S-38.180 QoS in the Internet - Fall 2002 / Exercise 3: Integrated Services

Mika Ilvesmäki
Helsinki University of Technology
Networking laboratory

September 30, 2002

Abstract

You will simulate two different CAC-algorithms and compare the network behavior to best effort -network. All simulation scripts are provided and new code production is kept to a minimum. You will write a report maximum of 15 pages (including figures) and submit it before the deadline.

Instructions

The exercise lecture is held on October 2nd in the computer classroom starting at 8.15am / 12.15pm.

- The exercise report of this exercise is due October 23rd, 4 pm. We strongly recommend you to return the exercise already by October 9th. Return is via email to lynx@tct.hut.fi or personally. If returned in electronic format, PDF-documents are appreciated. Postscript is acceptable, other formats are not recommended.

1 Introduction

In the current Internet, there are no guarantees for either relative or absolute QoS and it is debatable if we can ever expect the Internet to provide absolute end-to-end QoS [1, 2, 3, 4]. The contemporary Internet is characterized by the diversity of its networking technologies. In the core of the network ATM, which is able to offer QoS [5], is slowly pushing FDDI–solutions to the background. This trend would implicate that the core Internet could be able to offer some kind of service levels if the penetration of QoS capable technologies reaches an adequate level. However, in the edges of the network the multitude of network solutions is overwhelming. All the different LAN technologies, some capable to offer QoS and some not, create a substantial obstacle to an absolute end–to–end Quality of Service in the Internet. Furthermore, the ever growing diversity in access technologies, such as the introduction of xDSL techniques and the strong foundation of traditional PSTN–modem solutions suggests that offering a possibility of consistent QoS in the Internet would introduce a number of problems regarding, for instance, the definition of QoS parameters.

Quality, in the Internet frame of reference, could be understood as the combination of exactly defined measures such as data loss, delay, jitter and use of network resources associated with the feeling or notion of Quality that the user of the network experiences. Major difficulty lies in defining the Quality as a function of both the measures **and** the human factor.

In this light, the *Quality of Service* in general terms and when speaking of networking, means that the user of some service receives a predefined, but not necessarily a constant amount of resources from the network guaranteeing that the user's packets are delivered to their destination within the set parameters and performance bounds.

On a related issue, *Class of Service* (CoS) is a closely related concept to QoS. Using CoS instead of QoS means that the traffic of one user is treated better than the traffic of another. No absolute guarantees are given, only promise to differentiate traffic. The Class of Service solution will most probably be the concept first deployed in the Internet before the actual end-to-end absolute Quality of Service.

2 Integrated Services

The recent developments of software and the emerging new services with increasing commercial efforts suggest that QoS, or at least different levels of service, Class of Service (CoS), should be introduced to the Internet. The

emergence of various bandwidth hungry and delay sensitive applications, such as Voice over IP and video conferencing, require, or at least benefit from QoS or some other form of network performance guarantees. Similarly, the probable growth of new QoS sensitive applications [6] using the Internet protocol might expect some sort of QoS guarantees from the network. Several IETF[1] workgroups, such as IntServ[2] and DiffServ[3], have participated in the discussion and definition of Internet service architectures, but their work has not yet reached to any conclusive solutions. Various architectural and technological solutions have emerged and the heated debates for and against these solutions have dominated the discussion on the future Internet.

2.1 Integrated Services -architecture

The Integrated Services -architecture[4] proposal starts from the assumption that some traffic flows in the network need end-to-end Quality of Service guarantees [7] and that a relatively small number of flows ask for these guarantees (rate controlled applications) while the rest are satisfied with the normal best effort type of service (adaptive applications) [8]. To this end, the IntServ proposal uses the resource ReSerVation Protocol (RSVP). The IntServ has suggestions for several service classes but only two have been defined, in addition to the traditional best effort -'service':

1. Guaranteed service[5] that provides an assured level of bandwidth, a firm end-to-end delay bound and no queuing loss for the conforming packets of the traffic flow.

2. Controlled-load service[6] that provides no firm quantitative guarantees and tries to offer the flow a service level equivalent to lightly loaded best-effort network.

