A course on Quality of Service (QoS)

Jaume Barcelo

October 26, 2012

Contents

Chapter 1

About the course

1.1 Course Data

Code: 21738

Course name: "Protocols de qualitat de servei en xarxes"

Teacher: Jaume Barcelo

Credits: 4

Year: 3rd year

Trimester: Spring

1.2 Introduction

This is a course on Quality of Service in data networks, which is usually abbreviated as QoS. QoS is about discriminating traffic. It is about favouring some data packets at the expense of others. The name can be misleading, as one might think that all packets are benefited from the implementation of QoS. This is not the case.

A good parallelism to understand QoS is road traffic. There

are some vehicles (typically police, ambulance and firefighters) that receive priority over the others. This is not perceived as something negative, as this vehicles incur in tasks that are more important or urgent than the average vehicle.

This parallelism is very illustrative to convey the idea, that QoS does not make the road wider, it simply prioritizes some traffic.

At some point, networks engineers may face the dilemma of investing their efforts and money in either implementing QoS (prioritizing traffic) or increasing the available bandwidth (making the road wider). The latter option has the advantage that it benefits all the traffic of the network. Ideally, if the bandwidth is sufficiently over-provisioned, the packets never have to wait in the routers queues. In the road analogy, if the roads are wide enough, there are never traffic jams.

Unfortunately, making the roads wider or the networks faster does not always solve the problem. As the users perceive that there is plenty of bandwidth available, they might decide to put it a better use by downloading collections of movies that they will never have time to see. There is nothing wrong with downloading collections of movies, but the sheer volumes of data may fill up the queues and introduce unacceptable delay and jitter in VoIP calls.

And why is this a problem? Well, at some point network engineers that deploying separate networks for each service represented too much work (and money). From an engineering and economic point of view, it is much more advantageous to offer the different services on a single network. It is common nowadays that telephony, video-conference, web, remote backup and file sharing services share the same network. The term to refer to these networks that support various services is "converged networks". The only problem is that the different services have totally different requirements with regards to required bandwidth and delay.

The service that consume a large amount of bandwidth and do not have strict delay requirements can easily create a "traffic jam" that prevents the offering of services with low delay requirements that consume very little amount of bandwidth. Obviously, it is still possible to offer the two kinds of service simultaneously if we implement QoS mechanisms that prioritizes the low delay traffic.

QoS is a controversial topic. Net neutrality, which is related to QoS is even more controversial [1]. Nevertheless, QoS (or "bandwidth management") has been used by ISPs in practice [2]. We will try to cover the subject from a neutral point of view and you, equipped with the knowledge of the course, will take your own decision about the usefulness of QoS.

The course is divided in three conceptually different parts: lectures, seminars and lab assignments. In lectures I will introduce you to the nuts and bolts of QoS. In the seminars, we will review the concepts of queueing theory covered in previous courses, and extend them to consider different traffic classes. In the lab assignments you will implement some QoS tools and, when possible, validate them with the methods studied in the seminars.

1.3 Syllabus

- Lectures
    1. Introduction to QoS
    2. QoS Requirements
    3. QoS Service Level Agreements
    4. QoS Mechanics
    5. QoS Architectures
    6. Deploying Diffserv

7. Capacity Admission Control
8. SLA and Network Monitoring
9. Core Capacity and Traffic Engineering

- Seminars

1. Review of basic concepts. Exponential distribution. Poisson Traffic. Little's Theorem. PASTA theorem.
2. Delay in a network interface with Poisson arrivals, a single (finite) buffer and exponential transmission time.
3. Delay in a network interface with Poisson arrivals, two traffic classes and exponential transmission time. Preemptive priority and non-preemptive priority.
4. Delay in a network interface with a general transmission time. Priority queueing in a network with a general transmission time.

- Lab Assignments

1. Program a UDP Poisson traffic generator and a traffic sink capable of computing delay (min/avg/-max). Packet drop should also be measured.
2. Program a packet buffer. It should support both exponential and deterministic transmission time. The buffer size is taken as a parameter and it may be infinite.
3. Program a buffer that implements priority queueing. It should support both exponential and deterministic transmission time. The buffer size is taken as a parameter and it may be infinite.
4. Implement a QoS tool of your choice: policer, token bucket, leaky bucket.

