficient methods can be used, grouped in two main QoS categories, as defined by IETF:

Integrated Services (or IntServ)
Differentiated Services (or DiffServ).

There are two important issues towards the process of QoS implementation in today's LANs.
The first one is to ensure the support from the network infrastructure (routers running appropriate routing protocols, other network equipment like Layer 2 switches with IEEE 802.1D Class of Service).
The second is the development of QoS-aware applications and the need for a specification (QoS API) enabling the programmers to request specific parameters from the network. Several QoS API were developed, mainly as extensions of classical socket implementations. For the Linux environment, one of the proposed implementations of a QoS API is described in [11]. Another QoS API was proposed by Microsoft as an extension to Windows Sockets [9] running on Windows 2000 operating software. Support for QoS exists in MS Windows 98 and Windows XP but in a simpler implementation. This paper is focused on describing the steps for the development of QoS applications using Microsoft's implementation.

Generally there are three perspectives of Quality of Service: the application perspective, the network perspective and the business perspective[10].
From the application perspective, QoS represents the ability to serve an application without affecting its performance or functionality.
From the network perspective, QoS is represented by several parameters that can be influenced and measured: bandwidth, delay, jitter and packet loss.
From the business perspective, QoS represents the resources that can be guaranteed to each user.
This paper focuses on the application perspective, enabling the application/user to request specific values of QoS parameters .

The remainder of the paper is organized as follows:
Section II- *Developing a QoS application using Windows Sockets* describes the steps towards creating a QoS application, Section III- *Implementing the QoS solution,* describing the network testbed and commands implemented on the router and Section IV- *Conclusions* presenting observations based on the results for the tests performed.

II. DEVELOPING A QoS APPLICATION USING WINDOWS SOCKETS

The Microsoft implementation of QoS in Windows 2000 has three categories of components:
- Application-Driven QoS Components
- Network-Driven QoS Components
- Policy-Driven QOS Components

The QoS support in MS Windows 2000 is based on RSVP signalling and Admission Control Service (ACS) therefore implements IntServ. It also makes use of the IEEE 802.1D user_priority field. Also a policy-driven QoS is available. Multimedia applications make use of PATH respectively RESV RSVP messages. One example is Microsoft NetMeeting.

The RSVP SP (Service Provider) is available to the user by the means of QoS API, mainly used for creating a flow of data transmitted preferentially through the network. Therefore the preferred solution in Microsoft Windows 2000 is IntServ but is possible to use DiffServ by modifying the *Flowspec* structure.

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

In case of using DiffServ, the client application requests a certain value of DSCP (DiffServ CodePoint) to be inserted in the IPv4 or IPv6 header. The edge router performs several tasks after receiving the packets. These tasks are implemented usually by the TCB (Traffic Conditioner Block). The first component of TCB is the Classifier that identifies packets for assignment to classes. The second component is the Meter that checks commpliance to traffic parameters. The third component is the Marker that writes or rewrites the DSCP value. The Shaper/Dropper delays or drops packets, according to the profile, before sending data to the next router.

There are several queuing techiques that provide different levels of handling the classes of traffic and also there are several congestion avoidance algorithms. Among the most used queuing techniques in routers are: Weighted Fair Queuing (WFQ), Distributed Weighted Fair Queuing (DWFQ), Class Based Weighted Fair Queuing (CBWFQ), Low Latency Queuing (LLQ), Custom Queuing (CQ) and Priority Queuing (PQ). The most used congestion avoidance techniques are Random Early Detection (RED), Weighted Random Early Detection (WRED), Distributed Weighted Random Early Detection (DWRED) and Tail Drop.

LLQ provides strict priority queuing for CBWFQ for real-time traffic, for example RTP packets containing voice packets belonging to a VoIP conversation.

WRED drops packets based on Precedence or DSCP fields and usually is implemented in core routers.

Because the QoS requirements of the applications are different, we choose to implement a Voice over IP (VoIP) application that has requirements in terms of Bandwidth, Delay and Jitter.

We implemented a VoIP application that uses RTP over UDP for transmitting G.711, A-Law coded voice packets and uses SIP protocol for call-setup[6]. The majority of software-based VoIP applications are not using any QoS mechanism. Several applications are using IntServ (for example, Microsoft NetMeeting). The applications that are using SIP usually are not implementing QoS at the moment.

