A course on Quality of Service (QoS)

Jaume Barcelo

May 14, 2014

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 these 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 to 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.

An straightforward solution to prevent that peer-to-peer file sharing interferes with voice is keeping them in separate networks. Why not? Well, at some point network engineers realized 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 differ-

ent services have totally different requirements with regards to required bandwidth and delay.

The services 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 [2]. Nevertheless, QoS (or "bandwidth management") has been used by ISPs in practice [5]. 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. About the course
    2. QoS metrics
    3. QoS tools
    4. Scheduling

5. Active queue management
6. Differentiated services
7. RSVP and MPLS-DiffServ-TE
8. Individual continuous assessment quiz
9. Presentations of the projects

- 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. Theoretical analysis of a priority-aware traffic shaper.
5. (Optional) A seminar on a current QoS topic (e.g., LEDBAT, IEEE 802.11e, Bufferbloat)

- 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 Background

The course assumes the familiarity of the student with

- data networks, including IP networks, Ethernet networks, and MPLS networks,
- router and switching devices, interfaces and the existence of a control plane and forwarding plane,
- routing protocols such as OSPF,
- elementary algebra,
- differentiation,
- probabilities and random processes,
- basics of queueing theory,
- socket programming,
- multi-thread programming.

1.5 Bibliography

The first lectures follow the book:

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

The last lecture is based on

Ina Minei, Julian Lucek "MPLS-enabled applications"

Some pointers to queueing theory material:

Philippe Nain "Basic Elements of Queueing Theory (Application to the modelling of Computer Systems)" (Lecture notes)

Ivo Adan, Jacques Resing "Lecture notes on queueing theory"

1.6 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 (5 out of 10 or 25 out of 50) in all the different evaluation aspects.

1.7 Survival guide

1.7.1 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.

1.7.2 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.

1.7.3 Self-evaluation tests

This course includes self-evaluation tests in the form of multiple choice questions. Print the tests in advance and answer them right after the lecture. I will collect the tests at the end of the class and give them back to you the following class, so that we can correct them in group.

1.7.4 Collaboration Policy

You are encouraged to collaborate with other students in the resolution of problems and assignments. However, you should first 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.

1.7.5 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.

1.7.6 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.

1.7.7 Continuous feedback

At the end of the class, I will ask you to 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.

1.7.8 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 at my notes. 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.

1.8 About this document

This document lives in github:

https://github.com/jbarcelo/QOS-lecture-notes

You can fork it and improve it and send me a pull request. In the git repository you will also find the self-evaluation tests.

I generated this document prompted by the feedback I received in the first edition of this course, in which the students asked for some supporting material besides the blackboard lectures.

1.9 A bunch of questions

1. Is it better to have a network for every service or a single network for all the services?
2. What are triple play services?
3. What is the problem of having multiple services in the same network?
4. Have you ever tried VoIP with P2P traffic or other heavy downloading? What happens?
5. What are two possible approaches to solve this problem?
6. Advantages and disadvantages of both?
7. In which point in the path should we apply QoS?
8. When does QoS kick in? In which traffic conditions?
9. Do you think that QoS has been used to control P2P traffic? In which way?
10. Which of the current Internet applications consume the largest volume of traffic? Can this application be managed with QoS techniques as easily as P2P file sharing?
11. What is an RFC?
12. Which organization produces the RFCs?
13. There are two main approaches to QoS: Intserv (RFC1633) and Diffserv (RFC2475). What is the approach of each of them? What are the differences? What are the advantages and disadvantages?
14. What is MPLS?

15. What are the advantages and disadvantages of MPLS compared to IP routing?

16. What is a VPN?

17. What happens if some of the routers in the path support QoS and some don't?

18. Can you provide examples of applications that require QoS guarantees?

19. Which are possible classifications of traffic regarding QoS requirements and resource consumption? Can you provide examples by classifying specific applications?

Chapter 2

Advertisement

In this chapter we reproduce a *technical specification* from *BT*. It illustrates the kind of offers and jargon used by *ISP*s. The goal is that, by the end of the course, you have a good understanding of what this offer means and how the *ISP*s internally engineer their networks to be able to provide the advertised services.

2.1 BT MPLS

2.1.1 BT MPLS technical specification

We're striving for our high-speed, fully secure IP VPN platform to set the standard for others to follow. Our BT MPLS (Multi-Protocol Label Switching) service over a Cisco powered network offers you the following main advantages:

- High-performance packet switching along pre-determined paths.
- Independence of the underlying open systems interconnection (OSI) layer 2 connectivity.

- High scalability enabling rapid network expansion.
- Segregation of client VPN traffic from others by the use of unique route distinguishers.
- Retention of your existing IP address scheme.

BT MPLS technology combines with Differential Services (DiffServ) and traffic engineering techniques to provide Quality of Service. It does so through the use of Priority Queuing Class Base Weighted Fair Queuing, Committed Access Rate, Weighted Random Early Detection (WRED), Frame Relay and ATM traffic shaping. This combination of MPLS and traffic engineering enables BT MPLS to differentiate between time critical, high priority traffic and delay tolerant, low priority traffic. You can classify and prioritise your applications into different Classes of Service, matching network performance with business need. MPLS offers a similar level of in-built security to existing Frame Relay and ATM networks.

More mechanisms to manage the performance of all applications on your network are available when you combine BT MPLS with the Application Optimisation Service (AOS) of Application Assured Infrastructure (AAI). With AOS the performance objectives of each individual application can be set. The service dynamically assigns network resources to each application to ensure that your performance objectives are met.

2.1.2 6 Class of Service (CoS) model

BT MPLS' 6 Class of Service (CoS) model sets us apart in the global market. It is a world leading Class of Service proposition and addresses the growing trend of multiple application use by clients. This trend includes:

- Increasing use of multiple enterprise resource planning (ERP) applications to support critical business processes.

- Greater use of, and interest in, IP telephony and video streaming applications, with LANs becoming more sophisticated.

- Quality of Service (QoS) mechanisms being introduced in client networks with DiffServ becoming the de facto standard for CoS implementations.

The 6 Class of Service Model gives you the ability to enjoy a greater granularity in bandwidth prioritisation and partitioning. You can choose to prioritise mission critical applications such as Siebel, SAP, Oracle and Lotus Notes into distinct prioritisation data classes. This enables you to run multiple applications simultaneously. In addition, you are able to run multimedia applications as well as IP Voice, enabling you to gain a high level of convergence on a global platform.

The 6 CoS model provides high flexibility and scalability, enabling you to easily burst between classes without the need for complicated configurations. Changes to your network can be kept to a minimum to support the new CoS model with end to end transparency ensuring configurations occur easily and swiftly across multiple sites. With the use of the additional AOS of AAI you can maximise the performance of your network in accordance with your changing business priorities.

2.1.3 The Offer

The CoS model uses DiffServ Code Point. We have developed the BT MPLS CoS model with clients and their application providers.

The 6 CoS model contains:

1. Voice - such as VoIP and PSTN breakout.

2. Assured Data - ERP applications or Real-time Multimedia - such as video conferences and other interactive services.

3. Assured Data - ERP applications.

4. Assured Data - ERP applications.

5. Assured Data - ERP applications.

6. Standard - email, intranet, file transfer protocol (FTP), internet breakout telnet and other network management.

2.1.4 A communications solution for all

BT MPLS is proving extremely popular with small and medium enterprises as well as multi-national organisations. We have more than 2,000 clients and 43,000 ports connected to our platform globally. The service is for you if you require:

- Performance optimisation of your network applications to improve processes.

- Any-to-any connectivity - either through a meshed or partially-meshed network.

- The ability to prioritise the transmission of different types of data applications.

- Different performance and prioritisation levels for various application types.

- A network that can expand and contract easily in line with your own expansion strategy.

- LAN-to-LAN or WAN-to-WAN connectivity over private networks.

If you want to include dynamic optimisation for your network check the capabilities of our AOS from our AAI product.

BT MPLS is available in over 115 countries with full 6 Class of Service (Cos) capabilities, known as Native IP, and from more than 1200 Points of Presence (PoPs) in Europe, the Americas and the Asia Pacific Region.

2.1.5 Class of Service network performance guarantees

The BT MPLS 6 Class of Service (CoS) model addresses the growing need for multiple applications with different performance levels. In combination with AOS of AAI it provides you with increased visibility and control over your application performance. Current market trends include:

- Increased use of multiple ERP applications to support critical business processes.
- Greater use of and interest in IP Telephony and video streaming applications.
- Increasingly sophisticated LANs.
- Quality of Service (QoS) mechanisms being introduced into client networks.
- DiffServ becoming the de facto standard for CoS implementations.

2.1.6 Application Class of Service

Voice

An any-to-any IP service capable of delivering Voice over IP (VoIP)applications for those clients wishing to construct “do it

yourself" voice solutions. Underwritten by installation, availability, round trip delay, packet delivery and jitter service level agreements. Application example : IP Voice, Unified Communications

Multimedia

Optimised to support realtime video applications, the class is underwritten by service level agreements on installation, availability, round trip delay and packet delivery. A maximum of four Assured Data classes are available with the Multimedia class. Application example: Video conferencing, video streaming

Assured Data (up to four applications or mix of multimedia applications)

Up to four Assured Data classes are available for missioncritical or delay-sensitive data applications. Each critical data application can be assigned to a dedicated Assured Data class, enabling bandwidth to be dedicated to individual critical data applications. This prevents critical data applications competing for the same bandwidth and ensures the performance of each application. Assured Data classes are an any-to-any IP service underwritten by installation, availability, round trip delay and packet delivery service level agreements. Application example:

Missioncritical, delay sensitive applications such as:

- Oracle
- SAP
- ERP
- Data Warehousing

- Citrix
- Client Server
- Multimedia applications

Standard Data

A basic any-to-any IP transport service for delay-tolerant data applications such as email or Intranet access. The Standard Data class is underwritten by installation, availability and round trip delay service level agreements. Application example:

Delay tolerant applications such as:

- Internet/intranet browsing
- File transfers
- Other applications that can accommodate variable throughput delay

BT has an outstanding track record in implementing MPLS and provides solutions uniquely tailored to the requirements of your organisation. Clients as diverse as car hire companies and oil and gas service providers have benefitedfrom the successful introduction of BT MPLS.

With more than 20 years of providing VPN services to global businesses, and network coverage across five continents, BT is at the forefront of designing, implementing and managing global IP VPNs. As a key partner with Cisco, we were one of the pioneers of commercial MPLS services.

BT offers you comprehensive service level agreements and a single point of contact, in addition to our network design and service optimisation consultancy.

With BT MPLS, we can help you match your applications to Class of Service requirements, defining and optimising the routing protocol architecture, planning migration and troubleshooting inter-working problems.

The result is the kind of flexible and powerful network service that can help you enjoy business success and growth.

Chapter 3

An Introduction to BGP

As the Internet grew it was no longer possible to keep all routers in a single routing domain. It was necessary to add another layer of organization. All the routers that belonged to the same organization and were controlled by the same network administration team were grouped in an Autonomous System. For example, Universitat Pompeu Fabra belongs to autonomous system Red Iris with Autonomous System Number AS766. Guifi.net is a community network that has AS49835.