5. Combine the different QoS elements that you and your classmates have programmed in a QoS enabled network. Invent an scenario, describe the requirements and explain how your solution addresses such requirements.

1.4 Bibliography

The lecture closely follow the book:

John Evans, Clarence Filsfils "Deploying IP and MPLS QoS for Multiservice Networks".

1.5 Evaluation Criteria

The grading is distributed as follows:

- Lectures continuous assessment, 10%
- Seminars continuous assessment, 10%
- Blackboard problem solving, 10%
- Lab assignments, 10%
- Individual continuous assessment quiz, 10%
- Final exam, 50% (Possibility of re-take exam)

It is necessary to obtain a decent mark (4 out of 10 or 20 out of 50) in all the different evaluation aspects. To pass the course, 50 out of the total 100 points need to be obtained.

1.6 Survival guide

1.6.1 How to pass the course

Statistically speaking, you will pass the course if you do all the following:

- Attend lectures, participate and ask questions.
- Attend seminars, try to solve the problems on your own and discuss them in small groups.
- Volunteer to solve problems on the blackboard.
- Attend labs and use lab time to solve the lab assignments.
- Participate in the planning and coding of the labs assignments. Read carefully the code of other team members and make sure you understand it and can explain it to others.
- Study for the continuous assessment quiz, as it is a warming up exercise to face the final exams with success guarantees.

1.6.2 Continuous Assessment

In this course we implement continuous assessment. This means that if you work hard from day zero, the course will be pain-free.

Continuous assessment includes multiple-choice quizzes in lectures and seminars. You also have to write reports in labs (one for each group). The source code and the report for the labs is submitted via moodle. Remember to write always your name and NIA in all the material you hand in or upload to

moodle. You also have to include you name and NIA in all the source code files you send me.

There is not a template for the report. I recommend describing key design and implementation aspects, clarifying possible deviations or improvements on the initial assignment, and including examples. The example should include the input commands as well as the output results. The report and the code will be submitted at the end of the class via moodle. At the beginning of the next class, you will have to demonstrate that your programs actually work.

1.6.3 Collaboration Policy

You are encouraged to collaborate with other students in the resolution of problems and assignments. However, you should firs try to solve it on your own. Then, you can discuss your solution with others and work together to find a better solution. Finally, you must ensure that you can solve the problem or assignment alone.

In the labs assignments you will work in teams of three people. Unless when explicit permission is given to re-use code, each team has to write their own code.

1.6.4 Formula Sheet

This course is not about memorizing equations. It is about understanding them. For this reason, you are allowed to use a *formula sheet* in individual tests as long as it fulfills the following requirements:

- It is a single page (one side).
- It is handwritten. Your own handwriting.

- It is delivered together with your answers when the test is finished.

1.6.5 Questions and doubts

I like to receive questions and comments. Normally, the best moment to express a doubt is during the class, as it is likely that many people in the class share the same doubt. If you feel that you have a question that needs to be discussed privately, we can discuss it right after the class.

1.6.6 Continuous feedback

At the end of the class, I will ask you to anonymously provide some feedback on the course. In particular, I always want to know:

- What is the most interesting thing we have seen in class.
- What is the most confusing thing in the class.
- Any other comment you may want to add.

  In my previous experience, this information has proven to be invaluable in improving the course, detecting problems at an early stage, and adapting the course to the expectations of the students.

  In labs, I will ask each group to hand in a short (few paragraphs) description of the work carried out in class, and the members of the group that have attended the class. Note that this is different from the lab report, which is the one that it is actually graded.

1.6.7 How to make you teacher happy

Avoid speaking while I am talking. It is not that you cannot talk in class. You can talk as much as you want when I am silent. I will make plenty of breaks in which I will ask you to discuss a question with your classmates. You can also take advantage of the moments in which I erase the blackboard or just scratch my head while staring. As long as I am not talking, you can talk with your classmates as much as you want. Obviously, questions are welcome at any time.

Chapter 2

QoS metrics

If you go to the postal office to submit a packet, you will be offered different options. In addition to the regular service, it is possible that an urgent service exists. Probably there is also the possibility sending the packet as certified mail, and some option for delivery notification. It is likely that there are also special services for packets that are voluminous or heavy.