The necessary steps for creating a QoS application are described next.

First, on the local machine a protocol with QoS support must be identified, using the *WSAEnumProtocols()* socket call, searching for the flag XP1_QOS_SUPPORTED. Having the identifier of the service provider, the call to *WSASocket()* with the WSA_FLAG_OVERLAPPED flag set will return the socket that will be used on subsequent calls. The next function call is *bind()* that has no specific QoS options.

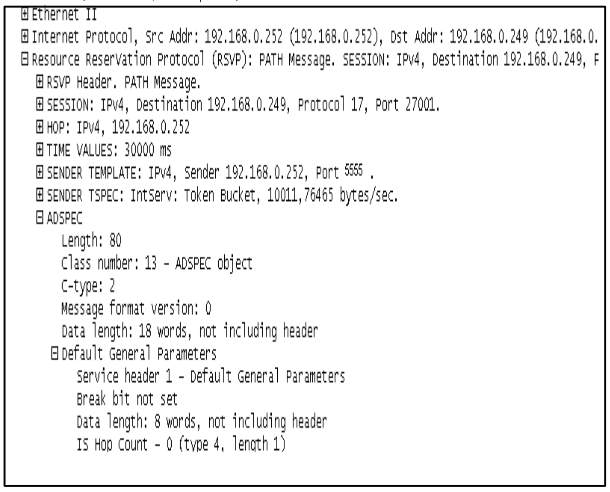
Next, a QoS structure must be filled, setting the *ReceivingFlowspec* and *SendingFlowspec* as desired. One option that can be used for the service type is SERVICETYPE_GUARANTEED.

The *WSAIoctl()* call with the SIO_SET_QOS will cause the RSVP packets to flow between the source and destination. Send and receive operations are performed using the classical *send()* or *sendto()* respectively *recv()* and *recvfrom()*. Fig. 1 presents the most important lines of code written in C language that are performing the operations described above.

```
LPWSAPROTOCOL_INFO Pro;
dword enumPro = WSAEnumProtocols (NULL,
NULL, &bufSize);
for(int i=0 ;i<enumProt ; Pro++, i++)
{
    if((Pro>dwServiceFlags1&XP1_QOS_SUPPO
RTED)&&(Pro->iAddressFamily==AF_INET))
    {
        break;
    }
}
SOCKET    QoSSocket    =    WSASocket(0,
SOCK_DGRAM,0,Pro,0,WSA_FLAG_OVERLAPPED);
QOS VoiceQ;
VoiceQ.SendingFlowspec.ServiceType
= SERVICETYPE_GUARANTEED;
VoiceQ.SendingFlowspec.TokenRate
= 8192;
VoiceQ.SendingFlowspec.TokenBucketSize
= 1000;
VoiceQ.SendingFlowspec.PeakBandwidth
= 65535;
dword ret;
WSAIoctl(QoSSocket,SIO_SET_QOS,&VoiceQ,s
izeof(VoiceQ),NULL,0,&ret, NULL,NULL);
```

