University of Bologna
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Engineering Bologna Campus

Class of
Computer Networks M

QoS basics and protocols

Antonio Corradi

Academic year 2015/2016

Stream Quality of Service

Many indicators and **parameters** to qualify **a stream of information** and its **functional properties**

Promptness in reply

delay, response time, jitter (variation in deliver delay)

Bandwidth **bit or byte per second (per application and system)**

Throughput **number of operations per second (transactions)**

Reliability **percentage of successes / failures**
MTBF, MTTR

Functional aspects (easily measurable) and **non functional**

Many aspects connected to **quality of service** are **non functional** but intertwined with the internal structure of system and specific application and dependent on external factors and observable and judged by final user only

QoS: other INDICATORS

The final user are **the ones to evaluate non functional** properties

- **image details**
- **image accuracy**
- **response time in variations**
- **audio/video synchronization**

the QoS can be guaranteed only through a **negotiated and controlled contract and after provisioning**

By observing the **system during execution** so to adjust **dynamically** the service to current operation conditions and adapting to the environment, by obeying **user** requests
Necessity of observation and feedback

QoS User INDICATORS

The typical **non functional properties requested by a final user** can be:

QoE (Quality of Experience)

- **Relevance (priority)**
- **QoS perceived (details, accuracy, synchronization and audio/video quality)**
- **Cost (per access, per service)**
- **Security (integrity, confidentiality, authentication, non disowning)**

QoS must consider all aspects at different system level and consider all the requirements

The **negotiated SLA must be verified during execution** to undertake **quickly corrective actions**

QUALITY OF SERVICE INDICATORS

Bandwidth (**throughput**): the *quantity of data transmitted by a channel with success per time unit (per second)*

Ethernet 10Mbps (*quantity information/sec*) 10 Mbit per second

Latency time: the *time spent to send an information unit (bit)*

also measured as the round trip time back and forth
*(**R**ound **T**rip **T**ime o RTT)*

$$T_L = T_{prop} + T_{tx} + T_q$$

T_{prop} depends on light **speed** inside the medium (Space / Speed)

T_{tx} depends on **messages** and **bandwidths** (Dimension / Bandwidth)

T_q depends on queuing **delays** in different intermediate points

T_q critical time because it involves all possible waiting overhead

Quality of Service

A good service requires to identify **bottlenecks** and must consider *resource management*

if send/receive of **1 byte** ⇨ latency domination **RTT**

if send/receive of **many Megabytes** ⇨ bandwidth domination

resources occupation: Product **Latency x Bandwidth**

resource data **channel**

latency 40ms and bandwidth 10Mbps → the product is 50 KB (400 Kb)
it is necessary that sender sends **50KB** before that first bit arrives to the receiver and **100KB** before an answer reaches to the sender

Some simple strategies always naively applied

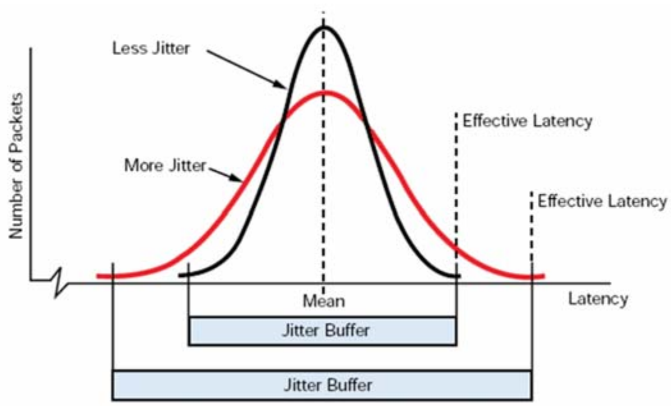
Infrastructures tend to keep pipes full with their sent messages to guarantee response time, **but time must be considered carefully**

A **buffering time inside applications** is typically automatically considered

Quality of Service - Jitter

JITTER defined as **variance** of **latency** in a stream

optimal situation if latency stable, but…



Sometimes, the **SKEW** is also relevant, defined as the possible offset between multiple flows composing a unique stream (for example, in an audio / video stream)

Interest to QoS

In case of **multimedia systems,** or for the distribution of **continuous information flows,**

Video on Demand (VoD) services for distributing streams, provided by an infrastructure Internet compatible

why the interest?

stream of audio and video information with real-time factors: bandwidth, delays, jitter, *variations of admissible delay*