The concept of Integrated Services is essentially of the per-flow type. It is intended that all of the network elements that take part in the packet forwarding have knowledge of the flow, or connection, that the packet belongs to. This knowledge consists of information that is needed to produce deterministic network characteristics in terms of available bandwidth, delay, jitter and packet loss to the flow. In essence, the Integrated Services aims

[1] http://www.ietf.org/
[2] http://www.ietf.org/html.charters/intserv-charter.html
[3] http://www.ietf.org/html.charters/diffserv-charter.html
[4] RFCs 2205-2216 and RFC 1633
[5] RFC 2212
[6] RFC 2211

to provide Quality of Service to the selected flows. The implementation and realization of the Integrated Services in an IP router is open for vendors but a committed effort has been seen in realizing the RSVP -protocol.

2.2 RSVP - Resource reSerVation Protocol

The RSVP was designed to enable the senders, receivers, and routers of communication sessions to communicate with each other in order to set up the necessary router state to support the IntServ service classes. RSVP accommodates all kinds of connection types, including multicast, it uses soft state, and it is designed to be relatively easy to implement in the Internet routers [7, 9]. RSVP identifies a communication session by the combination of destination address, transport-layer protocol type and destination port number. The primary messages used by the RSVP are the *PATH* and the *RESV* message:

- The *PATH* message originates from the traffic sender. The primary roles of the *PATH* message are to install reverse route state in the routers along the path and to provide receivers knowledge about the sender traffic.
- The *RESV* message originates from the traffic receiver. The primary role of the *RESV* message is to carry resource reservation requests to the routers between the receiver and sender.

The RSVP supports three types of reservations [10]:

1. The Wildcard Filter reservation is aimed particularly for multicast connections and is shared with all senders and extended automatically to new senders as they join the path.
2. The Fixed Filter reservation of resources is distinct and the sender is specified explicitly.
3. The Shared Explicit reservation is particularly suitable for multicast connections and the reservation is shared by selected senders.

The work in the Integrated Services -architecture has been mostly done based on the assumption that it is the user who initiates the resource allocation process. The role of the network is then to calculate if the requested resources are available and either accept or reject the request. This calls for an admission control unit in all of the routers in the packet path.

It is quite evident that charging schemes are needed to protect an IntServ network from arbitrary resource reservations and to create a funding mechanism to extend network capacity at the most desired locations at the expense of those users that actually use these resources [11].

2.3 Discussion on Integrated Services

As the goals of the Integrated Services -approach are rather ambitious, it has also met a lot of criticism. The main issues of controversy and debate have been identified as:

1. **Bringing state into the Internet.** The traditional paradigm of Internet has been stateless and the discussion circulates mainly around whether to bring state to the Internet or not. In the traditional Internet, routers do not keep connection state information. This is to improve the robustness of the communication system and routers are designed to be stateless, forwarding each IP packet independently of other packets. Consequently, redundant paths can be exploited to provide robust service in spite of failures of intervening routers and networks. All state information required for end-to-end flow control and reliability is implemented in the hosts, in the transport layer or in application programs. All connection control information is thus co-located with the end points of the communication, so it will be lost only if an end point fails. With the introduction of the IntServ -scheme, the need to know of the state of the traffic flows is unavoidable. To guarantee deterministic performance on a flow, all the intermediate parties need to be aware of the requirements to provide such service.

2. **Complexity.** One of the main problems with any resource reservation technology is the burden of implementing complex systems needed for setting up and maintaining state information. Since the processors and other physical building blocks are becoming ever so fast, it has been argued that this aspect of complexity should not be considered as the key obstacle.

3. **Scalability.** The essential issue with the IntServ -scheme is the per-flow state scalability. While the number of simultaneous connections (and state table requirements) may be reasonable at the edges of the network the size of the state tables increases easily to intolerable levels in the core of the network. This effect is further enhanced with the recent trend in traffic patterns where 80% of the traffic is forwarded outside of the LANs.