QoS-enabled packet switched networks also offer different kinds of services for data packet delivery. In this chapter, we will review the different metrics that are relevant for data networks. This metrics can be used to establish *service level agreements* (SLAs) which are contracts specifying the QoS expected from a network. These contracts should also specify how the metrics are actually measured.

As an example, if a network guarantees a delay below 100 ms, it should be specified whether this makes reference to the maximum, the delay or the 95% percentile. The measures will also differ depending whether 5 minutes averages or 1 hour averages are considered. This makes the specification of SLAs tricky.

2.1 Delay

Delay is the time that it is required to traverse the network from the entry point to the exit point. Delay is normally considered for real-time services such as voice over IP (VoIP). The total end-to-end delay is simply the sum of the delay suffered in each of the hops in the data network. As an example, 2.1 shows a network with four hops.

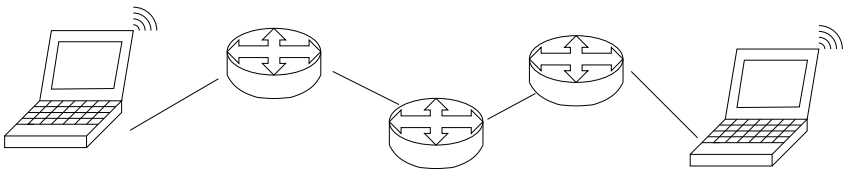

Figure 2.1: A network with two terminals, three routers and four hops.

In each hop, there are four different contributions to delay:

1. Processing: The time required for the router or switching device to put the packet on the outgoing interface queue. Very short.

2. Queueing: Waiting time on the outgoing interface queue. Very short if the queue is empty.

3. Transmission: The time required to put the packet on the transmission medium. It is a function of the packet length and transmission rate. Short in high-speed transmission media.

4. Propagation: The time that it takes for the packet to travel the distance from the hop source to the hop destination. Short time over short distances. And very long when it involves a trip to a geostationary satellite.

2.2 Jitter

Jitter is the variation of delay. This aspect is specially relevant for real-time and streaming applications. These applications expect the packets to arrive regularly in time. As an example, if the encoder application takes a voice stream and splits it into 20 ms chunks that are encoded and sent as packets, the receiver application will expect to receive one packet every 20 ms to reconstruct the voice stream.

If packets are sent every 20 ms but each of them requires a different time to traverse the network, the separation between packets will no longer be 20 ms at the receiving end. Applications sensitive to jitter use a de-jittering buffer that holds some packets and feeds the decoder at regular intervals. The packets that suffered a short delay in the network will wait for a longer time in the de-jitter buffer and the packets that suffered a longer delay in the network stay in the buffer for a shorter time. With this technique, jitter is effectively suppressed at the expense of increasing delay, as shown in figure 2.2.

Figure 2.2: The dejitter buffer removes jitter at the expense of adding delay.

The buffer size in terms of time and packets (or bytes) need to be carefully tuned, to prevent both packet overflow and underflow. Buffer overflow happens when the buffer is full and cannot accommodate an arriving packet. Buffer underflow occurs when the decoder asks for a packet and the buffer is empty.

2.3 Round Trip Delay

The round trip delay is the measure of delay used for elastic and interactive applications. It is the time required for a packet from source to destination and then back to the source again. You can easily find the round trip delay from your host using the ping command.

```
$ ping www.happyforecast.com
PING happyforecast.com (184.107.100.65)
    56(84) bytes of data.
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=1 ttl=48 time
    =122 ms
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=2 ttl=48 time
    =121 ms
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=3 ttl=48 time
    =122 ms
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=4 ttl=48 time
    =122 ms
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=5 ttl=48 time
    =122 ms
64 bytes from s106.panelboxmanager.com
    (184.107.100.65): icmp_req=6 ttl=48 time
    =121 ms
^C
--- happyforecast.com ping statistics ---
6 packets transmitted, 6 received, 0% packet
    loss, time 5004ms
rtt min/avg/max/mdev =
    121.864/122.066/122.181/0.230 ms
```

An interactive application (such as web browsing) requires a few round trip delays to complete the dialogue between the client and the server. Furthermore, the round trip delay also limits how fast the Transfer Control Protocol (TCP) congestion window can grow. The congestion window grows upon the reception of TCP acknowledgements. If it takes long for the TCP acks to arrive, the congestion window grows slowly.