The entities negotiate some quality SLA, for repeated services or frame flows, and tend to respect them by

- impose some initial delay of user provisioning to accumulate frames and to absorb mean jitter

- drop packets that arrive with delay higher than a threshold

QoS in DIFFERENT ENVIRONMENTS

TCP/IP WITH or WITHOUT CONNECTION

the entities communicate using resources available during execution (dynamic) without any predefined commitment

The IP level is responsible for best-effort semantic

IN OSI

the OSI entities commit resources and can also provide SLA, that must be respected from all parties in the path (intermediate nodes)

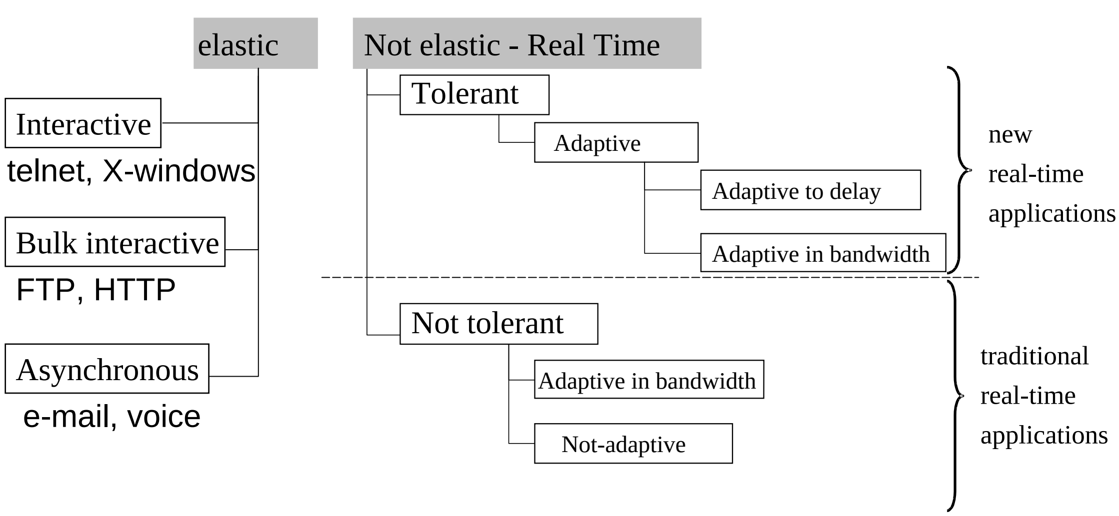
How to guarantee QoS in TCP/IP in best-effort environments?

Users require **new Internet application services**

APPLICATIONS CLASSIFICATION

quality requirements for applications

Elastic and Not Elastic Applications



MORE OR LESS ELASTIC APPLICATIONS

The elastic ones do not present quality constraints but they have different requirements independent from delays

they work better with low delays and work worst during congestions

Interactive with delays less than 200ms

The non elastic have constraints to be respected in time

less tolerant to be usable outside their allowed admissibility space (failure)

they should not work in those cases

The service can be **adaptive to requirements in two ways**

delay adaptive → **audio drop packets**

bandwidth adaptive → **video that adapt quality**

QoS MANAGEMENT

Good management can be granted by actions that must be active for the whole service time

Actions must be **both proactive** (before content distribution and in a preparatory phase) **and reactive** (during deployment)

both static (proactive), and dynamic (reactive)

Static actions

decided and negotiated before distribution

Dynamic actions

identified during distribution

It is necessary to define precise management models

monitor and quality models

QoS MANAGEMENT: STATIC PHASE

Static actions **Before distribution**

requirements definition and allowed variations

Precise specification definition of QoS levels
Definition of **Service Level Agreement** (**SLA**)

negotiation

Agreement between all entities and levels interested to grant QoS

admission control

Comparison between requested QoS and possibly offered resources to provide the service

reservation and **commitment** of required resources

Needed resources definition for allocation to obtain the requested and negotiated QoS

SLA represent the static agreement (how to describe it?)