4. "Flag day requirement". To work in a consistent manner the IntServ -functionality should be implemented throughout the packet path. This requires, in the case of IntServ, that the RSVP and related functionality should exist as well in the hosts at the edges as in the core routers. Essentially this means updating the whole of Internet to support RSVP functionality and this is not seen as a trivial task.

2.4 Summary

The current functionality in the Internet is capable of offering a fair sharing of resources using basically only the FIFO-method of queuing in the routers and advanced flow control in the TCP-protocol. It seems likely that the dominant position of the router will continue and it will be providing mechanisms for realizing Quality of Service also in the future Internet, although, bringing service differentiation or QoS to the Internet requires several changes in the router architectures and especially requires broadening the ways we think an Internet router should function.

Integrated Services -approach aims to provide end-to-end deterministic Quality of Service to the few selected users. While doing this it requires per-flow knowledge in throughout the network, which in turn introduces problems in the traditional Internet paradigm, issues in the complexity of the implementation and doubts in the capability to scale.

3 IntServ exercise with ns2

The support for IntServ in ns2 is very limited. The majority of functionality has been designed to evaluate admission control algorithms for the Controlled Load -service class [12] and observing the RSVP functionality. This exercise will totally concern on the former.

3.1 Exercise setting

You will be provided with complete topology, simulation and monitoring scripts. A short code walk-through is provided later on in this document and also in the document for the exercise on Differentiated Services. Some parameters need to be changed as you go through different admission control algorithms, but the changes occur only in a couple of lines and will be indicated in the exercise lecture. Furthermore, you will be provided with proper simulation files for each scenario. You begin your work by simulating the network in Best Effort -mode. This simulation will provide you a reference

point to which you can compare your further results. Most of the work you do will concentrate on analyzing the admission control algorithms and observing the effect the algorithms have on utilization of a link, the packet drop behavior and on delay behavior of voip-clients. You will also be provided means to extract some data out of the ns2-simulations. However, you are on your own regarding the calculations and visualizations of the simulation data. Use Matlab or something similar to plot the graphs and calculate the required data.

3.2 Admission control algorithms

On the details on the admission control algorithms, please refer to the appropriate exercise lecture and to material provided you in the course website.

3.3 Exercise questions

Write a report with maximum of 15 pages including figures, that has the following contents:

- Present *a comparison on the use of the bandwidth with different admission control algorithms*. Compare your results also to the Best Effort -case. The following results are required although you may present more if you think it will clarify your point of view:

    1. Graphic presentation on the use of bandwidth (you may use the xgraph-output when available). Note, that it may be rather difficult to get bandwidth usage plotted in the Best Effort -case. Therefore, it is not *required* to present the graph on the bandwidth usage with the BE-case.
    2. Calculate the average of the bandwidth use (a percentage value will be shown at the end of of the simulation in all scenarios, use temp.rands to confirm it).

- Present a comparison on the packet drops with different admission control algorithms. Compare your results with the Best Effort -case. At least the following results are required:

    1. What are the total packet drop rates in different scenarios (BE, IntServ with different CAC)?
    2. How do the dropped packets distribute themselves between different applications (two types of VoIP, http and ftp)? Elaborate.

3. Where (in the topology) do the packet drops occur?

- Study the delay behavior of the VoIP-clients.
  1. Present the starting times of different VoIP-clients. Indicate the differences between the two types of VoIP-clients. Elaborate on rejected VoIP-connections when available.
  2. Present the average delay and the delay variance for all of the VoIP-packets in one scenario. Plot also the delay distribution. Note that there might be some negative delays present. This is an error in the scripts, do not care and do not attempt to explain this.
  3. Present the average delay and the delay variance for two VoIP-clients, plot also the delay distribution for the same VoIP-clients.
  4. Compare the delay behavior with the best effort network and the IntServ-networks with the different admission control algorithms in use. Present your conclusions.
- Present a brief and concise in-depth analysis on your results.