AS numbers (ASN) are assigned by the Internet Assigned Number Authority (IANA) to Regional Internet Registries (RIR). RIPE-NCC is the European registry. Autonomous Systems have to apply for their ASN to the RIR. These steps are also followed for IP addresses assignments.

ASs can be classified in three different types:

- Multi-homed: An AS that is connected to more than one neighbouring AS. Even if one of the neighbours fails, a multi-homed AS can retain Internet connectivity via another neighbour.

- Stub AS: In this case, the AS appears to be connected

to a single neighbouring AS. It can be that an Stub AS has private peering agreements with other ASs.

- Transit AS: An AS that uses its networks to carry traffic of other ASs.

Within an autonomous system, the system administrators configure an Interior Gateway Protocol IGP. Examples of IGP are RIP, OSPF and IS-IS. The network administration team agrees in the protocol that is going to be used and the per-hop metrics. Then the protocol computes the routes that minimize those metrics. IGP decide which is the next router in the path towards the destination.

There is another layer of routing between Autonomous Systems. When a network has a packet destined to another autonomous system, it needs to decide to which neighbouring autonomous system that packet will be forwarded.

The protocol that is used in the Internet to route between Autonomous Systems is the Border Gateway Protocol (BGP). A key difference between IGPs and BGPs is that the latter routes packets between different organizations that not necessarily trust each other nor share the same goals. Choosing the shorter path is not always the best option, and it is not possible in agreeing in common metric for each connection between two ASs. If two alternative paths exist, one AS may prefer one path and the other AS the other path.

There are ASs of many different sizes and scopes. Some ASs are local and provide access to end users, while others are international and only provide services to other ISPs. An example of the former is Jazztel and an example of the latter is Cogent.

The connections between ISPs can be of two different types: either transit or peering. In the peering relation, two ISPs exchange traffic without charging each other. In the case of

transit, there is one ISP that is a customer of the other and pays for the carried traffic.

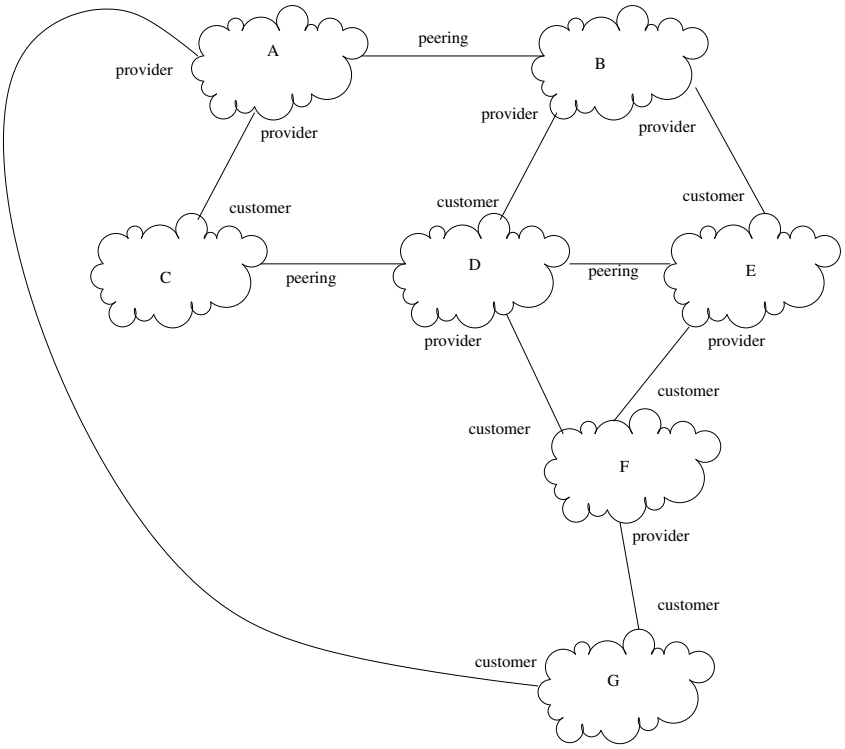

Figure 3.1: Example network topology with Autonomous Systems and peering and transit connections

Fig. 3.1 shows an example topology in which clouds represent autonomous systems that are connected with transit and peering links. Different autonomous systems normally connect at Internet exchange points. Catnix is one example of an Internet exchange point.

In routing between different ASs, it is necessary to account for policy. Each ASs has a different policy that is translated to routing announcements and routing decisions.

For example, if C has a packet for the AS G, the shortest

path is via AS A. However, AS C has to pay for the transit that is sent to A, and therefore C prefers to send the packet to AS D with which it has a peering agreement. In this case, policy is taken into account to make a routing decision.

An example of route announcement conditioned by policy is that AS D does not announce a route to AS E to AS C even though this route exists. The reason is that AS D is not being payed for carrying transit from AS C to AS E and in principle has no incentive to carry that traffic.

With BGP both route announcements and routing decisions can be conditioned by policy. In principle, the order of preference for routing a packet is

- to a client (the AS makes money)
- to a peer
- to a provider (the AS pays money)

Regarding the announcements, the normal policy is

- announce the routes of clients to everyone.
- announce the routes of peers to clients.
- announce the routes of providers to clients.

It is possible to also set policy in the form of BGP filters. Back in 2008, Pakistan telecom hijacked Youtube traffic by announcing a more specific prefix. This announcement was not filtered. Instead it was filtered and propagated by other AS effectively redirecting Youtube traffic to Pakistan Telecom.

The following is a list of events[1]

[1] quoted from http://www.circleid.com/posts/82258_pakistan_hijacks_youtube_closer_look

- 18:47:45 First evidence of hijacked route propagating in Asia, AS path 3491 17557
- 18:48:00 Several big trans-Pacific providers carrying hijacked route (9 ASNs)
- 18:48:30 Several DFZ providers now carrying the bad route (and 47 ASNs)
- 18:49:00 Most of the DFZ now carrying the bad route (and 93 ASNs)
- 18:49:30 All providers who will carry the hijacked route have it (total 97 ASNs)
- 20:07:25 YouTube, AS 36561 advertises the /24 that has been hijacked to its providers
- 20:07:30 Several DFZ providers stop carrying the erroneous route
- 20:08:00 Many downstream providers also drop the bad route
- 20:08:30 And a total of 40 some-odd providers have stopped using the hijacked route
- 20:18:43 And now, two more specific /25 routes are first seen from 36561
- 20:19:37 25 more providers prefer the /25 routes from 36561
- 20:28:12 Peers of 36561 start seeing the routes that were advertised to transit at 20:07
- 20:50:59 Evidence of attempted prepending, AS path was 3491 17557 17557
- 20:59:39 Hijacked prefix is withdrawn by 3491, who disconnect 17557

Pakistan Telecom advertised a prefix (208.65.153.0/24) that was more specific that the one advertised by Youtube (208.65.152.0/22)

and therefore it was used by BGP routers. The response of Youtube was to announce two even more specific routes /25.

3.1 Path Routing

To make it possible to implement policy routing, BGP uses path announcement. For example, if G contains the 12.34.56.00/24 network, it will announce it as:

`through ‘‘G, ’’ reaches 12.34.56.00/24.`

AS F will re-announce the route as

`through ‘‘F, G, ’’ reaches 12.34.56.00/24.`

And AS D will re-announce the route as

`through ‘‘D, F, G, ’’ reaches 12.34.56.00/24.`

Then AS C compares this announcement with other announcements such as

`through ‘‘A, G, ’’ reaches 12.34.56.00/24.`

And makes a decision according to its policy.

To prevent loops, an AS never accepts a route announcement if it finds itself in the announced path.

3.2 Route aggregation

On of the challenges of the current internet is the growth of the global routing table. At the time of writing, a DFZ router can see more than half a million routes. A partial solution to the growth of the number of routes is route aggregation.

A BGP router can combine different announcement of consecutive networks in a single one. For example, if AS D in Fig. 3.1 announces 12.34.58.0/24 and AS E announces 12.34.59.0/24, then AS B can announce 12.34.58.0/23.

```
through ‘‘Sequence(B), Set(D, E) ’’ reaches 12.34.58.00/
```

Sequence means that the whole network is announced through B. Set means that a fraction of the network is announced through D and E.

3.3 BGP connections and messages

IGP protocols such as OSPF and RIP try to discover neighbouring routers to exchange routes. This is not the case with BGP routers. In BGP, the administrator configures the BGP neighbours with which the BGP router will exchange routes. The two neighbours (or peers) then establish a TCP connection and exchange messages.

The BGP messages are the following:

- OPEN to start the connection.
- UPDATE to send routing information. When the routing information changes, the BGP routers will send additional UPDATE messages.
- KEEP-ALIVE to maintain the connection open when there is no other data to send.
- ROUTE-REFRESH to ask the other party to re-send all routing information.
- NOTIFICATION to inform of errors.

The BGP protocol includes a security feature that requires both parties to agree in common secret and use it to derive an MD5 hash of the exchanged messages that is included in the messages. The receiving end can re-compute the MD5 value and compare with the received one. Using the shared secret, BGP can prevent that a malicious attacker forges BGP messages.

3.4 Interior BGP

It is normal that an autonomous system has multiple connections with other AS. It may have different BGP routers at different locations of the network. It is necessary that those routers coordinate and for this purpose they can use Interior BGP (or IBGP).

IBGP routers are normally at different borders of the network. They are not directly attached to each other. They talk using TCP sessions that cross the AS traversing different routers.

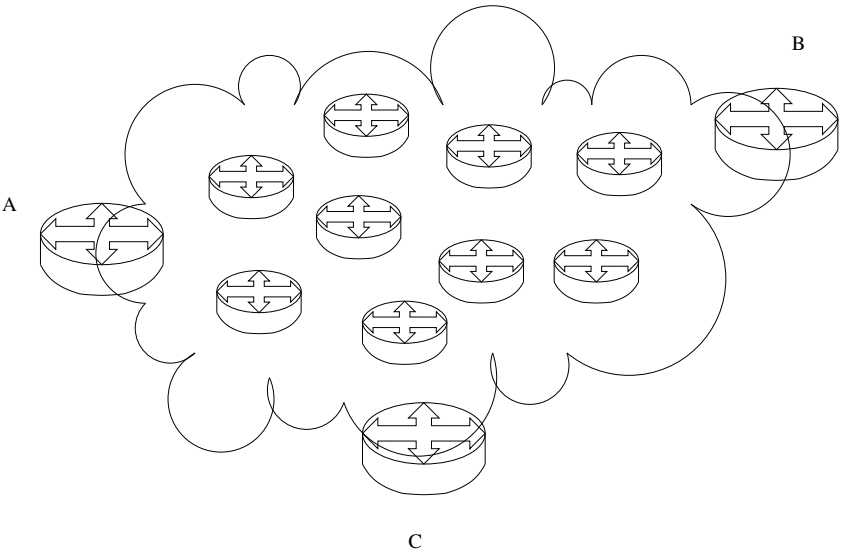

Figure 3.2: An autonomous system with three BGP routers.