QoS MANAGEMENT: DYNAMIC PHASE

Dynamic actions **During distribution**

monitoring of properties and eventual changes to respect the **defined policy**

Continuous measurements of QoS level and SLA parameters

respect control and **synchronization**

Verify of fulfillment and potential need of synchronization of different resources (video / audio)

renegotiation of **necessary** resources

New contract to respect QoS and grant SLA

change of resources to maintain **QoS** and
adjustment to new situations

After renegotiation, the new SLA fulfillment must be ascertained and regularly checked

MANAGEMENT and MONITORING

We have a hard problem of the cost of the tools for guaranteeing QoS

We need dynamic data collection mechanisms and policies that do not require too many resources (also used by application execution)and do not affect too much applications

Any correct management must deal with that requirement to reserve as least resources as possible

Performance area (monitor and data management) must define tools and policies that are least intrusive as possible

Minimum intrusion principle

that is:

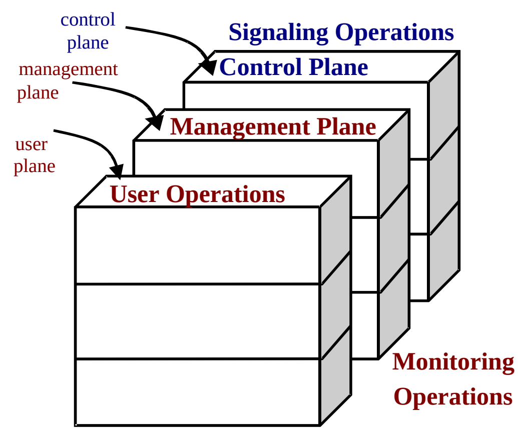
to attempt not to compete too much with applications

MANAGEMENT and MONITORING

Necessity to **match application plan** (or user) with **strategies and tool for** efficiency control

User Plan

for defining the user protocols
(in telephone, the voice)



Management Plan

for service management and monitoring
(in telephone, the QoS handling)

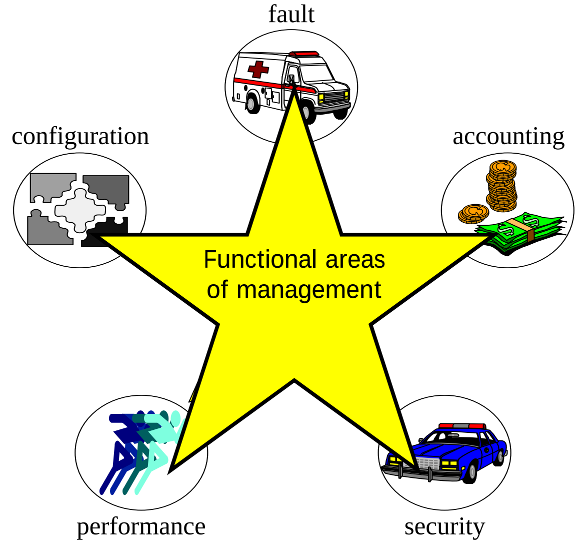
Control Plan / Signaling

to establish the connection, to negotiate and signal between levels, not necessarily in band (in telco, this level establishes the call and works before it)

MANAGEMENT and MONITORING

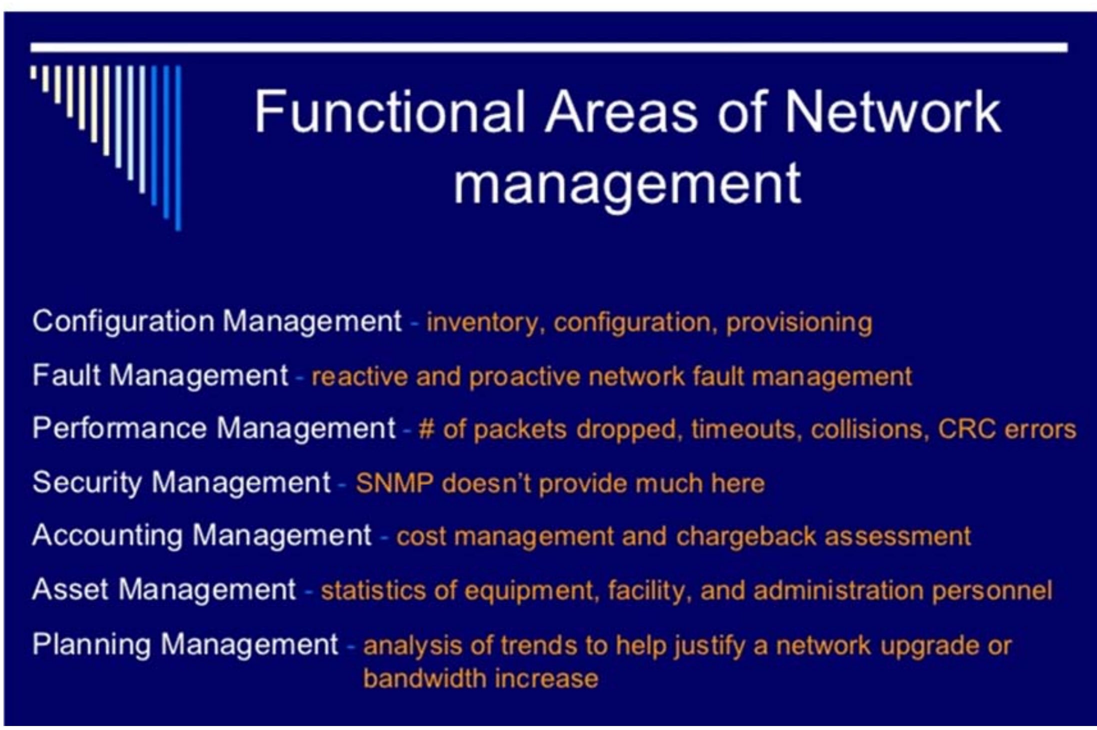
Management functional areas for different management Standards