3.4 How to get started?

1. Login to a workstation and enable the ns2-simulator version for this exercise.
2. Obtain the following tcl-files from

   http://www.tct.hut.fi/opetus/s38180/s02/exercises/H3/intserv.zip

   - intservnet[BE,HB,MS].tcl - this is the main file, the BE/HB/MS -extension indicates the simulation scenario.
   - topology[BE,HB,MS].tcl - this sets up the topology, the change in BE/HB/MS -extension indicates the use of different admission control algorithms or no algorithm at all
   - peer_setup.tcl - this sets up the traffic sources - you do not need to change anything in this file!
   - monitoring.tcl - here are some routines to get the measurement data - you do not need to change anything in this file!
3. Start the simulation by typing *ns2 intservnet[BE,HB,MS].tcl*, choose the BE/HB/MS -extension as appropriate. Note, that the different simulations produce result files with similar names. Therefore, if you need to run the simulations simultaneously, do so in separate directories.

4. The simulation will run around five minutes, you'll see lots of info rolling on the screen. Above all, this will ensure that the simulation is proceeding.

5. After the simulation is done (300s) you'll see a lot of output files produced: The ones useful to you in this exercise are the *mon_bottle.mon*, *temp.rands*. You can use the data in them to calculate the bandwidth utilization.

6. Familiarize yourself with the *tr_intservnet.out*.

   You may find http://nile.wpi.edu/NS/analysis.html useful.

7. Do the post processing on the *tr_intservnet.out* as indicated at the end of *intservnet[BE,HB,MS].tcl* -file. You should end up with *delay.out* file that you can access for all packet delays, and delays for particular src-dst pair. The columns are src dst packetid and delay.

8. Post-process the *tr_intservnet.out* as indicated in the *intservnet[BE,HB,MS].tcl* again to get the data on packet drops.

9. Use, for instance, matlab to plot and calculate the required results. Answer the exercise questions.

3.5 Brief code walkthrough - *intservnet[BE,HB,MS].tcl*

First of all, the scripts provided to you are somewhat commented. Look into them. By now, you should be relatively familiar in reading ns2-code, although, you might still be hesitant to write any. Second, the ns2 code in the IntServ exercise is a derivative of the DiffServ-code that you'll go through later on in this course. If you have not already done so, you may want to familiarize yourself with the code walkthrough in that documentation. The *intservnet[BE,HB,MS].tcl* files contain the simulation script for Best Effort-, Hoeffding bound- ,and Measured Sum- admission control algorithms. The same applies for *topology[BE,HB,MS].tcl* files.

3.5.1 Setting up the environment

There is some IntServ -specific code first and then we start by creating the simulator:

```
set ns [new Simulator]

 # Simulated seconds and random seed
```

```
 # to be used (0 := random seed for every run)
set testTime 300.0
$defaultRNG seed 0

 # Activate tracing for all links
 # (potentially very slow and disk-space-consuming)
 # In IntServ this enables for delay calculations
 # Output file: tr_intservnet.out
set trace_all true
```

The *tr_intservnet.out.gz* will be a large (zipped) file containing information on each and every packet that traverses through the network. We will need this file for obtaining delay information on the voip-clients and the packet drop information. Do check that the *trace_all* variable is set to true. Files that are zipped with *gzip* may be viewed with *gzcat*

Next we set up the network topology and the clients and nodes. The original topology is also shown in Figure 1 from [13]. In this exercise the topology is slightly altered by doubling every VoIP-client and setting the additional VoIP-client packet size to 201 bytes. This will help you to distinguish between the two VoIP-client groups.