Fig 3.2 shows an autonomous system with many routers. Three of them are border routers that exchange information between them using IBGP. Let's consider that this is AS 33.

In IBGP the routers must accept routing advertisements even if there is their own ASN in the path. The announcement received via IBGP are not propagated to other IBGP peers.

Therefore, routers B and C in the example will not relay the announcement to each other. This prevents the creation of roting loops.

As a consequence of the fact that IBGP announcements are not relayed to IBGP neighbours, it is necessary that all IBGP routers of the AS are connected to each other. If the AS is large and has several IBGP routers, it is not scalable to have all IBGP routers connected as a full mesh. In this case, there are two possible alternatives: *route reflectors* and *confederations*.

Chapter 4

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 average 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.

4.1 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, 4.1 shows a network with four hops.

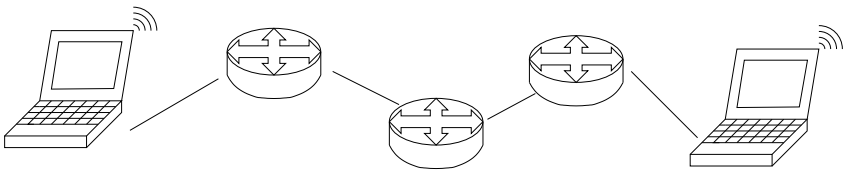

Figure 4.1: A network with two terminals, three routers and four hops.

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.

4.2 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 4.2.

Figure 4.2: The dejitter buffer removes jitter at the expense of adding delay.

The de-jitter 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.

4.3 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 throughput 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 [9].

4.4 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 0.2 seconds worth of audio that are likely to be 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.

4.5 Throughput

Throughput (sometimes also referred to as bandwidth) 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 throughput 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 throughput as we move up the layer stack.

Quite often the term throughput 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 throughput 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 throughput 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 consumers/producers. In this contract there is a commit (or baseline speed) for a given price (say 20 Mbps for 180 Euro). The throughput 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).

The ISP normally have different kind of offers the user can choose from, depending on the predicted user consumption. For example, the same ISP considered in the previous paragraph might have an offer of 40 Mbps for 300 Eur plus 5 Eur for additional Mbps.

4.6 Packet re-ordering

Many applications need that the packets are received in the same order than they are sent. To this end, TCP re-orders packets and doesn'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 they maintain packet ordering. A basic principle is to 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.

4.7 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 system 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.

4.8 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 throughput (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 throughput as it is available. In this attempt to reach the limits of throughput 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 throughput 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.

4.8.1 VoIP

It has tight delay constraints (below 200ms) and does not consume too much throughput (around 100 Kbps per flow). 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.

A as a target reference, packet loss should be kept below 1%.

4.8.2 Videoconference

The requirements for voice are the same as for VoIP. Regarding the video, it requires more throughput 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.

4.8.3 Live Streaming

It requires a lot of throughput 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.

4.8.4 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.

4.8.5 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 throughput requirements are larger and get closer to those of Video on demand.

4.8.6 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.

4.8.7 Gaming

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

4.9 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.

4.9.1 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 used 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 5

QoS Tools

5.1 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. In case of wireless communication, bandwidth is a limited natural resource. 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 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).

5.1.1 Backup and maintenance

Private users normally have Internet access with no bandwidth guarantees. They pay for a 20 Mbps ADSL line, but how much they can get out of it is unknown.

Business users often choose a service with bandwidth guarantees, which is much more expensive than the one that has no guarantees. Furthermore, for increased reliability, it is normal that business pay for a second line for backup purposes. As paying for two high-speed lines would be too expensive, a more economic option is to choose a slow connection for backup.

If the main (expensive) line is out of service (either due to

failure or programmed maintenance), the backup line has to support all the traffic. At this point it is important to prioritize critical business applications at the expense of others that can be delayed to another time or another day.

As an example, if web browsing is not a critical business application but access to a remote database is, the users might not be able to browse the web until the main line has been restored. However, this fact will not seriously impact the ability of doing business as usual.

5.2 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).

5.3 IP flow, IP class

IP packets can be grouped in flows. As an example when you are browsing a webpage and send multiple HTTP request packets, all those packets belong to the same flow. The headers of the packets of the same flow have three fields in common:

- The layer four protocol (either TCP or UDP)
- The source and destination IP address
- The source and destination port.

It is important that all the packets of a flow follow the same path and are stored in the same queues, to prevent packet reordering.

The number of IP flows traversing a network is humongous and therefore it is not possible to apply QoS on a per-flow basis. The alternative is to group several flows with similar QoS requirements on the same "Class of Service".

As an example, all web traffic flows could be mapped to the same CoS. Also flows from other application with similar requirements (IRC, email) could be included in the same CoS.

The millions of flows traversing a router are assigned to reasonable number of traffic classes, such as four or eight. Then, the router treats each of these classes differently in what is known as the per-hop-behaviour PHB.

Having a reduced number of classes makes the implementation and configuration of router much more easier. It makes it even possible to use automatic tools to configure and reconfigure the routers of a network.

5.4 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. 5.1 which shows a QoS tools chain for the inbound and outbound interface of a router.

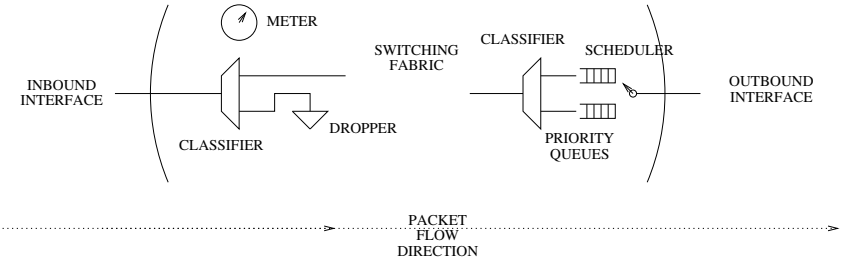

Figure 5.1: QoS tools chain of the inbound and outbound interface of a router.

5.4.1 Classifiers

To apply QoS, it is required to discriminate among different classes of traffic. Therefore, the first step in any QoS tool

chain is always classification. This classification is performed by a tool that we call classifier.

These are some of the classification criteria that can be applied by a classifier:

- Incoming interface: Packets can be classified using the interface (either physical or virtual) that they are coming from. As an example, we can have all the IP phones in our network in the same VLAN, and then classify the packets of this VLAN as expedited forwarding (EF).

- Metering: We can take a classifying decision based on traffic volume. This is typically done with token buckets (e.g. RFC 2698 [8]). Traffic conforming to a token bucket of a given rate and burst size receives a different classification than non-conforming traffic. It is also possible to classify the traffic in three different colors (green/yellow/red) according to the CIR/PIR criteria.

- QoS fields: We have mentioned that IP, Ethernet and MPLS packets all have a field for QoS. The bits (the marks) in these fields can be used for classification purposes.

- Other headers: The other headers of the packets can also be used for classification purposes. For example, if we have a server devoted to VoIP services, we can prioritize all the packets destined to (or originated from) that server.

- Deep packet inspection: This involves opening the packet and looking into its contents to take a decision. It can be useful to detect signatures of virus, and identify higher layers protocols. In the field of QoS, it is used for example to classify P2P traffic (bittorrent, kazaa).

DPI is also used for security, surveillance, espionage and censorship purposes.

- Stateful inspection (SI): In stateful inspection we use not only information in the packet, but also information of previous packets. As an example, it can be used to identify encrypted P2P traffic by identifying patterns such as communications with many other devices taking both the role of client and server.

DPI and SI are computationally expensive. It would be desirable to perform them only once, rather than repeating the effort in every router. The problem is that all the classification effort is lost after the packet leaves the router, unless the QoS headers are explicitly changed. In the next subsection we introduce the marker, which is the tool that changes QoS markings.

5.4.2 Markers

Classifying can be computationally expensive. Additionally, some classifications can be done only at the entry point of the network. As an example, imagine a classification based on a metering tool for the traffic of a given client. It is feasible to use the metering tool and the classifier in the interface that the client uses to connect to the network. However, after that traffic has been aggregated with traffic of other customers, it might not be possible isolate it with the purpose of metering. In core routers with the larger line-speeds, only very simple (hardware based) classifications are possible.

For all these reasons, it is necessary that the routers can pass some information to each other regarding classification. The tool for this is the marker, that changes the bits of the QoS headers of the packets.

Where to classify and mark?

Classification is done in the entry point of the network, which is also called the trust boundary. If a network wants to identify P2P traffic to throttle it down, it will do it at the edge routers, which are the first routers that have the opportunity to inspect the packets. Then, the QoS field of the packet headers will be marked so that other routers of the network don't have to repeat the expensive classification process. The classification done by the other routers will be straightforward as all they have to do is to check the QoS field.

The volumes of traffic in edge routers are smaller than in core routers. Edge routers have the necessary time to classify and mark the packets. Edge routers also can easily differentiate traffic of different customers for metering purposes.

This initial marking may also be accompanied by policing. A common example is to classify the traffic coming from a customer into three colors in the edge router. Green packets are the ones conforming the CIR. Yellow packets are the ones conforming the PIR. And Red packets are the ones exceeding the PIR.

Red packets will be dropped (policed). Yellow packets will be marked for drop precedence. And green packets will be forwarded unmarked. In case of congestion, yellow packet will also be dropped.

If the yellow packet that has been marked for drop precedence continues its trip towards its destination, it may be the victim of active queue management (AQM) tools at some later point.

5.4.3 Policers

Policers combine a meter and a dropper. The meter is a classifier that has been explained before, and the dropper simply drops packets. Policers play a role in enforcing CIR/PIR agreements and also in guaranteeing bounded delay for packets traversing the network, and a fair bandwidth distribution.

If an SLA defines a PIR and the customer sends traffic exceeding the PIR, the ISP will police the exceeding traffic. A reasonable customer might police the traffic himself, to select which are the less important packets and avoid that the ISP drops packets. Another alternative is that the customers shapes its traffic. We will cover shapers in a later subsection.

Policers can also be used before queues. By controlling the input rate and the size of a queue (or a set of queues) it is possible to provide delay and bandwidth guarantees to each queue.

As an example, if a queue for VoIP packets is serviced at a rate of 100 Mbps and the input is policed at a rate or 1 Mbps, with a bucket size of 100 Kbits, the delay is bounded to $\frac{10^5 bits}{10^8 bps} = 10^{-3} s$. Note that the delay guarantee is derived from the bucket size (or somewhat equivalently, queue size) and not from the rate.

The rate limit is important when the voice queue is prioritized over other queues. If the priority queue is not policed, it can starve other queues in case of misbehaviour.

Note that policers introduce packet drop but don't introduce delay. There is a token bucket that stores tokens, but not packets. The next subsection introduces traffic shapers that can prevent packet loss at the expense of delay.

5.4.4 Shapers

We have seen that policers drop traffic that does not conform to a given traffic profile. An alternative of dropping is buffering. When a shaper receives a packet and there is not enough tokens in the bucket to serve it, it will store it in a queue. As soon as there are enough token in the bucket to serve the first packet of the queue, the shaper will allow the first packet of the queue and decrement the number of tokens in the bucket accordingly.