- **Fault** Management
- **Configuration** Management
- **Accounting** Management
- **Performance** Management
- **Security** Management



See OSI of ISO
SNMP of IETF
TINA of CCITT

NETWORK MANAGEMENT AREAS



SYSTEMS MANAGEMENT - OSI

OSI Management Standard (long life standard)

Model of standard network management with very flexible and dynamic operations and based **on abstract objects**

The mapping of abstract to real objects is not standardized

for example, user interfaces are not standard but there are some standard de facto

OSI Distributed Management

Use of standard description of **objects and actions**

Common Management Information Base (**CMIB**)

Management Information Service (**MIS**)

Unique management of information

Common Management Information Service Element (**CMISE**)

OSI is more sophisticated than TCP/IP management

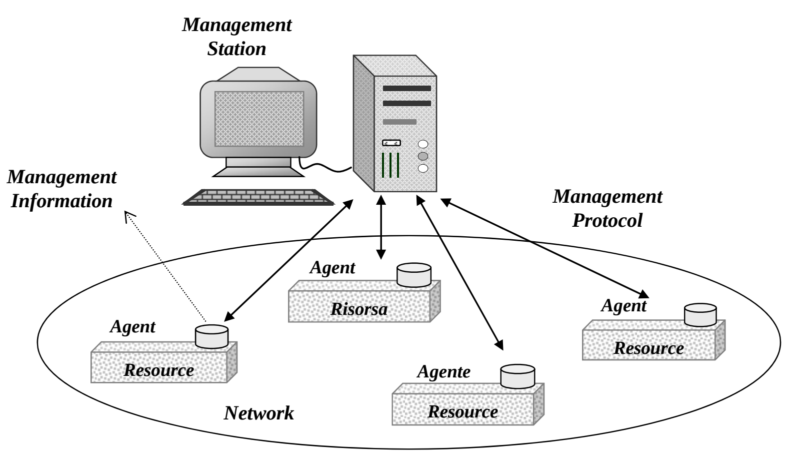
It is an example on how to manage any distributed system to obtain resource distributed management

NETWORK MANAGEMENT

Management Standard based on two roles

- **manager** and
- **agents** that are responsible of managed **resources**

The model does not impose constraints and can lead to very simple implementations