In fact, it becomes difficult to take fully advantage of connections with a high bandwidth x delay product. These are known as "long fat pipes" or "long fat networks" (which is shortened as LFN and sometimes pronounced elephant), as explained in RFC 1072 [3].

2.4 Packet Loss

The networks may loose some of the packets being sent. Packet loss has different sources:

- Physical layer errors. The physical layer may flip some of the bits of the packet. As the packet typically includes some "cyclic redundancy check" (CRC), this errors are detected and the whole packet is dropped. In wireless communications, the occurrence of errors is likely and therefore some additional mechanisms are used. Forward error correction (FEC) codes can correct some of the errors. Furthermore, automatic repeat request (ARQ) are used to request retransmission of faulty packets.

- Queue packet dropping. In normal network conditions the queue should be almost empty. However, when the queues start to fill up, it is necessary to discard packets. Packets can also be discarded preventively, to regulate the pace of TCP flows.

- Network failure. Network failure can result in the loss of many packets, depending on the gravity of the failure and the restoration time.

From a QoS perspective, it is not only important how many packets are lost, but also the distribution of the loss in the data flow. As an example consider two networks that loose 10 packets out of 1000. The first network looses a packet out of every 100 packets, while the second one looses the 10 packets consecutively.

A VoIP application using packet loss concealment will be able to hide the packet loss of the first network, but not of the second one. If a packet voice is lost, the VoIP application can play the trick of playing the previous packet for two consecutive times, and the listener will hardly notice it. However, the loss of 10 consecutive packets may represent 2 seconds worth of audio that will be certainly noticed by the listener.

For a TCP application, the loss of a single packet will result in a quick retransmission. However, loosing 10 consecutive packets may move the TCP session back to the "slow start" mode and substantially reduce the throughput.

2.5 Bandwidth

Bandwidth or throughput is the amount of data transmitted per unit of time. Unfortunately, in practice it may have several different meanings.

It can be related to the line speed, which is the maximum data achieved by the physical medium (e.g, 100 Mbps or 1Gbps). Some technologies have a variable data rate, such as WiFi, that adapts the transmission speed to channel conditions. Furthermore, for a given technology, the bandwidth depends on which is the protocol layer under consideration.

As an example, a 54 Mbps WiFi device is capable of 54Mbps at the PHY layer, but roughly half of that is available at the network layer. The protocol overhead decrease the bandwidth as we move up the layer stack.

Quite often the term bandwidth refers to the amount of data per unit of time that it is actually sent, not at the capabilities of the network. Or it may refer to the amount of bandwidth that has been contracted to the Internet Service Provider (ISP). It was common to define this contracts in terms of "committed information rate" (CIR) that the ISP promised to carry, and the "peak information rate" (PIR) that the ISP would carry if it was possible. This bandwidth could be measured using "token buckets" that allow for some burstiness.

Another interesting concept is the 95th percentile billing, which is the kind of contract between ISPs and heavy traffic. In this contract there is a commit (or baseline speed) for a given price (say 20 Mbps for 180 Euro). The bandwidth consumption is measured in 5 minutes averages, and the 95th percentile value is considered. Then, if the 95th percentile is higher than the committed rate, the customer has to pay extra fee for each Mbps exceeding the committed rate (e.g., 10 Euro per additional Mbps).

2.6 Packet re-ordering

Many applications need that the packets are received in the same order than they are sent. To this end, TCP re-order packets and don't offer the data to the upper layers if there are gaps in the data flow. Similarly, in a VoIP application the voice packets should be handled to the decoder in order.

It is a desirable property of the networks that the maintain packet ordering. Some basic principles is assign packets

belonging to the same flow to the same queue. In case of a router that performs load balancing among different links, a hash of the source/destination ip address/port can be used to decide the path of the packet, to make sure that all packets of the same flow follow the same path.

2.7 Availability

High availability is a critical issue for data networks. It is often said that "five nines" or 99.999% is required from carrier grade networks. This means 5.26 minutes of (non-scheduled) downtime per year.