As the buffer size is finite, it is possible that the buffer fills up and the shaper loses packets.

Shapers are normally not appropriate for VoIP, as they introduce delay. However they can be used to set the pace of TCP flows effectively and for this reason it is said that they are TCP friendly.

By holding packets in the buffer, the shaper increases the RTT delay. The result is that the TCP rate is slowed down. In the event that the buffer fills and a packet is lost, TCP halves its transmitting rate. However, as the buffer is full, the shaper keeps transmitting at a full rate for some time and gives time to TCP to build up the congestion window and increase the transmitting rate again.

An alternative implementation of the shaper is to use a leaky bucket instead of a token bucket. The leaky bucket simply stores packets and "leaks" them at a given rate. Therefore, the leaky bucket does not allow any kind of traffic burstiness.

A shaper is used when the effective use of available bandwidth is more important that the delay. Excessive buffer can be detrimental as it increases the delay and interferes with normal operation of the TCP protocol.

5.4.5 Queues, droppers and schedulers

This is a critical component of QoS. Packets are stored in one or more FIFO queues and there is a scheduler that "serves" the queues by taking the first packet of a queue. Another element of the system is the dropper, that drops packets.

There are several alternatives for the implementation of the scheduler and the dropper and we will discuss them in the next chapter.

We want to mention here that the queues do not usually contain the packets as it is much more effective to store pointers to the packets.

5.4.6 The TX-ring and the interleaver

The last element of the tools in router, just before the transmission line, is the transmission ring (TX-ring). The normal configuration is having a queue system and the scheduler taking packets from the queues and placing them in the TX-Ring. Differently from the queues, the TX-ring does not contain a pointer to the packet. It contains the actual packets, ready to be transmitted. The goal of the Tx-ring is to feed the transmission line, to make sure that it is continuously transmitting if there are packets to be serviced. The Tx-ring is normally dimensioned in such a way that there is at least the packet that is currently under transmission and one additional packet that will follow.

The TX-ring is completely FIFO and there is no possibility to apply prioritization over the packets when they reach it. This is not a problem in high-speed transmission lines, when transmitting a packet takes less than a millisecond. However, in low-speed lines, the packets stored in the TX-ring can substantially delay a high priority packet.

Consider the following example. A router transmits two kinds of traffic: VoIP and backup data. VoIP are short (100 bytes, 800 bits) and have strict priority over backup packets, which are long (1500 bytes, 12,000 bits). At a given point of time, the VoIP queue is empty and therefore the scheduler serves packets of the low priority queue. The low priority queue may have a large number of packets (let's say 10) but the scheduler makes sure that there are not more than two packet in the Tx-ring. We have a situation in which we have two long packets in the Tx-ring and a VoIP packet arrives to the high priority queue. The VoIP packet has priority and therefore it will be the first one to be served by the scheduler. Still, it will have to wait for the two long packets in the Tx-ring which have already been served by the scheduler.

Transmitting two 12,000 bits packets over a 1Mbps line takes around 24 ms, which is a substantial delay for VoIP packet. To prevent this problem, an interleaver can be used in low speed link. The idea is that long packets are divided in smaller "chunks". Then, the scheduler serves chunks of a packet, instead of the whole packet. This approach reduces the delay and jitter for high priority traffic.

Chapter 6

Scheduling

Scheduling is a key tool in QoS and it is complex enough to deserve its own chapter[1]. A scheduler serves two or more queues and has to take decissions about which queue has to be served first. The combination of queues and scheduler can change the order of the packets. For this reason it is important that all packets of a same flow are mapped to the same queue, to prevent packet re-ordering.

The QoS-aware queues are called software queues. The software queues do not store the actual packets. They just store pointers to the actual packets that are stored in memory. The queue sizes are typically expressed in number of packets. Software queues are placed before the scheduler. The scheduler takes packets from the software queues and places them in the Tx-ring.

Considered as whole, the system of a software queues is not FIFO. The packets do not necessarily leave the system in the same order that they entered. However, each of the queues

[1] A word of warning is needed. In different books, articles and information sources, different names are used for the same scheduling techniques. To add to the confusion, a same technique name has different meanings when looking at different sources.

conforming the system, when considered on its own is FIFO.

The Tx-ring is placed in an area of memory that is directly accessible by the interface hardware. The hardware can continuously send the packets found in the tx-ring without requiring the intervention of the CPU. The tx-ring is FIFO and the packets leave the tx-ring in the same ordered that they entered.

For the scheduling policy to be effective, it is necessary that the tx-ring is relatively small, typically 2 or 3 packets. Otherwise, the queue of packets is build into the tx-ring where the scheduler cannot affect the order in which the packets are sent.

6.1 Strict Priority Queues

Prioritizing is a core idea in QoS. Imagine that a big packet belonging to backup tool such as Dropbox arrives to a router. Right after that, a second packet arrives to the same roter. Imagine that this second packet is a small VoIP packet.

The VoIP packet is in a hurry, because it must make it to the decoder before the playout time. For this reason, the combination of queues and scheduler will allow the VoIP packet to advance the backup packet and be transmitted in the first place.

In the simplest case we have only two queues: the high priority queue and low priority queue. A classifier maps each of the packets to one of the queues. The scheduler will serve the high priority queue whenever there is a packet in that queue. Only when the high priory queue is empty, the scheduler will serve the low priority queue.

Note that this concept can be easily generalized to the case in which there are more than two queues. The principle is that

a queue will be served only when all the higher priority queues are empty.

6.1.1 Preemptive strict priority

In preemptive strict priority, if a high priority packet arrives while a low priority packet is being served, the service of the low priority queue is interrupted and the high priority packet is served immediately. Theoretically, the service of the low priority packet is resumed when the high priority queue becomes empty.

This approach is very beneficial for high priority packets as they are never disturbed by low priority packets. The only problem with this approach is that, in practice, it is not trivial to stop a transmission of a packet and later resume it.

6.1.2 Non-preemptive strict priority

In non-preemptive strict priority, if a high priority packet arrives while a low priority packet is being served, the transmission is not interrupted. The high priority packet will wait for the transmission of the low priority packet to finish and then it will be transmitted.

This is the approach that is used in practice in high speed transmission lines. The problem with this solution is that, in a slow transmission line, a high priority packet might need to wait for a long time if it arrives while a long low priority packet starts being serviced.

6.1.3 Fragmenting and interleaving

An intermediate solution between the two mentioned before is fragmenting each packets in chunks and serving one chunk

at a time. Using this technique we can emulate a behaviour similar to preemptive strict priority. As it has been explained in the previous chapter, fragmenting and interleaving are used in low speed lines.

In higher speed lines there is less benefit in interleaving, as the duration of a packet transmission is very short. Furthermore, the process of interleaving consumes CPU and introduces some processing delay. As the number of packets per second increases, these two negative aspects become more accentuated.

6.1.4 Starvation and policing

The problem of strict priority queueing is that the high priority queue can easily eat up all the bandwidth. If there are always packets in the high priority queue, the other queues will never be serviced. This might be an undesirable situation.

In the following we will review other queueing strategies that avoid this problem. Nevertheless, priority queueing is still used for real-time traffic such as VoIP, as it minimizes the delay for high priority traffic. The solution to prevent starvation, is two police the high priority queue.

The high priority traffic is policed before entering the queue to a fraction of the total bandwidth available in the interface (e.g., 10% or 30%). By doing so, the other queues are safe from starvation. Even if a virus, a worm, or a misconfigured device generates a low of high priority traffic, it will not kill the network.

Another reason for policing a queue with strict priority is to provide bounds on delays. The maximum delay suffered by the packets is the depth of the token bucket divided by the line rate.

6.2 Round Robin Scheduling and Weighted Round Robin

One of the most direct ways of sharing the available bandwidth among different queues is to serve one packet of each queue at a time in order. As an example, if there are three queues, a round robin (RR) scheduler will serve Q1, Q2 and Q3. And then again Q1.

If one of the queues is empty, it is simply skipped. This property is called the work-conserving property, and guarantees the full utilization of the link while there are packets waiting.

If we want to prioritize Q1 over Q2 and Q3 and Q2 over Q3, we can configure the scheduler to serve the queues in this order: Q1, Q2, Q1, Q3, Q1, Q2. And then repeat the same schedule again. Note that with this particular schedule, Q1 is served thrice in each round, Q2 is served twice and Q3 is served once. This particular approach in which different queues have different weights is called Weighted Round Robin (WRR)

Round robin is flexible and precise when it comes to adjust the number of packets transmitted by each queue. The actual bandwidth allocated to each queue depends on the schedule, but also on the length of the packets.

Recovering our example of three queues, if the packets in Q3 are 1200 bytes long and the packets of Q1 are 200 bytes long, Q3 will receive twice as much bandwidth as Q1. If we want to allocate a fraction of the available bandwidth to each of the queues and we don't know the length of the packets in advance, it will be impossible to use WRR.

6.3 General Processor Sharing

General Processor Sharing is an idealized sharing approach that assumes a fluid model. It assumes that the scheduler can serve an infinitesimal amount from each of the queues. It is easy to imagine with the example of liquid reservoirs instead of queues, and pipes draining liquid from each of the reservoirs. One pipe may be draining one liter per hour and the other two liter per hours. Still, at any given point of time, both reservoirs are drained simultaneously, even though one is served twice as fast as the other.

In the practice of packet networks, it is not possible to implement this idealized model. An approximative implementation is used and we explain it in the next section.

6.4 Deficit Weighted Round Robin

In deficit Weighted Round Robin (DWRR) each queue has an associated "token bucket". It is called the "deficit counter", and it is measured in bytes. The scheduler visits all the queues, one by one. In every visit, the scheduler increases the deficit counter by a value which is called "quantum". If the value of the deficit counter after being increased is larger than the size of the first packet in the queue, the packet is served. The deficit counter is decreased by a value equal to the size of the packet that has been served. If, after being decremented, the deficit counter is larger than the size of the following packet, another packet is served and the deficit counter is decremented accordingly. This process is repeated until the deficit counter is smaller than the first packet of the queue or the queue is empty. In the former case, the value of the deficit counter is saved for the next round. In the latter case, the value of the deficit counter is reset. At this point, the scheduler moves on

to serve the next queue.

By adjusting the quantum of each queue, we can set the weight (in terms of bandwidth) of each of the queues. As an example, if we set a quantum of 400 bytes for the high priority queue and a quantum of 200 bytes for the low priority queue, the high priority queue will receive twice as much bandwidth.

Let's look at an example. The high priority queue contains a 500 bytes packet, and the low priority queue a 400 bytes packet. Both deficit counters are zero. In the first round, the deficit counters are incremented to 400 and 200, and no packet is served as the values of the counters are lower than the size of the packets. In the second round, the counter of the high priority queue is incremented to 800. The first packet is serviced and de counter is decremented to 300. While the first packet is being serviced, a 100 bytes packet and a 500 bytes packet arrive to the high priority queue. When the service of the first packet finishes, the deficit counter of the first queue (300 bytes) is larger than the size of the head-of-queue packet (100 bytes). Consequently, the head-of-queue packet is served and the deficit counter is decremented to 200. At this point, the counter is smaller than the size of the head-of-queue packet and therefore the scheduler jumps to the next queue.