SYSTEMS MANAGEMENT - SNMP

Management Standard IETF

definition of a **simple management protocol**

SNMP Simple **N**etwork **M**anagement **P**rotocol

By using **TCP/IP** and used in **UNIX** and **LAN** environments

SNMP operates on CMIP subset

incompatible with CMIP standard

with variables that agents check by reading and writing them

SNMP has passed many redefinitions and redesign phases

to keep count of **security** need

to keep count of **flexible management** model

to keep count **existing legacy** systems

…

and to manage not only devices, but also **entities of any type**

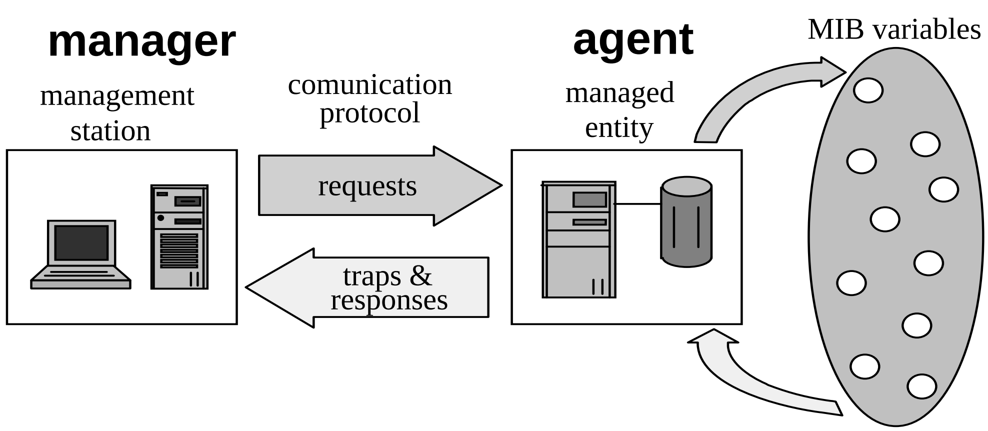
Simple Network Management Protocol

SNMP Simple **N**etwork **M**anagement **P**rotocol

SNMP uses one **manager** (*only one*) and some **agents** *(predefined)* that control **variables** representing the objects, identified by unique names (OID in hierarchical directories)

Manager requests actions (*get* and *set*) and receives response

Agents wait requests and can also send *trap*

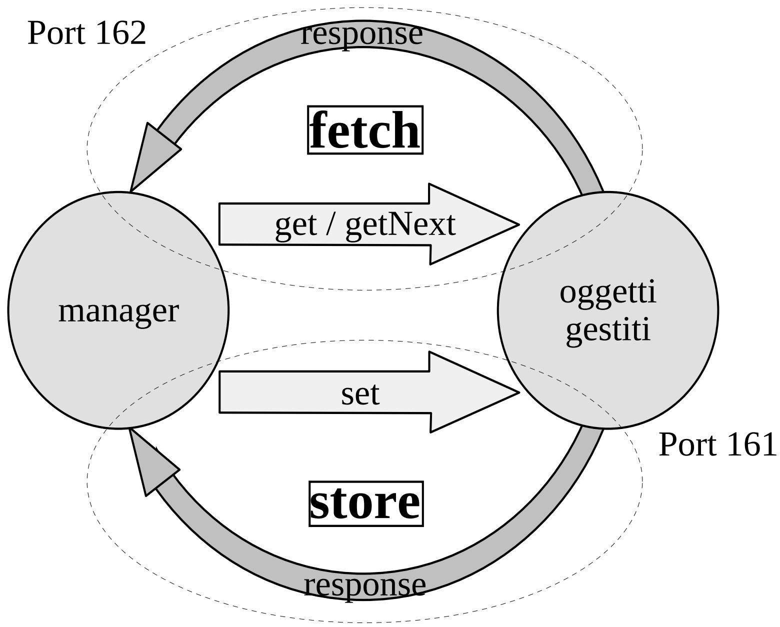


Simple Network Management Protocol

Very simple and limited messages are used
Set,
Get,
Get_Next
(multiple attributes),
Trap,
Simple indications

UDP usage
Port 161 messages
port 162 inside manager for traps

Which resources can be controlled and which measurements can be taken?
Only a few