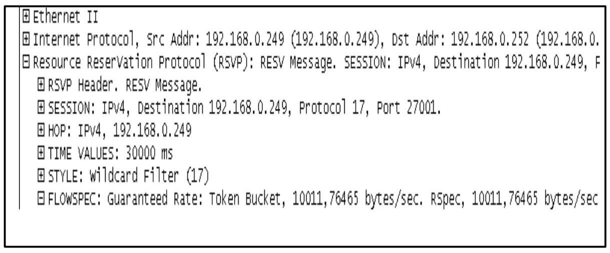
Figure 1. Source code using QoS calls

For tracing the flow of RSVP messages, Ethereal network monitor tool (http://www.ethereal.com) was used. The sending UDP client has the IPv4 address 192.168.0.252 and the receiving UDP client has the IPv4 address 192.168.0.249. Two of the captured RSVP packets are presented in Fig. 2 and Fig. 3 respectively. In Fig. 2 is presented a RSVP PATH message, containing the requested parameters. In Fig. 3 is presented a RSVP RESV message containing the accepted parameters, being sent from the receiver to the sender along the reverse data path. For closing the QoS flow, a RSVP PATH TEAR message is sent.



```
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: 192.168.0.252 (192.168.0.252), Dst Addr: 192.168.0.249 (192.168.0.
⊟ Resource ReserVation Protocol (RSVP): PATH Message. SESSION: IPv4, Destination 192.168.0.249, F
  ⊞ RSVP Header. PATH Message.
  ⊞ SESSION: IPv4, Destination 192.168.0.249, Protocol 17, Port 27001.
  ⊞ HOP: IPv4, 192.168.0.252
  ⊞ TIME VALUES: 30000 ms
  ⊞ SENDER TEMPLATE: IPv4, Sender 192.168.0.252, Port 5555 .
  ⊞ SENDER TSPEC: IntServ: Token Bucket, 10011,76465 bytes/sec.
  ⊟ ADSPEC
      Length: 80
      Class number: 13 - ADSPEC object
      C-type: 2
      Message format version: 0
      Data length: 18 words, not including header
    ⊟ Default General Parameters
        Service header 1 - Default General Parameters
        Break bit not set
        Data length: 8 words, not including header
        IS Hop Count - 0 (type 4, length 1)
```

Figure 2. RSVP PATH message captured



```
⊞ Ethernet II
⊞ Internet Protocol, Src Addr: 192.168.0.249 (192.168.0.249), Dst Addr: 192.168.0.252 (192.168.0.
⊟ Resource ReserVation Protocol (RSVP): RESV Message. SESSION: IPv4, Destination 192.168.0.249, F
   ⊞ RSVP Header. RESV Message.
   ⊞ SESSION: IPv4, Destination 192.168.0.249, Protocol 17, Port 27001.
   ⊞ HOP: IPv4, 192.168.0.249
   ⊞ TIME VALUES: 30000 ms
   ⊞ STYLE: Wildcard Filter (17)
   ⊟ FLOWSPEC: Guaranteed Rate: Token Bucket, 10011,76465 bytes/sec. RSpec, 10011,76465 bytes/sec
```

Figure 3. RSVP RESV message captured

Of particular interest is to add QoS support to existing applications. If IntServ is used, the support can be in a form of a separate module that starts the RSVP protocol, negotiating the necessary parameters on the sending and receiving ports of the applications, generating the RSVP PATH and RESV messages.
Using DiffServ on the client computer is more difficult because of the need to mark every packet sent by the selected application. We developed a solution that consists of installing a driver that captures all packets that are sent over one network interface and marks certain packets according to predefined rules, for example the destination IP address, source/destination port, DSCP value. We built a NDIS 5.0 Intermediate Driver, modifying the *NDISSend()* function call to replace the DSCP value at the corresponding offset, depending if the packet is an IPv4 or IPv6 one. NDIS Intermediate Driver acts as a Miniport Driver for Transport Protocols and as a Transport Protocol for Miniport Drivers.
If the packet is IPv4 and the network interface card belongs to the Ethernet family, the offset for DSCP/ Type of Service is 120 bits from the start of frame if the card is not using IEEE 802.1p and 152 bits from the start of frame if the card is using IEEE 802.1p/802.1Q header. If the packet is IPv6, the offset from the start of frame is 116 bits if the card is not using IEEE 802.1p and 148 bits if the card is using IEEE 802.1p. The Version field is the first of the IP header and is used to apply the DSCP at the proper offset [5].

III. IMPLEMENTING THE QoS SOLUTION

In addition to develop a QoS application there is the need to setup the network configuration.
The network testbed is presented in Fig. 4. The router (CISCO 1750) is used to test its RSVP interoperability with Microsoft implementation in Windows 2000. The Layer 2 switch (AT 8224XL) implements IEEE 802.1D, having CoS (Class of Service) enabled. The stations connected to the Fast Ethernet Hub are used for baseline reference (no CoS provided). In order to test the effect of the QoS settings, a traffic generator is used. The network configuration is "router on a stick". On the Win2000 workstations the QoS VoIP software is running and on the Cisco 1750 router different configurations were tested, IntServ as well as DiffServ. The router is a VoIP gateway and is also used as an terminal in VoIP calls.