It increases the counter of the second queue by 200, and the resultant value (400) is equal to the size of the head-of-queue packet of the second queue. This packet is being served and the counter decremented.

The scheduler moves to the first queue again, it increases the counter by the quantum (400) to 600. Then it serves the head-of-queue packet, which has 500 bytes, and decrements the deficit counter by 500.

6.5 Actual Implementations

Actual implementations are very different to the simple and structured material presented in this chapter. The implementations that we find in practice combine many different elements such as classifiers, policers and schedulers in a single tool.

6.5.1 Stochastic Fair Queueing

Strictly speaking, this is not a scheduling policy. It is more a classification approach. However, as it is usually presented as a queue, it is worth mentioning here. In stochastic fair queueing a large number of queues is used, e.g. 1024. Then, for each packet, a hash from the IP addresses, layer 4 protocol and port numbers fields is derived. This has is used to place the packet in one of the queues. This ensures that all the packets of the same flow are in the same queue. Finally, some kind of scheduling discipline (such as RR or DRR) is used to serve the queues. This approach favors flows with less traffic and therefore benefits interactive applications such as SSH.

It is still possible that, by chance, two flows end up in the same queue. To prevent that this detrimental situation affects the flows for a long time, a perturbation to the hash is periodically introduced to randomly compute new queues for all the flows.

6.5.2 Some Cisco Queues

Cisco's Weighted Fair Queueing

Cisco's weighted fair queueing is used in low speed interfaces of Cisco routers by default. This scheduling algorithm is similar to the above mentioned fair queueing but it uses the concept of

sequence number for its implementation. It takes into account both the size of the packets and the Type of Service field of the IP header to compute a sequence number that is used to decide the order in which the packets will be transmitted.

Cisco's Class-Based WFQ

This scheduler considers different classes and it is possible to reserver a fraction of the total bandwidth for each class. The bandwidth reservation can be either an absolute value or a percentage.

Cisco's Low-latency Queueing

This a strict priority tool that can be combined with CBWFQ to offer better delay and jitter guarantees to real time traffic. The strict priority queue is policed to prevent the starvation of the other queues.

6.5.3 Some Linux Queues

Linux's PFIFO_FAST

This is the default discipline. It contains three queues that use strict priority and packets are placed in different queues according to their Type-of-Service field in the IP header. We can see the queue length with the command

```
cat /sys/class/net/eth0/tx_queue_len
```

Linux's Stochastic Fair Queueing

It uses 1024 different queues. The parameters include a *limit* which is the size of each of the queues, a *perturb* which determines how often the hashing process will be changed and a

quantum, which is the amount of bytes to be send from each queue in each round.

6.5.4 Linux's Hierarchical Token Bucket

It allows setting the bandwidth reserved for each class of traffic (*rate*). Also the maximum amount of bandwidth consumed by a class of traffic (*ceil*). It also offers the possibility of configuring a hierarchy to distribute the bandwidth that is unused by one of the traffic classes.

Chapter 7

Active Queue Management

In Chapter 4 we mentioned that some of the Internet applications are elastic in the sense that they make use of all the available bandwidth. The interesting part is that these application don't know in advance how much bandwidht is available, which means that they have to find it out somehow. In this chapter we describe how TCP congestion control and Active Queue Management (AQM) work together to throttle bandwidth hungry applications.

The interplay between TCP and AQM is critical to deliver satisfactory performance. In 2009, mainstream media[1] reported performance problems in AT&T's wireless data network. The alleged cause was the increase of data usage due to the popularity of IPhones. The underlying reason seamed to be a poor buffer dimensioning[2].

[1] E.g. www.nytimes.com/2009/09/03/technology/companies/03att.html

[2] `http://blogs.broughturner.com/2009/10/is-att-wireless-data-congest`

7.1 TCP congestion control

We will introduce the working principles of TCP. This is not meant to be an accurate description of TCP, just an overview of the concepts used for congestion control. TCP is indeed very complex, heterogeneous (different in different OS), and continuously evolving. An accurate description of TCP is beyond the scope of this course.

TCP is a flow-oriented transport (layer 4) protocol that is used by the elastic applications in Internet. This includes most of the applications, such as web browsing, email and file transfer. Take the example of a large file transfer using FTP, which involves the transmission of a large amount of data. TCP doesn't know how much bandwidth is available, so it will start transmitting two MSS worth of data. This is the initial value of the contention window (CW) which is the maximum amount of in-flight unacknowledged data is permitted. The CW is increased in one MSS for each received acknowledgement. This process is called the slow start phase of TCP. Even though it starts slow, the truth is that it is growing exponentially and, at some point, the transmitting rate will exceed the available capacity and a packet will be dropped.

When TCP detects that a single packet has been dropped (signaled by a duplicate ack), it halves its congestion window and starts increasing it linearly at a rate of one MTU for RTT. This phase is called congestion avoidance, as TCP slowly grows its congestion window trying to find the limits of the available capacity. If another packet loss occurs which results in another duplicate ack, the congestion window is halved again. The result is that the sending rate of TCP approaches the available capacity.

If the loss of a packet is detected due to a timer timeout, TCP assumes that something is very wrong with the network and moves back to the initial slow start behaviour.

In summary, TCP will always try to increase its sending rate. Only when a packet is lost, TCP will reduce its sending rate as the packet loss is interpreted as a sign of congestion. Therefore, in principle, it is necessary to drop packets to signal TCP to lower its sending rate.

7.2 Long Fat Networks

As the bandwidth of the Internet lines increase, TCP needs to keep up with this progress and be able to fill this pipes. If you have a high-speed connection (say 1 Gbps) and a long end-to-end delay (100ms), you need a lot of data to fill it. In this particular example, 100 Mbits of data are needed to fill the pipe.

If TCP does not fill the pipe, and pushes only 1 Mbit of data in it, then it is wasting 99% of the available bandwidth. For this reason, modern operative systems use techniques (basically, window scaling) that make it possible to push much more data into the pipe than the original TCP, which allowed only for 64 Kbytes.

7.3 Bufferbloat

It can be tempting to dimension a buffer so that there is no packet loss. However, increasing the buffer size can increase the delay to unacceptable limits. This problem seems to be prevalent in today's Interned, and has been termed "bufferbloat".

7.3.1 Good Queues and Bad Queues

Queues play a critical role in data packet networks. They absorb a temporal burst and smooth it out for transmission over a line. The burst can be caused by bursty sources and also by statistical multiplexing. A good queue absorbs the burst, fills up, and then empties completely.

But there are also bad queues. Bad queues are full for a long time. Note that a queue that is full for a long time is not useful for absorbing bursts (as it is full and cannot absorb anything) and it has the negative effect of increasing delay. Bad queues appear frequently when there is a high-speed to low-speed transition.

A good example is a 1 Gbps home network that connects to an ADSL which can upload 2Mbps. If you start to upload a large file to the Internet, TCP will do its best in "filling the pipe", and the effect of that action would be to fill the buffer of the ADSL router. If that router has a large buffer, it will introduce a lot of delay. TCP will believe that it is dealing with a "long pipe", as the delay is long, and will persist in the efforts of filling it.

The result is an artificially bloated delay, which not originated by the long distance, but by the large full buffers. A large full buffer does not provide any benefit to the network, other than unnecessarily increasing the delay. Delays of several seconds have been reported due to bufferbloat.

It is important to realize that large buffers may have the effect of penalizing the network performance.

7.4 Taildrop and Weighted Taildrop

In the good old times, high speed memory was expensive and therefore the amount of it in routers and other switching de-

vices was limited. Consequently, the queues were not very large because there was no room to store a lot of data in the router. As memory got cheaper, manufacturers beefed up their devices with more memory. This turned out not to be an improvement on the network performance, as larger buffers resulted in "bufferbloat".

It is obvious that it is good to keep queues to a limited size, e.g., to avoid infinite queues. A first possibility, is to have a queue size and drop packets that arrive when the queue is full. This approach is called taildrop, as packets are dropped from the tail of the queue.

The size of the queue can be measured in packets, in bytes or in milliseconds. To make the measure in time, we need to take into account the speed of the interface. This is normally a good idea, as the amount of data that needs to be buffered is closely related to the speed of the line. Expressing the queue length in time also gives a direct idea of the delay that the queue can introduce.

As an example, a 10 ms queue size seems to be a fair size for many applications. A 10 seconds queue is probably excessive for many applications, such as web browsing.

In the simplest approach, all the packets arriving to a taildrop queue are subject to the same restrictions. If the queue is full, the packet is discarded. Otherwise, it is accepted.

It is possible to refine this approach taking into consideration QoS differentiation. Imagine that in a previous stage we have marked packets as being either in-contract or out-of-contract. Then we can set two different queue sizes for different packets. We can accept in-contract packets if the queue size is below 10 ms and out-of-contract packets if the queue size is below 5 ms. In this case, out-of-contact traffic has a higher dropping precedence that in-contract traffic, and we lessen the extent to which in-contract traffic can be delayed

(or even dropped) due to out-of-contract traffic.

The technique that uses different queue sizes for different packets is called weighted tail drop.

A problem of taildrop is that, for the dropping signal to reach the sender, it takes a long time. While the buffer has packets, they will be transmitted and the receiver will continue sending positive acks.

Another problem is that it is likely that a few consecutive packets are dropped. Consecutive packet loss increases the chances that TCP time-outs, which in general are much more detrimental than the loss of a single packet. Even worse, when the buffer is full, several TCP flows sharing the same buffer may time-out almost simultaneously, leading to the problem of TCP global synchronization described in the next section.

7.5 TCP global synchronization

TCP global synchronization occurs when a tail-drop queue accommodates packets of several TCP sessions. The TCP sessions keep increasing their congestion windows and transmission speed until the queue fills up. At this point, the tail-drop queue drops all the arriving packets.

All the TCP sessions detect packet loss and halve their sending rate (or reset it) simultaneously. As a result, the queue empties and then the link remains unused for some time. Then the TCP sessions starts growing again, and the story repeats one more time.

Observe that bufferbloat and TCP global synchronization are complementary problems. In the former, the queues are never empty. In the latter, the queue is empty for a substantial fraction of time. Ideally we would like that there was always a packet ready to be transmitted, to take full advantage of the

link capacity. Simultaneously, we would like that there was only a single packet, as the presence of many packet introduces unnecessary delay.

Active queue management (AQM) addresses tries to address these problems.

7.6 Random Early Detection

A first approach to prevent that queues fill up and the bufferbloat and TCP global synchronization problems appear, is to start discarding packets before the queue is full. The general idea is to accept all packets when the queue is empty (or almost empty), some packets as the queue starts to fill up, and no packets when the queue is full (which should not happen). Packet loss is induced deliberately before the queue is full to signal the involved TCP flows that they should reduce the sending rate. The big difference is that not all the flows receive this signal simultaneously. With RED, the packet drop is distributed evenly in time, so that the TCP halve their transmission rates at different instants and therefore the aggregated sending rate evolves smoothly in time.