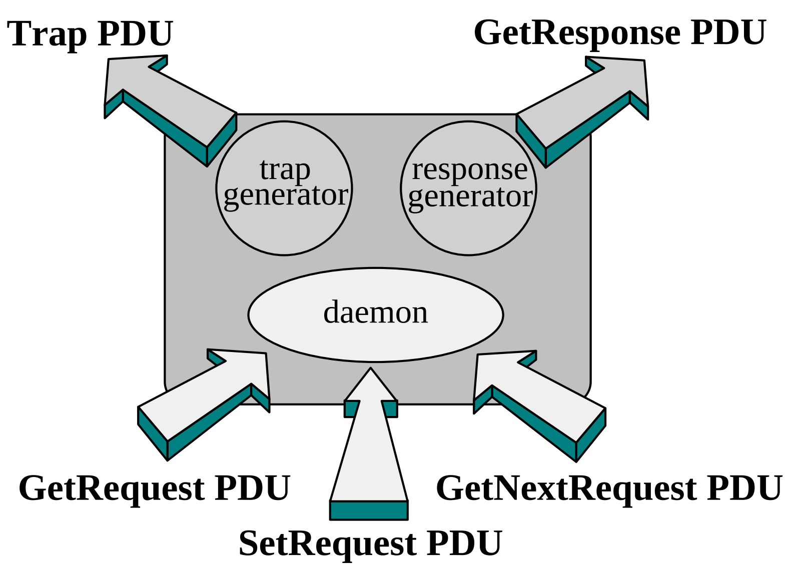


SNMP - Agent

SNMP agent
structure

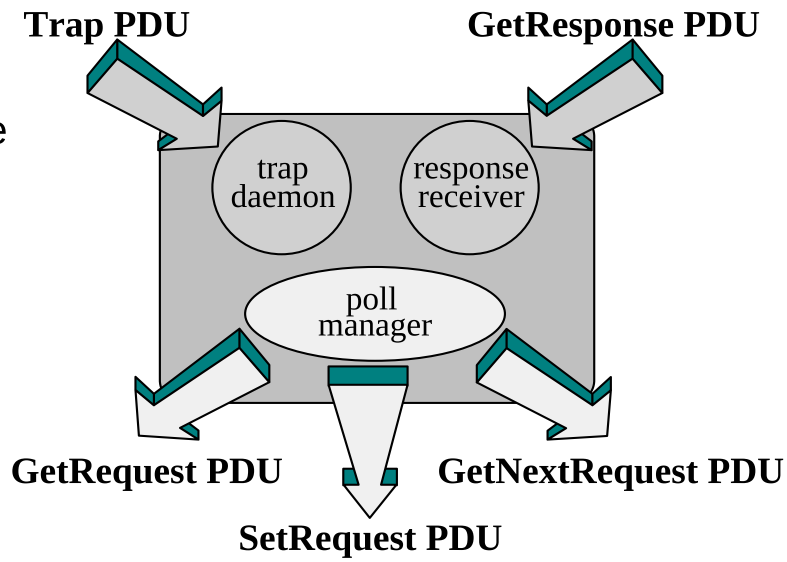
The Agent must handle requests of get and set actions arriving from the manager

The Agent can generate traps when some events occur



SNMP - Manager

SNMP manager
structure



Once requested, the Manager must handle the responses arriving from agents after get and set actions

the Manager must handle Traps arriving from agents

SNMP - STANDARD

SNMP must provide **manager- agent communication** in **a standard way, via standard packets**

Use of **data description** ruled by

SMI (Structure of Management Information)

MIB (Management of Information Base)

Both standardized in a precise way

- **SMI** define rules for objects names (ASN.1 e BER) and **MIB** objects, types, and relationship collections (according OSI X.500)
- The SMI can specify that the exchange involve *3* integers each of *32 bit* and the MIB that we are referring to an object that is in a precise directory (1.3.6.1.2.1.7.1, the IP UDP datagrams inside the basic directory of IETF)

SNMP PROBLEMS

SNMPv1

Extremely simple with Limited expressivity

Addresses only the area of **configuration** management (**fault**)

Limited provision of traps (actions started by the object)

SNMPv2

Overcoming the contraints of C/S manager agent hierarchy

SNMPv3

Introduction of security **S-SNMP**

Integrity information problems is dealt with (also stream), masquerading, privacy (prevent disclosure)

denial of service and traffic analysis are not dealt with

In general, SNMP embeds CMIP and CMISE properties

with a very predetermined vision, before execution and very little capacity of dynamically varying during run time