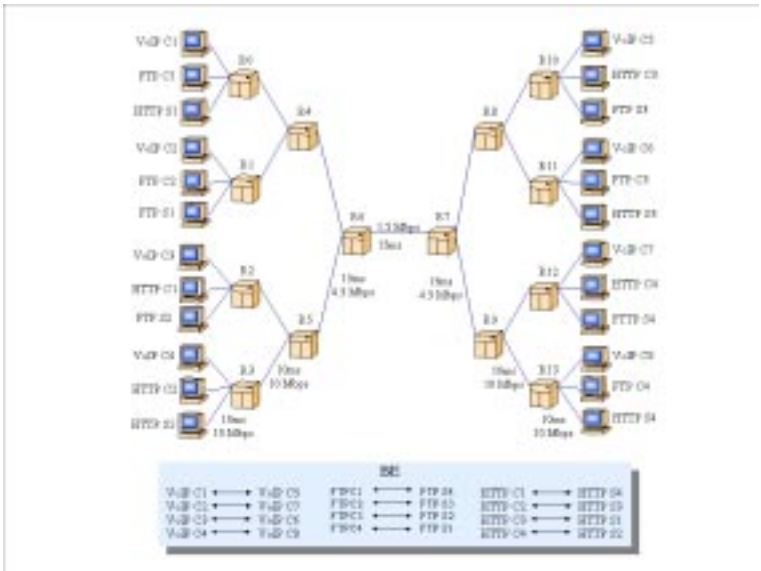

Figure 1: Topology used in IntServ -exercise [13] – Note: The bottleneck link is an IntServ-link in this exercise

```
 # Get functions for creating HTTP, FTP and VoIP traffic
```

```
source peer_setup.tcl

 # Create topology
source topology[BE,HB,MS].tcl
```

Familiarize yourself with traffic characteristics and the way in which the topology is created. Note, how you may change the network to a completely Best Effort -network.

3.5.2 Changing the admission control algorithms - *topology[BE,HB,MS].tcl*

In *topology[BE,HB,MS].tcl* you'll find the lines defining the admission control algorithms to use.

```
# MS (measured sum) and HB (Hoeffding bounds)
# as admission control ADC.
# EST determines the estimation method,
# MS uses TimeWindow, and HB uses ExpAvg
set ADC HB
set EST ExpAvg
```

In the exercise you're required to evaluate the MS (Measured Sum) and HB -algorithms. Do not use any other algorithms. Specify also the estimation method as stated in the code comments. Do not use any other estimation methods than indicated.

3.5.3 Setting up the measurements - *intservnet[BE,HB,MS].tcl*

Here we record the bytes that are sent through the bottleneck link. This piece of code is borrowed as is from the test suite for IntServ in ns2.

```
# IntServ link util measurements
    set qf2 [open "mon_bottle.mon" w]
    set qmon [$ns monitor-queue $n(r6) $n(r7) $qf2]
    set l67 [$ns link $n(r6) $n(r7)]
    $l67 set qBytesEstimate_ 0
    $l67 set qPktsEstimate_ 0
    $l67 set sampleInterval_ 0
# Show results after the measurement period,
# comment out if not using int-serv on the link
    $ns at @meastime "$l67 trace-util 0.9 $qf2"
```

The next lines will give you an xgraph -produced figure that shows the estimated and actual bandwidth usage. You may use the figure to your advantage as you wish. The code is also copy-pasted from the test suite of IntServ in the ns2.

```
$qmon instvar pdrops_ pdepartures_ bdepartures_
    set utlzn [expr $bdepartures_*8.0/(1.5e6*($simtime-$meastime))]
    set d [expr $pdrops_ /($pdrops_ + $pdepartures_)]
    puts "Packet Drops : $d Utilization : $utlzn"
    if [ info exists qf2 ] {
        close $qf2
    }
    set output [ open temp.rands w ]
    puts $output "TitleText: $file"
        puts $qf2 "Device: Postscript"

    exec rm -f temp.p1 temp.p2
        exec awk {{print $1,$2}} mon_bottle.mon > temp.p1
        exec awk {{print $1,$3}} mon_bottle.mon > temp.p2
    puts $output [format "\n\"Estimate"]
    exec cat temp.p1 >@ $output
    puts $output [format "\n\"Actual Utilzn"]
        exec cat temp.p2 >@ $output
        exec xgraph -bb -tk -m -x time -y bandwidth -lx 1,1,300 temp.rands &
```