By avoiding the sharp drop of the aggregated sending rate in TCP global synchronization, it is possible to achieve a better utilization of the available bandwidth[3].

The first step to implement RED in a router is to compute the average delay[4]. This is computed using an exponential weighted moving average with parameter w. The current average q_{avg} is computed using the previous average value q_{prvavg}, the last measured value $q_{measured}$ and the parameter w as fol-

[3]Cisco provides a figure supporting this statement in `http://www.cisco.com/image/gif/paws/10582/60c.gif`

[4]Note that recent papers point out that average queue delay is not a relevant metric to differentiate between good queues and bad queues [10].

lows.

$$q_{avg} = \frac{2^w - 1}{2^w} q_{prvavg} + \frac{1}{2^w} q_{measured} \tag{7.1}$$

If q_{avg} is below a minimum threshold (q_{min}) the packet is always accepted. If q_{avg} is larger than the maximum queue length q_{max}, the packet is rejected. In the case that $q_{min} \leq q_{avg} \leq q_{max}$, the packet is accepted with probability

$$P[packetaccepted] = \frac{q_{avg} - q_{min}}{q_{max} - q_{min}} P_{max}, \tag{7.2}$$

where P_{max} is yet another configuration parameter that determines the dropping probability when q_{avg} approaches q_{max} from the left. This curve is illustrated in Fig. 7.1.

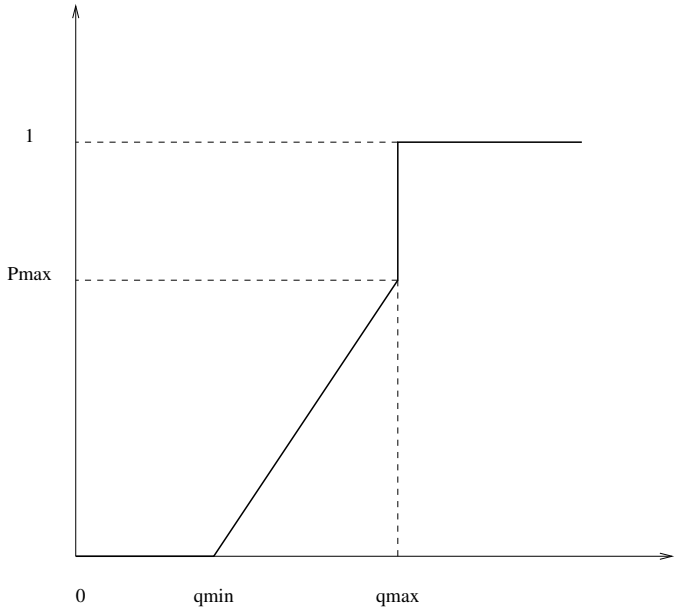

Figure 7.1: Dropping probability as a function of average queue delay in RED.

It is straightforward to generalize RED to weighted RED in

which different packets use different profiles (different configuration parameters) in order to prioritize some packets over the others. Typically, packets with higher dropping precedence are exposed to more aggressive RED profiles.

7.7 Explicit Congestion Notification

It is sad to drop packets when they are half-way towards their destination, but in principle it is the only mechanism to signal TCP that it has to reduce its sending rate. There is an alternative, introduced in RFC 3168 [11] which allows notifying TCP without dropping packets.

First it is necessary that both communication endpoints can support ECN and agree to use it. If they do, they will use the least two significant bits of the DiffServ field to indicate it. These bits will be set to either 10 or 01 if ECN is being used, and 00 otherwise.

If a router that supports ECN decides to drop a packet, it will first check whether the packet belongs to a flow that supports ECN. If it does, it will set the ECN bits of the packet to 11 and will not drop the packet. The receiving host will use the Explicit Congestion Echo (ECE) flag to notify the sender. The sender will reduce the congestion window and activate the Congestion Window Reduced (CWR) flag. The receiver will keep ECE activated in all the transmitted packets until receives the CWR from the sender.

Upon reception of this ECE, the originating host will reduce its congestion window (and thus its transmitting rate) just as if a packet dropped has occurred. The advantages are that there is no need for packet re-transmission and that the reaction time is shorter. This way, it is possible to achieve congestion control without packet dropping.

A particularly noticeable improvement is perceived in inter-

active protocols such as HTTP, telnet and SMPTP. If the last packet (and probably single) packet of a message is dropped, the RTO expires and a retransmission occurs. This retransmission can be avoided using ECN.

Despite all the advantages of ECN, it is not widely used yet. The reasons are that, for being effective, it is necessary that both endpoints and also the network equipment supports ECN. Furthermore, there are broken implementations in networking devices that do not recognize ECN and simply drop the packets that include 10 or 01 in the ECN field.

7.8 CoDel

The latest trend in AQM is Controlled Delay (or CoDel) [10], which discards packets when the minimum queue occupation exceeds a threshold (in time). This technique seem to be easier to configure and provide better link utilization while preventing bufferbloat.

Chapter 8

Differentiated Services

The IETF has proposed two main solutions for the Internet QoS architecture: Integrated Services and Differentiated Services [4]. The first one implies per flow signaling and reservation, and therefore it does not scale well in large network. The focus of this chapter is on the second alternative. In differentiated services, traffic is classified in a reduced number of classes, typically less than eight. The routers implement different PHB for the different classes, which provide the desired QoS.

Note that since in Diffserv there is not a per flow signaling and admission control, in principle it does not guarantee that the QoS requirements are actually satisfied. In reality, this is achieved with a combination of traffic conditioning at the edge of the network and capacity planning. This is not an exact science, but we'll cover the basics through examples and typical configurations.

8.1 Key aspects of DiffServ

DiffServ is an scalable solution for QoS provisioning in IP networks. DiffServ offers the tools for network administrators to implement QoS in an IP domain. This can be used to offer VPNs that satisfy SLAs and to support services with tight QoS requirements such as VoIP.

The interconnection of different DiffServ domains is somewhat more complicated, as needs the collaboration of the different organizations that manage the different networks. DiffServ is not commonly used for generic Internet access, as the Internet involves a humongous collection of networks which makes the provision of QoS guarantees an enormous challenge.

Three key aspects of DiffServ are border traffic conditioning, DiffServ markings, and PHBs.

8.1.1 Traffic Conditioning

Together with the SLA, a traffic conditioning agreement (TCA) is also established. The TCA is often defined as a packet marking combined with token bucket, of a given rate and depth. For example, it may specify that packets with VoIP traffic will conform to a token bucket of rate 1Mbps and depth 1Kbps.

At the edge of the network, the routers will perform classifying, metering and policing to ensure that the offered load complies with the specified TCA. The traffic exceeding the TCA may be either dropped or marked as non-conformant.

As the tasks of classifying, metering and policing require computational resources, they are done only at the edge. At the edge of the network line speeds are lower than in the core of the network, and therefore it is possible for the router to process all the packets. In the core of the network, the switching speeds are much higher and the time required to process each

packet should be minimized, and therefore there is no time for classification, metering and policing. As the edge of the network grows when the network grows, this solution is scalable and therefore appropriate for large networks with large volume of traffic.

According to the initial classification, the edge router will mark the packets using a field in the IP header which is the DiffServ Code Point (DSCP).

8.1.2 DiffServ Code Point

The DiffServ is a 6 bits field in the IP header that is assigned to each packet in the edge of the network. The DSCP assigns a packet to one of the possible traffic classes supported by the network. There are some standardized values for this field, such as Expedited Forwarding (EF), that we will review later. But its value is of significance only within the DiffServ domain and therefore the network administrator have the decision of which DSCP values are used and what is the meaning of each of them. This marking can be used later by the other routers of the network to quickly classify the packet, and treat the packet in accordance of the traffic class to which it belongs. This treatment is the per-hop-behaviour (PHB).

8.1.3 Per Hop Behaviour

For each of the traffic classes (DSCP values) supported by the network there has to be an associated PHB. This can be a many-to-one mapping, as different DSCP values can be offered the same PHB. The PHB is a high level description of the service that a packet will receive when it arrives to a router. It does not detail the QoS tools that will be used to implement that behaviour.

8.2 IETF defined per-hop-behaviour

The IETF has provided recommendations for four PHB. A recommendation combines a description of the PHB with a suggested DSCP.

8.2.1 Default PHB

It is a required behaviour and it is assigned the DSCP 000000BIN. RFC 4594 [1] specify that there is typically some bandwidth guarantee for this kind of traffic, but the packets can be lost, reordered, duplicated or delayed at random.

8.2.2 Expedited Forwarding (EF)

Expedited Forwarding (EF) is defined in [6] and uses the code point 101110BIN (46DEC). The purpose of this PHB is to serve real-time traffic such as VoIP or videoconferencing. It should provide guaranteed bandwidth with low delay, jitter, and packet loss.

This can be attained by using strict priority queueing in combination with a buffer that is large enough to accommodate the traffic bursts. The traffic is policed to control its rate and burstiness. The rate needs to be controlled to ensure that this priority traffic does not starve other classes of traffic. The bucket depth needs to be smaller than the allocated queue space, to guarantee that no packet loss will occur due to full queues. The depth is also important as it determines the maximum delay that a packet may suffer, which is the token depth divided by the transmission rate. The maximum delay obviously also influences the maximum jitter that the packets will suffer.

8.2.3 Voice Admit (VA)

Similar to EF, with the difference that Voice Admit (VA, [3]) is subject to a call admission control procedure.

8.2.4 Assured Forwarding (AF)

Assured Forwarding is actually a group defined in RFC 2597 [7] that comprises 12 DSCPs. The packets belonging to this group will be forwarded if the offered rate is kept below the committed rate. It differentiates four different classes: AF1x, AF2x, AF3x and AF4x. AF1x is the lowest priority and AF4x is the higher priority. A typical implementation is to assign each of the classes to a queue and use DRR to distribute the bandwidth among the classes.

Within each of the classes, there are three possible drop precedences. As an example, we have AF11, AF12 and AF13. This means that, if congestion occurs, AF12 packets will be dropped with higher probability than AF11 packets. And AF13 packet will be dropped with higher probability than AF11 and AF12 packets.

A typical scenario is to use a two-rate meter within a class. If we take AF1x as an example, in-contract traffic is assigned to AF11, out-of-contract traffic is assigned to AF12, and traffic exceeding the PIR is simply dropped.

AF11 and AF12 will be mapped to the same queue. This is an important aspect as otherwise packet re-ordering may occur. Then, different RED profiles will be applied to AF11 and AF12. AF12 will suffer a more aggressive RED profile to ensure that out-of-contract packets are the first to be dropped.

RFC 2597 [7] recommends the following DSCP values for AF.

	Class 1	Class 2	Class 3	Class 4

Low Drop Prec	001010	010010	011010	100010
Medium Drop Prec	001100	010100	011100	100100
High Drop Prec	001110	010110	011110	100110

8.2.5 Class Selector (CS)

This was included to provide some kind of backward compatibility with pre-DiffServ approaches. The first three bits (0-2) to indicate precedence: higher value means higher precedence. A practical use is to ease mapping from IP QoS markings to EXP QoS markings. The recommended DSCPs take the form XXX000BIN.