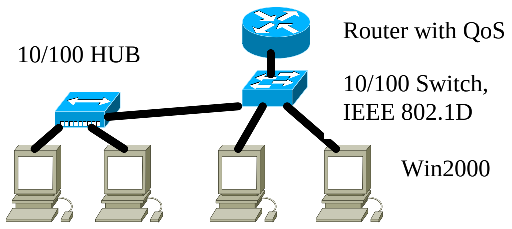


Figure 4. Testbed configuration

For DiffServ configuration of the router with DSCP=46, EF (Expedited Forwarding), WRED, the commands are presented in Fig. 5.

```
R1750(config)#class-map        match-all
premium
R1750(config-cmap)# match ip dscp 46
R1750(config-cmap)#exit
R1750(config)#policy-map VOIP
R1750(config-pmap)#class premium
R1750(config-pmap)#priority 500
R1750(config-pmap)#exit
R1750(config)#interface FastEthernet0
R1750(config-if)#random-detect dscp ef
R1750(config-if)#exit
R1750(config)#dial-peer voice 1 voip
R1750(config-dial-peer)#ip qos dscp ef
```

Figure 5. Router settings for DiffServ configuration

The commands for using IntServ on the router are presented in Fig. 6 (RSVP protocol, fair queue, Bandwidth=50 Mbps).

```
R1750(config)#call rsvp-sync
R1750(config)#interface FastEthernet 0
R1750(config-if)#fair-queue
R1750(config-if)#rsvp bandwidth 50000
50000
R1750(config-if)#exit
R1750(config)#dial-peer voice 770 voip
R1750(config-dial-peer)#destination-
pattern 0264
R1750(config-dial-peer)#session target
ipv4:193.226.6.171
R1750(config-dial-peer)#req-qos
guaranteed-delay
R1750(config-dial-peer)#acc-qos
guaranteed-delay
```

Figure 6. Router settings for IntServ configuration

In the first set of experiments the workstations had IPv4 addresses in the same network in order to determine the traffic where the quality of the voice call drops below the acceptable level. In the second set of experiments the workstations had IPv4 addresses in different networks, with the router linking the two networks in order to investigate the efficiency of the QoS settings made on the router configuration.

IV. CONCLUSIONS

A VoIP application was developed using Microsoft QoS API on Windows 2000 environment. A testbed was implemented and different QoS settings on the router were experimented.

The QoS method (IntServ or DiffServ) depends on the type of application. IntServ is considered to have scalability problems for large networks. For VoIP applications, a combination between DiffServ in the local domain and IntServ (RSVP messages between the endpoints) offers the best results.

If DiffServ is used for VoIP applications, the best results are obtained with DSCP set to EF (Expedited Forwarding), LLQ and WRED.
If IntServ is used, the best combination is with LLQ.

One possible solution in order to avoid user applications to request QoS that cannot be delivered by the network is using COPS (Common Open Policy Service) to obtain network traffic policy information from a central location in the DiffServ domain.

REFERENCES

[1] S. Blake et al, “An Architecture for Differentiated Services”, *RFC 2475*, December 1998
[2] D. Grossman, “New Terminology and Clarifications for Diffserv”, *RFC 3260*, April 2002
[3] J. Wroclawski, "The Use of RSVP with IETF Integrated Services” , *RFC 2210*, September 1997
[4] S. Shenker et al, "Specification of Guaranteed Quality of Service”, *RFC 2212*, September 1997
[5] V. Dobrota, “Retele digitale în Telecomunicatii., Volumul 3. OSI si TCP/IP”, Editura Mediamira Cluj-Napoca 2002
[6] V. Dobrota, D. Zinca, C. M. Vancea, G. Lazar, “Voice over IP Solutions for CAMAN: H.323 versus SIP”, *The First RoEduNet International Conference*, Cluj-Napoca, April 2002, pp. 37-43
[7] 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
[8] V. Fineberg “A Practical Architecture for Implementing End-to-End QoS in an IP Network”, *IEEE Communications Magazine*, January 2002, pp. 122-130
[9] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/qos/qosstart_2cdh.asp
[10] http://www.cisco.com/warp/public/732/Tech/qos/
[11]http://www.ittc.ukans.edu/~pramodh/courses/linux_qos/mainpage.html