Some common metrics in terms of availability are the mean time between failures (MTBF) and the mean time to restore (MTTR). The availability can then be computed as $Availabilty = \frac{MTBF}{MTBF+MTTR}$.

Given a system which is composed of several subsystems either in "series" or in "parallel", it is possible to compute the syste overall availability. In "series" means that the system works only when all subsystems are working, and in "parallel" means that the system fails only when all subsystems fail.

2.8 Application requirements

Broadly speaking, we can classify the applications in two different kinds: inelastic (or real-time) and elastic (non-real time). An example of an inelastic application is VoIP and an example of an elastic one is remote backup.

Inelastic applications have a given bandwidth (and delay) requirements and are not flexible at all about this. As an example, if an streaming service requires 2Mbps, it will not work with 1.5 Mbits. The good part of the inelastic applications is

that they only consume what they require. Using the same example streaming application, it will use only 2 Mbps even if there is more available.

On the other side, elastic applications are more flexible about their requirements. If a backup service has only 1.5 Mbps available instead of 2 Mbps, the backup will take longer to complete that this is not typically a problem. Another characteristic of elastic applications is that they are often greedy, which means that they will take as much bandwidth as it is available. In this attempt to reach the limits of bandwidth availability, they create some temporary congestion or queue build up that is detrimental to real time services.

Somewhere in between real time and non real time applications we find interactive applications. They don't have the tight bandwidth and delay constraints of real time applications, but still they have to be responsive for the user to enjoy the experience. Examples of interactive applications are web browsing, Internet messaging.

2.8.1 VoIP

It has tight delay constraints (below 200ms) and does not consume too much bandwidth. As delay grows, conversation seems more artificial as there is the sensation that the other party is not responding. The human conversation protocols break down and it can be irritating. It is also unpleasant if there are noise and artifacts that make it difficult to understand the conversation, and for this reason packet loss has to be very low.

2.8.2 Videoconference

The requirements for voice are the same as for VoIP. Regarding the video, it requires more bandwidth but the quality is usually not that important. Users don't get too annoyed if the image freezes for half a second, as long as the voice quality is fine.

2.8.3 Streaming

It requires a lot of bandwidth and the display quality is important. The advantage is that it is possible for the receiver to have a buffer to partially alleviate jitter or even packet loss if retransmission is possible.

2.8.4 Video on demand

In this case the, the buffer can be larger as the content is pre-recorded and it is not critical to show it in real-time.

2.8.5 Web browsing

In principle, the load placed by web browsing is low and the delay requirements are mild (a few hundreds of milliseconds is not a problem). However, as the video on the web gains popularity, the bandwidth requirements are larger and get closer to those of Video on demand.

2.8.6 Peer to peer file exchange and remote backup

This may involve the transmission of massive amounts of data. The advantage is that they can often be delayed and accommodated in off-peak hours.

2.8.7 Gaming

Real time online gaming requires short delays. Normally the volume of data exchanged is low.

2.9 Service Level Agreements

A company owning different networks at different locations normally pays an Internet Service Provider (ISP) for connectivity among those networks. The ISP offers different services that are labelled with marketing names such as gold, silver and bronze. Each service has different characteristics in terms of QoS metrics.

The company has to take the decision to take one or more of those services according to the applications that are needed. Typically, for VoIP applications, the highest QoS (gold) is needed. Other applications such as nightly backup may use other services that allow to transfer high volumes of data at low rates (bronze).

Besides the QoS commitments, it is common to include in the contract a Committed Information Rate (CIR) and Peak Information Rate (PIR). The ISP provides QoS guarantees for the traffic which is below the CIR, and allows rates up to the PIR with no QoS guarantees. As an example, traffic over the CIR and below the PIR may be discarded in case of congestion.

To be more specific, the agreement includes not only a rate, but also a depth of a token bucket, which is a measure of the burstiness of the traffic. Token buckets will be covered in more detail in the next chapter.

2.9.1 95% percentile billing

An alternative of the CIR/PIR approach is the 95% percentile billing. The client commits to a given data rate. Then, the actual traffic is measured in averages of 5 minutes. The measure that is using for billing is the one that occupies the 95% percentile. That means that 95% of the measures are lower that this value and only the 5% are higher (peaks).