8.2.6 DSCP recommendations

RFC 4594 [1] offers recommendations about which DSCP to use for different kinds of traffic. Normally, a network administrator will choose only a subset of those DSCP, according to the traffic that needs to be supported in a particular network.

Service Class Name	DSCP Name	DSCP Value	Application Examples
Network Control	CS6	110000	Network routing
Telephony	EF	101110	IP Telephony bearer
Signaling	CS5	101000	IP Telephony signaling
Multimedia Conferencing	AF41,AF42 AF43	100010,100100 100110	H.323/V2 video conferencing (adaptive)
Real-Time Interactive	CS4	100000	Video conferencing and Interactive gaming
Multimedia Streaming	AF31,AF32 AF33	011010,011100 011110	Streaming video and audio on demand
Broadcast Video	CS3	011000	Broadcast TV & live events

Low-Latency Data	AF21,AF22 AF23	010010,010100 010110	Client/server transactions Web-based ordering
OAM	CS2	010000	OAM&P
High-Throughput Data	AF11,AF12 AF13	001010,001100 001110	Store and forward applications
Standard	DF (CS0)	000000	Undifferentiated applications
Low-Priority Data	CS1	001000	Any flow that has no BW assurance

The same RFC offers recommendation about edge conditioning, PHB, scheduler configuration and AQM.

Service Class	DSCP	Conditioning at DS Edge	PHB Used	Queuing	AQM
Network Control	CS6	See Section 3.1	RFC2474	Rate	Yes
Telephony	EF	Police using sr+bs	RFC3246	Priority	No
Signaling	CS5	Police using sr+bs	RFC2474	Rate	No
Multimedia Conferencing	AF41 AF42 AF43	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP
Real-Time Interactive	CS4	Police using sr+bs	RFC2474	Rate	No
Multimedia Streaming	AF31 AF32 AF33	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP
Broadcast Video	CS3	Police using sr+bs	RFC2474	Rate	No
Low-Latency Data	AF21 AF22 AF23	Using single-rate, three-color marker (such as RFC 2697)	RFC2597	Rate	Yes per DSCP
OAM	CS2	Police using sr+bs	RFC2474	Rate	Yes
High-Throughput Data	AF11 AF12 AF13	Using two-rate, three-color marker (such as RFC 2698)	RFC2597	Rate	Yes per DSCP

Standard	DF	Not applicable	RFC2474	Rate	Yes
Low-Priority Data	CS1	Not applicable	RFC3662	Rate	Yes

8.2.7 Differentiated Services in MPLS

An MPLS header contains: a label value (20 bits), an experimental EXP value used for QoS (3 bits), a bottom-of-stack flag (1 bit) and a TTL field (8 bits). There are only three EXP bits for QoS markings, compared to the six available in IP. If the "class selector" PHB are used, which use only the first three bits of the six available in DSCP, it is very easy to map from DSCP to EXP and the other way around. All is needed is to copy those three bits when an IP packet enters the MPLS domain.

If "class selector" is not used, but there are less than eight different classes, it is still possible to map from DSCP to EXP. This would be the case if AF11, AF12, AF21, AF22, AF31, AF32, AF41 and AF42 are used. The approach in which the information about the traffic class is contained in EXP bits is called EXP inferred PHB selection (E-LSP, where LSP stands for label switched path).

If the classification used has more than eight different DSCP values, a first option is to opt for a many-to-one mapping. As an example if the eight classes mentioned above are used and in addition also EF is necessary, a possible option would be to treat AF1x packets just as AF2x packets.

An alternative to the many-to-one mapping is to use the label for QoS. In our example, we could use five different labels. One for each of the AF1x groups and the fifth one for EF. When the LSP identifier carries QoS information it is called label inferred LSP (L-LSP).

Chapter 9

RSVP and MPLS-DiffServ-TE

In this chapter we will present an overview of a set of technologies that offer a tight control on QoS. This control comes at the price of complexity, but the ISPs and the manufacturers have decided that QoS is worth it. This technologies offer control on the path followed by the packets, bandwidth reservation (and therefore guarantees) and fast re-route capabilities.

9.1 Multi Protocol Label Switching (MPLS)

In the IP paradigm, the routers forward the packets based on the destination IP of the packet. MPLS offers a completely different alterative, in which the forwarding is not done using the destination address. Instead, labels are used at each hop to decide the outgoing interface. This approach is called "virtual circuit packet network" as it somehow emulates a circuit on a packet network, and it is opposed to the "datagram" IP

paradigm. The virtual circuit approach has some advantages and disadvantages compared to the datagram approach.

In the MPLS jargon, a virtual circuit is called a Label Switched Path (LSP). Each of the packets has a header which is called "the label". In fact, a packet may contain multiple labels and it is said that they are "stackable". The last label can be found on top.

Upon reception of a packet, the outermost label is inspected. Each router has a lookup table indicating, for every incoming label, the outcoming label and the outcoming interface. The router simply performs the lookup, pops the outermost label and pushes a new label before forwarding the packet. To populate the lookup table, the LSP is established before starting forwading packets.

In MPLS, the routers are called Label Switch Routers (LSR) and the edge routers Label Edge Routers (LER). The LSP is established between two LER and then all the packets in the LSP follow exactly the same path.

MPLS is protocol agnostic in the sense that can carry any kind of packets. Examples are IP packets and Ethernet packets. This functionality can be used to create both Layer-3 and Layer-2 virtual private networks. MPLS simply sticks a label on top of any packet (ethernet, IP, etc.) in the LER which is the entry point of the MPLS domain. This MPLS packet is forwarded by MPLS by looking only to the label. The label is removed (popped) at the last or penultimate hop of the MPLS domain.

One of the advantages of MPLS is that the same core can be used to transmit any kind of data. Another advantage is that it makes it possible to control the path that the packets follow. The engineering of the paths that the data streams follow within the data networks is called traffic engineering (TE).

9.2 Traffic Engineering

Consider the example scenario in Fig. 9.1. All the links are of 100 Mbps and a propagation delay of 10 ms. The only exception is link $B - C$ that has a larger delay. Imagine that you have to carry two streams from router A to router D. One of the streams is a 25 Mbps VoIP stream that requires low delay, jitter and loss. The second stream is a 80 Mbps stream for remote backup that does not have tight delay and jitter requirements.

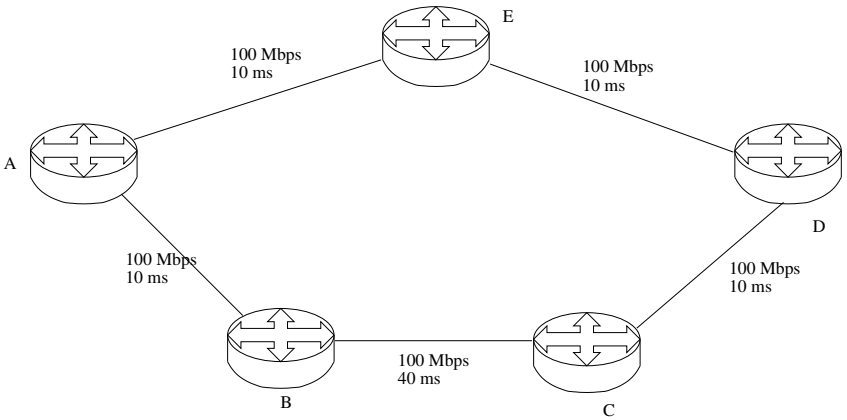

Figure 9.1: Example scenario to explain traffic engineering.

The best route between A and D is $A - E - D$. It has the same bandwidth, less hops, and less total delay than the alternative path $A - B - C - D$.

A datagram-oriented network would choose the best route for all the packets. Notice that this might not be the best of ideas. If we direct a total of 105 Mbps (25 of VoIP + 80 of backup data) to 100 Mbps links, queues will build up, and delay, jitter and packet loss will be high.

A better alternative is to direct the 25 Mbps of VoIP traffic to the short path $A - E - D$ and the 80 Mbps of backup traffic

to the long link $A-B-C-D$. Using this solution, we make a better use of our network resources. All links are utilized, and they are all below 100% utilization. This solution provides lower delay, jitter and packet loss for all classes of traffic, as the queues will not build up.

This distribution of data streams in the network is called traffic engineering (TE). A reason to engage in TE is to balance the load among different parts of the network to prevent that some links are overloaded while others are empty. TE is also useful to prevent that delay sensitive streams cross high-latency links (such as satellite links). Finally, if combined with bandwidth reservation, it can provide guarantees which are important to meet the SLAs that are common in virtual private network services.

9.3 Virtual Private Networks using MPLS

It is common that companies have different sites. Imagine that a company has a site in the city of Girona and another site in the city of Tarragona. This company wishes to have a single network shared among the two sites. The option of deploying a fiber to connect the two sites might be an overkill. Another option might be to approach an ISP and pay for a "virtual connection" between the two sites.

This virtual connection is described in terms of a rate, a burst size and a delay. The company can push a packet in Girona, and it will "magically" appear in Tarragona. In reality, the ISP will establish a LSP between the two sites. The ISP takes the packet received in Girona, sticks a label on to the packet, and it switches it through the network until it reaches Tarragona. Note that it does not matter what the packet is. Whatever it is, it will be taken from one site and delivered to

the other.

From the company's perspective, it is exactly the same as having a pipe that connects the two sites. In order to guarantee a given bandwidth and delay, the ISP must make a reservation in all the packets that are traversed by the LSP from Girona to Tarragona. This reservation has to be in terms of bandwidth and buffer space to accommodate the burst size. The protocol used to make the reservation for the LSP of the VPN is called "Resource Reservation Protocol for Traffic Engineering" (RSVP-TE).

9.4 Bandwidth reservation and RSVP-TE

When a new LSP needs to be established, the first step is to decide the path that it will take. This step is performed with the constrained shortest past first (CSPF) protocol. CSPF is very similar to the popular open shortest path first (OSPF) protocol. The only difference is that in CSPF we can apply some constraints to prune the tree.

This constraints can be placed in terms of bandwidth availability or "colors". The "colors" simply mark a given property of a link. As an example, a network administrator might decide to mark the high-latency link in Fig. 9.1 as "red". Then, when looking for a path for the VoIP link, CSPF can take into account that "red" links must be avoided. In fact, CSPF prunes all the links that do not satisfy the constraints and then uses OSPF in the remaining topology.

The protocol that is used for bandwidth reservation along the path is RSVP-TE. The LER at the network entry point of the LSP sends a RSVP-TE "path" message to the endpoint. This message is forwarded using CSPF along the path that the

LSP will follow. Each router in the path, adds his identifier to the path message. This information will be used to populate the forwarding table.

When the "path" messages reaches the LSP tailend, the tailend router creates a "rsvp" message that will traverse exactly the same path as the "path" message but in the opposite direction. The "rsvp" message is the one that actually creates the reservation. The only purpose of the "path" message is to mark the forward path. Remember that the paths on a data network are not necessarily symmetrical. For this reason, it is needed that the "path" message if forwarded using the routing protocol from the headend to the tailend. The "rsvp" message is not forwarded using the routing protocol, but with the information gathered by the "path" message.