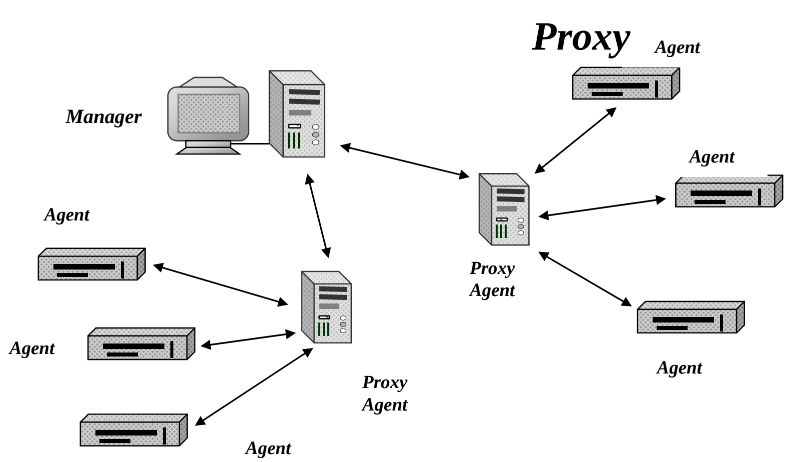
CONGESTION PROBLEMS in SNMP

SNMPv2

Concept of **proxy agent** that is **agent and manager**

Entity that acts as agents and also as managers

overcoming the problem called **micro management** (*i.e., the congestion around manager*)



The manager orders a read operation and the proxies actuate it anyone in their locality For example, the proxy can collect results, and send results in an aggregated and shortened form

NETWORK MANAGEMENT & RMON

SNMP can deal only with variables locals to agents

If you need to manage the network (traffic)?

Remote **MON**itor

RMON controls the support parts of the communication and allows to access to related statistics

RMON may increase user visibility on traffic

how to monitor the network?

Introduction of **monitor agents** and of the interaction protocol between **manager** and **monitors**

RMON1 developed to have multiple and grafted operations

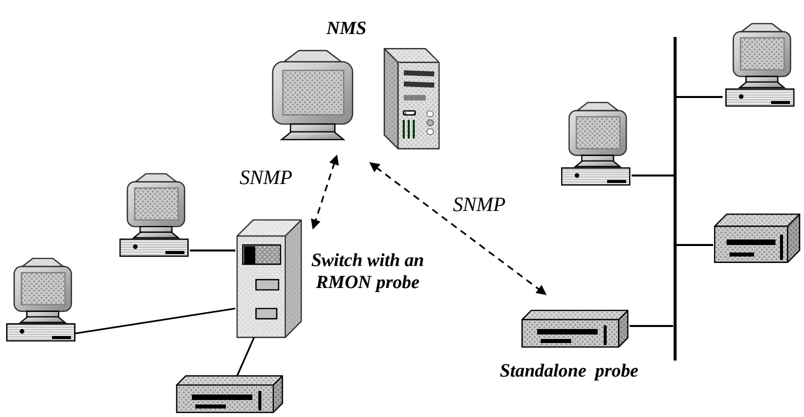
RMON2 and to guarantee security

RMON e PROBE

The **RMON** approach is **oriented toward traffic and bandwidth** and not toward devices

probe an entity capable of monitoring packets on the network

The probes *can work autonomously* and also disconnected from the manager to track subsystems and report filtered information to the manager

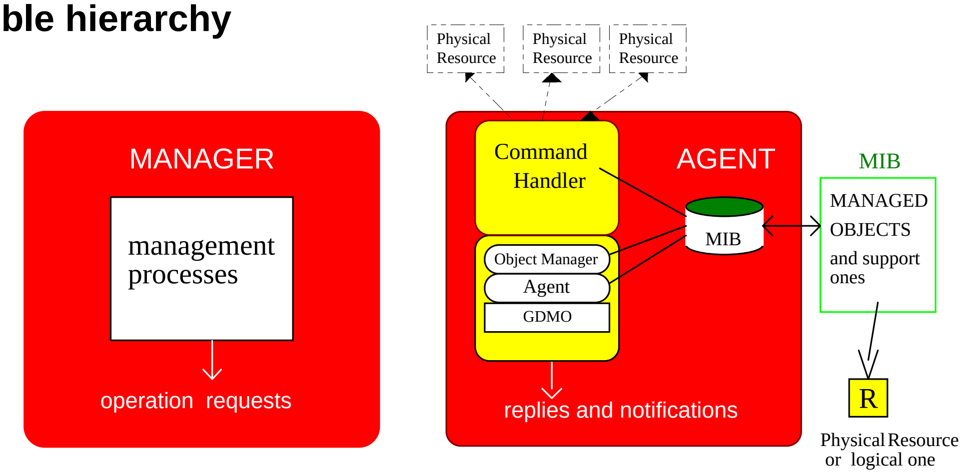


OSI NETWORK MANAGEMENT

Enhanced model of **Distribute Management** based on

active entities	**(manager)**
managed entities	**(objects)**
intermediate entities	**(agents)**

with objects that can be **managers in their turn, to organize a flexible hierarchy**