If the measured 95% percentile is higher than the committed rate, the client has to pay a penalty that is specified in the SLA. For example, 10 Euro for each Gbps in excess of the committed rate.

Chapter 3

QoS Tools

3.1 Why QoS tools?

In the last chapter we revised different applications with heterogeneous QoS requirements. If all these applications coexist in the same network, it is necessary to make sure that they all see their requirements fulfilled.

One possible alternative is bandwidth overprovisioning. If there is more than enough bandwidth, the queues are always empty and the packets suffer minimum delay and packet loss.

One of the problems of overprovisioning is that sometimes bandwidth can be very expensive. One of the challenges of the newtwork engineers is to satisfy the QoS requirements at a reasonable cost.

Another problem of overprovisioning is the elastic demand. Imagine that you overprovision a network to make sure that the queues are empty and the VoIP packets suffer no queueing delays. Some users will notice that the network is blazing fast. "Hey, I can download a HD movie in no time, I will download many of them so I can choose which one I want to watch." It is normal that users respond to bandwidth availability by

consuming extra bandwidth.

Finally, and very similar to the previous case, there are network security threats such as worms that will consume as much bandwidth as it is available. Sometimes a defective or misconfigured device will also consume as much bandwidth as it is available.

For all these reasons, overprovisioning cannot be the answer to everything. An alternative to overprovisioning is to use QoS tools to manage the available bandwidth in such a way the different QoS requirements can be met by prioritizing one traffic over the others and shaping the traffic flows.

Using the road analogy, if we have an ambulances with strict delay constraints, we can make the roads wider or use prioritization and other road traffic flow management (such as traffic lamps and reversible lanes).

A QoS solution classifies the traffic into different classes accordingly to SLA requirements and treats each of these classes differently across the network. Each of these classes is a "class of service" (CoS).

3.2 QoS at different layers

Our interest is on the provision of end-to-end QoS. The layer that provides end-to-end (internetworking) connectivity is the layer 3. In the TCP/IP stack, this is the IP layer.

Nevertheless, it is also worthy to know that Ethernet, WiFi (IEEE 802.11) and MPLS also include support for QoS. Ethernet packets include three bits in the header (Priority Code Point, PCP) to specify the class of service. MPLS tags also have three bits to indicate the priority. They are called EXP bits because they were in principle reserved for experimental use.

The case of WiFi is special because it is a shared medium, in which the different nodes contend to access the channel. There is also support for QoS in the sense that it is possible for stations with priority packets to contend more aggressively.

IP packets have a byte for QoS. Six of the bits are there to indicate the class of service, which is called "differentiated services code point". The other two bits are for "explicit congestion notification" (ECN).

3.3 QoS tools inside a router

To achieve QoS end-to-end, it is necessary to provide QoS in all of the intermediate hops. As an example, if we want to provide end-to-end bounded delay, it is required to provide bounded delay in every single hop.

To offer QoS for a traffic class, we need to define the behaviour of the routers the networks for that class of traffic. This is known as the per-hop-behaviour (PHB).

To implement this PHB, we will use a selection of QoS tools in each router. Normally we will need more than one tool to achieve the goal. Some of the tools that we will cover in the course are classifiers, meters, shapers, policers, queues and markers.

For each interface of the router, and for each direction, a chain of QoS tools is used. An example is given in Fig. 3.1 which shows a QoS tools chain for the inbound and outbound interface of a router.

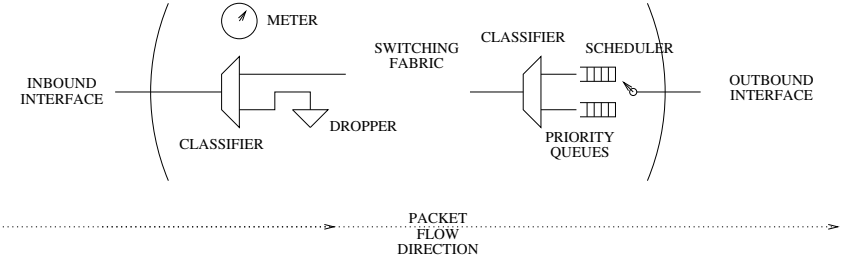

Figure 3.1: QoS tools chain of the inbound and outbound interface of a router.