The "path" and the "rsvp" messages contain the information about the stream, in terms of data rate and bucket size. The "rsvp" makes the actual reservation and it is also used for each router to indicate the label it will use to receive the packets of the LSP. As an example, if a LSP is established through the routers $A - E - D$ in Fig. 9.1, the tailend router D will say in his RSVP message to E: "I am expecting to receive the packets marked with label 123". And E will tell to A: "I am expecting to receive the packets marked with label 121". And E will include an entry to its lookup table with this information: "When I receive a packet through the interface connected to A with the label 121, I will swap the label to 123 and I will forward it to the interface connected to router D". Router E will also reserve the bandwidth and the buffer space required for the stream, and distribute the updated information of its available resources. Note that these actions take place in the control plane. It might be necessary to include a policer in the network entrypoint to ensure that the streams adhere to traffic profile (rate/bucket depth) that they have reserved.

The RSVP-TE protocol has a soft-state, which means that

it is maintained as long as it is refreshed by "path" and "rsvp" messages. If none of these messages is received for a long time, the router cancels the reservation and frees the resources.

The combination of selecting the path (using CSPF) and reserving router resources makes it possible to offer the guarantees agreed upon in the SLA. If no path is found that satisfies the criteria, or one of the routers cannot commit the requested resources (because the CSPF path has been computed with outdated information), the LSP cannot be established.

RSVP-TE includes another ingredient which is the priority of a LSP. A LSP of higher priority can preempt a lower priority LSP. The low priority LSP then has to look for an alternative path. In the creation and destruction of LSPs, the "make before break" rule is always applied, to make sure that no traffic is blackholed.

It is a common practice to assign higher priorities to larger (or more restrictive) streams, as smaller streams (or streams with lesser restrictions) are easier to accommodate. It is like filling a car's trunk. People normally starts by putting the largest suitcases and then fitting the smaller ones in the empty spaces.

9.5 Fast Re-Route

One of the QoS requirements is availability. In case of failure, is it important to restore network services as soon as possible. The kind of traffic with more stringent requirements is VoIP. A network failure that lasts for more than 50 ms might be noticed by the users of a VoIP conversation.

The protocols exchange "hello" messages with their neighbours to detect failures. If a router stops receiving the periodical hello messages from a neighbour it means that something is going wrong. Additionally, RSVP has built in mechanisms

to signal the failure of a path. When the headend of the LSP is notified of the failure, it can re-compute the path when the routing protocol has re-converged.

The only problem is that all this process, specially routing re-convergence, can take more than a second. And we would like to have path restoration below 50 ms. The solution is that, upon failure, a "patch" is applied to the path in the form of a "detour". The packets will be forwarded around the failed node or link until a new LSP is created. The "patched" LSP will be destroyed when the brand new LSP is operative, in a "make before break" fashion.

9.6 MPLS-DiffServ-TE

So far we have not discussed per-hop-behaviours in the context MPLS-TE. If we want to have different classes of traffic and different PHB, is it interesting to take them into account when making reservations. A clear example is the use of EF for VoIP and the rule of not devoting more than 30% of the bandwidth to this class.

In the reservation process, we are not interested on how much bandwidth is available in a node. Instead, we are interested of how much EF bandwidth it is available. MPLS-DiffServ-TE provides the extensions to make the reservations on a per-class basis. Obviously, it is also necessary to distribute the information of bandwidth availability for the different classes.

Chapter 10

Linux QoS

In this chapter we will overview the queues and QoS tools available in linux.

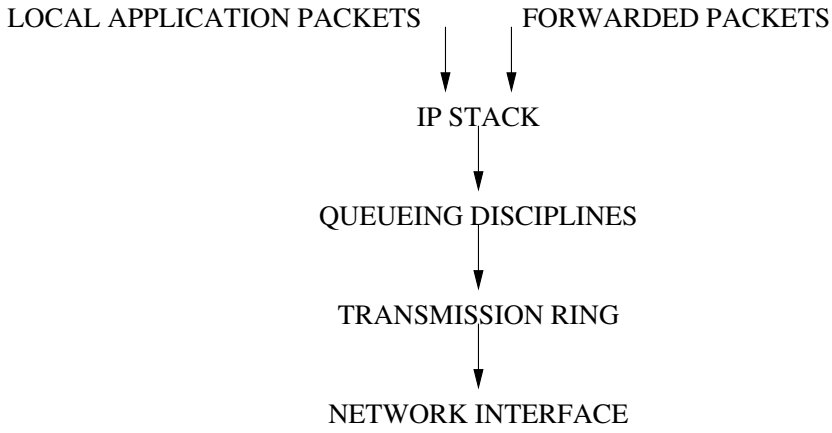

Figure 10.1: Flow of packets through different layers and queues.

As shown in Fig. 10.1, a linux box transmits two kinds of packets. On the one hand we have the packets coming from local applications, such as a web browser. On the other hand, there is traffic that has been received from another interface

and it is being forwarded towards the final destination. The second kind of traffic is present only if the computer is working as a router.

The IP stack places the packets into the *queueing discipline* layer. It is in this layer in which linux can apply QoS tools to the outgoing traffic. Then, the packets are sent to the driver queue (or transmission ring queue), which is a simple FIFO queue in which there is no chance of using QoS tools. Finally, the network interface card takes packet

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 pdf 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. Add also a final trailing space.

The traffic sink takes a port number as a parameter and computes packet delay and packet loss (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. Then these values will no longer be zero.

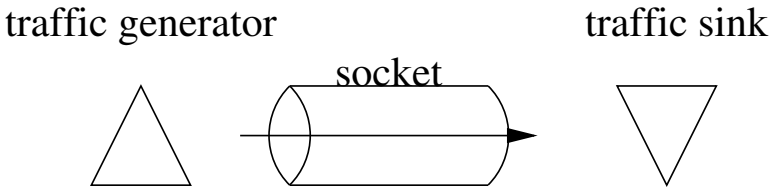

Figure A.1: Scenario to test in lab assignment.

A.1.1 Extra challenges:

Save the inter-arrival time between packets and plot a frequency hystogram. Verify that you obtain a decaying exponential shape.

Test your programs with other groups programs to verify interoperability.

A.2 A queue

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

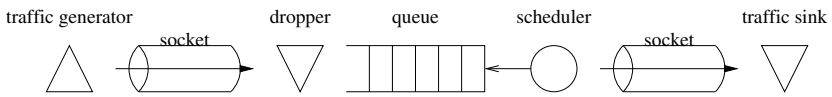

Figure A.2: Scenario to test in lab assignment 2.

Prepare a plot comparing the packet delay and packet loss obtained from the simulation and theoretical results.

A.3 Priority Queues

The third module that you have to construct in this course contains two priority queues (that include a dropper and 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.

After you have completed the basic scenario take some measures and explain the results. Then you can implement also a classifier as in Fig. A.4 and take additional measures.

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

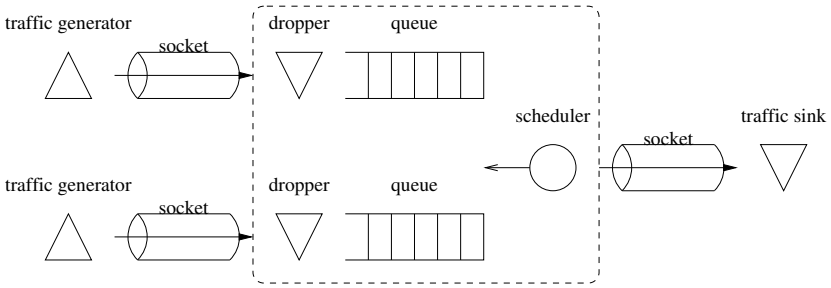

Figure A.3: Scenario to test in lab assignment 3.

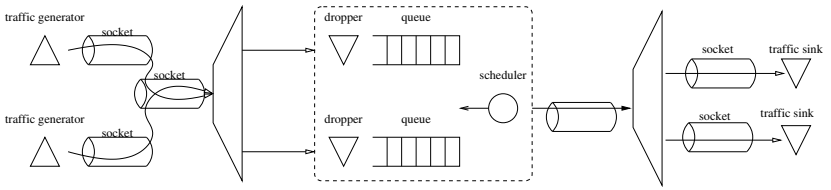

Figure A.4: Advanced scenario to test in lab assignment 3.

delay for each of the traffic classes.

Prepare plots combining analytical results and simulation results.

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 tools that we have seen in class, e.g., metering, token-bucket policer, (weighted) RED, weighted taildrod, 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.

- RED It takes a rate and a size as input parameters. The probability of dropping an arriving packet is equal to the occupancy of queue at the moment of arrival.

- Weighted taildrop It takes a rate and a size as input parameters. The queue only accepts high priority packets when the queue is half-full. 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.

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.

A.6 QoS in Linux

Run a speed test that measures upload/download speed in your linxu box. Then install a Token Bucket Filter. You can use, for example, the following command:

`tc qdisc add dev eth0 root tbf rate 1000kbit latency 50ms burst 1540.`

Explain the results. Remove the qdisc.

`tc qdisc del dev eth0 root tbf rate 1000kbit latency 50ms burst 1540.`

A.6.1 Fair Queeueing

Connect two computers using ethernet. Install `iperf` in both computers. Change the speed of your ethernet interface (as root) to 10Mbps to accentuate the delay suffered by the packets.

`ethtool -s eth0 speed 10 duplex full autoneg on`

Send a ping from one computer to the other to measure the round-trip time. Launch an iperf flow that saturates the link. You can use for the server:

`iperf -u -s`

And for the client

`iperf -c 10.80.25.96 -u -b 10M -t 20`

Observe what happens to the ping RTT.

Now, as root, install fair queueing.

`tc qdisc add dev eth0 root sfq perturb 10`

Repeat the experiment of measuring the RTT with a ping and then launching an iperf test. What happens? Is there any difference compared to the previous experiment?

Remove fair queueing.

```
tc qdisc add dev eth0 root sfq perturb 10
```

The latest releases of Linux include changes that improve the behaviour of TCP. Try the experiment using iperf with TCP. What happens? Why?

A.6.2 Changing the DSCP value

Try to change the DSCP value of packets with a destination port equal to 80 and then observe the changes using wireshark.

```
tc qdisc add dev eth0 root sfq perturb 10
```

Bibliography

[1] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational), August 2006. Updated by RFC 5865.

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

[3] F. Baker, J. Polk, and M. Dolly. A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic. RFC 5865 (Proposed Standard), May 2010.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475 (Informational), December 1998. Updated by RFC 3260.

[5] 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.

[6] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246 (Proposed Standard), March 2002.

[7] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597 (Proposed Standard), June 1999. Updated by RFC 3260.

[8] J. Heinanen and R. Guerin. A Two Rate Three Color Marker. RFC 2698 (Informational), September 1999.

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

[10] K. Nichols and V. Jacobson. Controlling Queue Delay. *Communications of the ACM*, 55(7):42–50, 2012.

[11] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. Updated by RFCs 4301, 6040.