You can use the commands on the following lines to post-process the tracefile and provide you with information on the delays between different voip-clients and packet drops. You'll have to enter these by copy-paste (suggested) or by hand. Please note, that if the .out-file is zipped you need to unzip it with *gzcat*.

```
 # Finish up the simulation
proc finish {} {
...
# Start producing the delay data, you'll have to enter these by hand
    #cat tr_intservnet.out | grep "exp 200" | grep "r" > voip_received.out
        # exp 201 if you're looking for the reference sources
        # cat may also be gzcat if .out file has been zipped.
        # The r means for packets received, useful in determining the delays.
        # The d collects the dropped packets.
```

```
        # You could use wc -l to determine the packet count when searching for d
        # For instance cat tr_intservnet.out | grep "exp 201" | grep "d" | wc -l
        # would give you the number of dropped packets for the reference voip-cl
        # Now let's start determining the delays.
    #rm tr_intservnet.out, do not if you still need to analyze the dropped packe
        # Let's modify the output and get timestamp src dst pkt#
    #sed 's/[.]/ /2' voip_received.out | sed 's/[.]/ /2' |
    # awk '$3==$9 || $4==$11 {print $2,$9,$11,$14}' > voip_mod.out
    #sort -k 4,4 voip_mod.out -o voip_sorted.out
    #rm voip_mod.out voip_received.out
    #cat voip_sorted.out | awk '($2==old2 && $3==old3 && $4==old4)
    #{printf("%d\t%d\t%d\t%f\n",$2,$3,$4,($1-old1))}
    #{old1=$1;old2=$2;old3=$3;old4=$4}'>delay_voip.out
    #rm voip_sorted.out
        # ok, now we have end-to-end (well, almost) delays for src,dst,pkt#
        # use cat delay_voip.out | awk '{print $4}' to get all the delays,
        # this might be useful in examining the overall performance
        # use cat delay_voip.out | awk '($1==src && $2==dst){print $4}'
        # to get the delays for particular src,dst pair, then input
        # the delays into matlab etc. to get the delay
        # distributions, averages, variances, jitter and what have you
...
}
```

Acknowledgements

Thanks to Marko for inventing the topology, to Timo for implementing the topology and stuff to ns and also thanks for bearing with my questions. Thanks to Esa, Markus and Jouni for showing me the ways of sed and awk and telling me what the cat does in unix-systems.

References

[1] M. Borden, E. Crawley, B. Davie, and S. Batsell. *Integration of Real-Time Services in an IP-ATM Network Architecture.* August 1995.

[2] S. Shenker, C. Partridge, and Guerin R. *Specification of Guaranteed Quality of Service.* February 1997.

[3] Eric Crawley, Raj Nair, Bala Rajagopalan, and Hal Sandick. *A Framework for QoS-based Routing in the Internet*. March 1996.

[4] Bala Rajagopalan and Raj Nair. *Quality of Service (QoS) -based Routing in the Internet - Some Issues*. October 1996.

[5] *Traffic Management Specification Version 4.0*. ATM FORUM, April 1996.

[6] Eugenio Guarene, Paolo Fasano, and Vinicio Vercellone. IP and ATM integration perspectives. *IEEE Communications Magazine*, January 1998.

[7] Paul P. White and Jon Crowcroft. The integrated services in the internet: State of the art. Technical report, University College London, 1996.

[8] Alistair Croll and Eric Packman. *Managing Bandwidth - Deploying QoS in Enterprise Networks*. Internet Infrastructure Series. Prentice-Hall, 2000.

[9] Paul P. White. RSVP and integrated services in the internet: A tutorial. *IEEE Communications Magazine*, 35(5):100–106, May 1997.

[10] Paul Ferguson and Geoff Huston. *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*. John Wiley and Sons, 1998.

[11] Martin Karsten, Jens Schmitt, Lars Wolf, and Ralf Steinmetz. Cost and price calculation for internet integrated services. 1998.

[12] Sugih Jamin, Peter B. Danzig, Scott J. Shenker, and Lixia Zhang. A measurement-based admission control algorithm for integrated services packet networks (extended version). *IEEE/ACM Transactions on Networking*, February 1997.

[13] Marko Luoma. Simulation studies of differentiated services networks. *Licentiate thesis*, Helsinki University of Technology, 2000.