Appendices

Appendix A

Lab Assignments

In these labs you can use any programming language that you want. In each assignment you must deliver the source code and a brief explanatory document explaining how you solved the assignment. It should also include some examples, including the commands that you used to test it and the results. Some assignments may ask for additional information, such as plots.

Pack all the files in a zip file (not rar) and submit it using moodle. Remember to include the names and NIA in all the source files and in the document.

Prepare the assignment in advance, so that you can complete it during the class. The submission deadline will be one week after the class.

A.1 Traffic Generator and Sink

In this lab assignment you will program a Poisson traffic generator and a traffic sink. The Poisson traffic generator takes the following parameters: destination host, destination port, packet rate and traffic class.

It generates UDP packets with a string that contains three integer values separated by a blank space. The integers represents a packet id (starting with 0), time stamp in milliseconds (local significance only) and traffic class.

The traffic sink takes a port number as a parameter and computes packet delay and packet loss for each packet class (computation of jitter is optional).

Test it with a traffic generation rate of 10 packets per second.

Note that since the generator and the sink are directly connected, the delay, the jitter and the packet loss will will be zero.

In the next assignment we will place a queue in between and these values will no longer be zero.

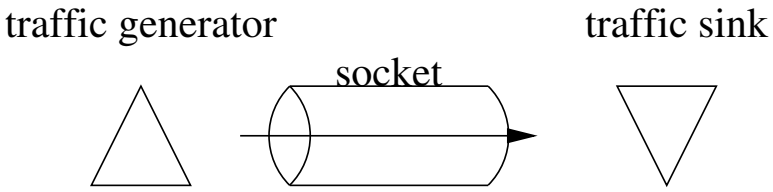

Figure A.1: Scenario to test in lab assignment.

A.2 A queue

The second module that you have to construct in this course contains a queue (that includes a dropper a buffer) and a scheduler.

Note that each module has to be a separate program. The different modules communicate (send packets to each other) using sockets.

This program listens at an udp port and transmits the

packets to a given udp port and address. Consequently, it has to be simultaneously an UDP server and an UDP client. You may consider the possibility of using different threads for the dropper and the scheduler.

All port numbers and the destination should be configurable as parameters. An additional parameter will configure the queue size (number of packets). If the queue size is set to zero, it means infinite queue length. If a finite queue is used, a taildrop policy will be applied.

The scheduler should be configurable to be able to choose an exponentially distributed service time or a deterministic service time. In either case, the service rate should be taken as an input parameter.

Combine the Buffer with the traffic generator and traffic sink modules to make measures of packet loss and delay.

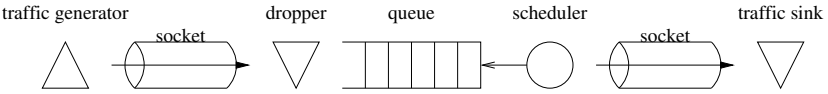

Figure A.2: Scenario to test in lab assignment 2.

A.3 Priority Queues

The third module that you have to construct in this course contains two priority queues (that include a dropper a buffer) and a scheduler. We will name H the high priority queue and L the low priority queue.

Each module has to be a separate program. The different modules communicate (send packets to each other) using sockets.

This program listens to two udp ports and transmits the packets to a given udp port and address. Consequently, it has

to be simultaneously an UDP server and an UDP client. You may consider the possibility of using different threads for the droppers and the scheduler.

The scheduler strictly prioritizes queue H. That is, it starts to serve queue L only if there are no packets at queue H. Nevertheless, this is non-preemptive priority, which means that the server will not interrupt a service to queue L when a packet arrives to queue H. The server will complete the service to the packet of queue L and only then it will serve the packet that has arrived to queue H.

All port numbers and the destination should be configurable as parameters. An additional parameter will configure the queue size (number of packets). If the queue size is set to zero, it means infinite queue length. If a finite queue is used, a taildrop policy will be applied.

The scheduler will draw service times from an exponential distribution.

Combine the priority queues with two traffic sources that generate different classes of traffic and obtain statistics of the delay for each of the traffic classes.

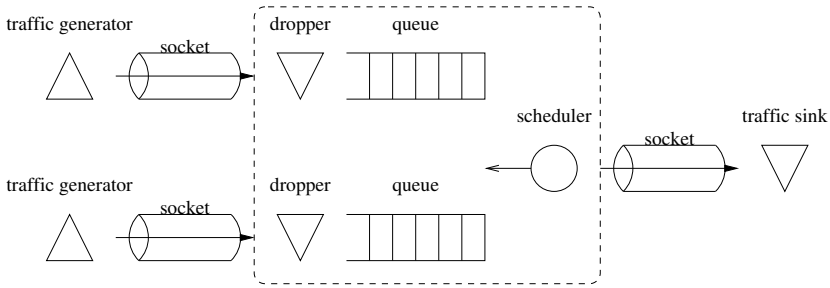

Figure A.3: Scenario to test in lab assignment 3.

A.4 Optional QoS Tool

In this lab assignment you will implement one ore more QoS tools of your choice. You can choose any of the following tools that we have seen in class: classifier, metering, token-bucket policer, leaky bucket shaper with RED, re-write.

A brief description of each of the tools follows:

- classifier: It has an input socket and several output sockets (one for each possible class of service). The classifier checks the class of server marking of the packet and redirects the packet accordingly.

- metering: It has an input socket and three output sockets. It uses two token buckets (as described in rfc2698) and it takes as an input four parameters: CIR (packets per second), PIR (packets per second), CIR burst (packets) and PIR burst (packets). The green, yellow and red packets are sent to each of the three different output sockets.

- token-bucket policer: It takes two parameters: rate (packets per second) and burst size (packets). Non-compliant packets are discarded.

- leaky bucket shaper with RED It takes a rate and a size as input parameters. There are two different approaches to implement RED: class-aware and class-blind. In the class-aware approach, only the high priority traffic is accepted when the occupancy of the queue exceeds 50%. In the class-blind approach, the probability of dropping an arriving packet is equal to the occupancy of queue at the moment of arrival.

- re-write It changes the class field of the packet.

Another option is to implement queueing disciplines from the ones that we have seen in class (e.g., weighted round robin, weighted fair queueing and deficit weighted round robin)

The idea is that different groups implement different tools. Keep in mind that in the next (last) lab assignment you should combine your tool and some of your classmate's tools to create a QoS scenario in which different service classes will receive a different treatment. You may first think about your scenario and then choose the tool you want to implement accordingly.

The complexity of metering, token-bucket policer and leaky bucket shaper with RED is considerably higher than the other options. It is highly recommended that each group implements at least one of these three tools.

A.5 Design and evaluate your own scenario

This is a free assignment. With all the knowledge gathered throughout the course and all the developed code, you have to invent an scenario and implement some sort of QoS.

You have to include some invented motivation. As an example, it may be your home network in which your flat mate is downloading a linux distribution while you are playing an interactive real-time online game. The download is generating one thousand packets per second and is filling the buffers while the game generates only 10 packets per second and is suffering excessive delays.

You can use different combinations of tools that we have seen and developed during the course. You can try different parameter configurations (queue length, burst size, rates, different kinds of RED, etc.) and report the results obtained with each of them. Offer an expert recommendation about which

is the best solution and why.

You can also combine and use the queueing theory techniques that we have seen in class, if you wish.

You will have to prepare a presentation (5-10 minutes) to explain your classmates about your project (motivation, possible solutions, test plan, etc.). Additionally, when you have performed all the tests and computations, you will have to deliver a short report (max. 5 pages) detailing your work.

You can also use code generated by other groups. Include the slides, the report and all the used code in one zip file that will be submitted via moodle. Make sure to clarify which code has been generated by you and which code has been developed by other groups and re-used in your project.

Bibliography

[1] Gari R. Bachula. Testimony of Gary R. Bachula, Vice President, Internet2.

[2] A. Cooper, A. Jacquet, and A. Soppera. Bandwidth Usage and Management: A UK Case Study. In *Research Conference on Communications, Information and Internet Policy (TPRC)*, 2011.

[3] V. Jacobson and R.T. Braden. TCP extensions for long-delay paths. RFC 1072 (Historic), October 1988. Obsoleted by RFCs 1323, 2018, 6247.