ADVANCED NETWORK MANAGEMENT

Managed Object are the resources, described as **objects**

An object can abstract away and represent one or more resources of the system, by defining and allowing complex operations

simple resources *a modem,*

or complex ones *more interconnected systems*

*… and can be **created dynamically***

The **Managers** realize **management policies based on managing different agents of other managers**

A manager can both insert a resource and remove dynamically from the management system

The **Agents** act on manager request to **provide functions to execute on request**

services of command execution, information gathering

but also resource insert, new agent creation, new manager, …

ADVANCED NETWORK MANAGEMENT

Management entities use the **CMISE/P** protocol

Set of remote operations for communication between manager and agents to realize a dynamic model at its maximal degree

Set-Modify	to establish, add or remove an attribute to an object
Get / Cancel Get	to read of an attribute of an object (and cancel read)
Action	action on one or more objects
Create/ Delete	to request of creation/destruction to an agent
Event Report	to send of an event notified by agent to the manager

***Note the dynamic addition** of attributes, actions, agents, and events to change the structure of the system during execution (also deletion)*

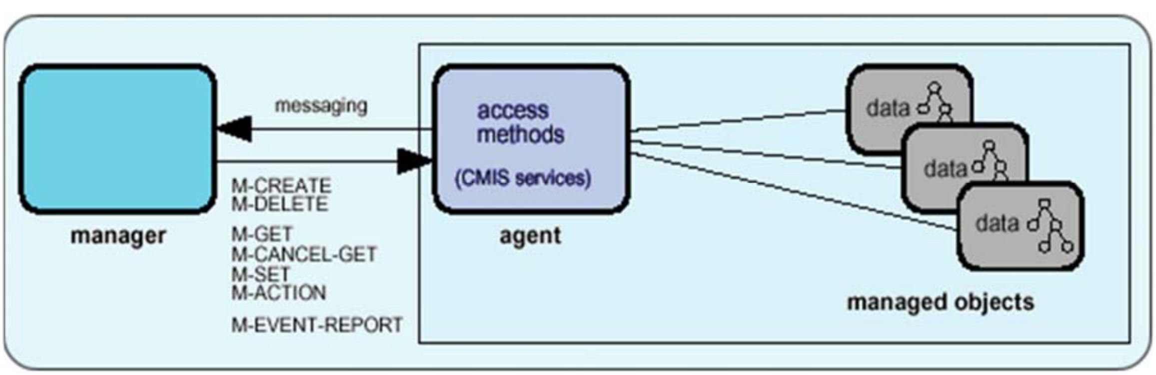
ADVANCED NETWORK MANAGEMENT

Management **operations** in **OSI**

to allow **a dynamic control by defining**

operations to create agents and new actions

and to delete them while operating



MANAGEMENT and MONITORING

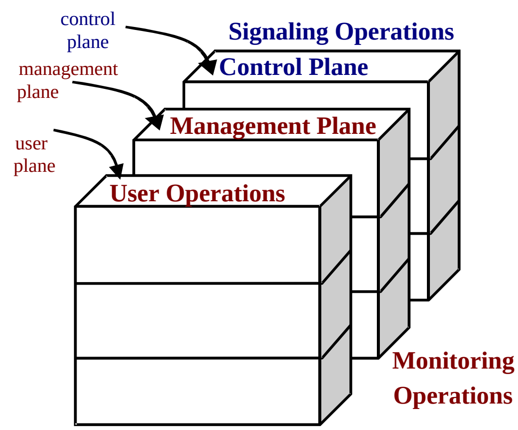
Different plans, from user plan to the plans for operations control and support

User Plan

for user protocols

Management Plan

for service management and monitoring



Control Plan / Signaling

to establish the connection, to negotiate and signal between levels, not necessarily in band (in telco, this level establishes the call and works before it)

SESSION PROTOCOL SIP

Necessity of protocols **able to support and manage multimedia sessions,** similarly to the management of telephonic communication on control plan

The protocol is **Session Initiation Protocol o SIP**

SIP (RFC 2543) is not an old protocol (1999) and often updated, increased, and extended (e.g., with events - RFC 3261- 2002)

The objective is to **define and manage a session to support a multimedia service** that is provided by other protocols

- SIP has the ability of **signaling** for establishing, modifying or closing a multimedia session
- SIP is based on the communication of **HTTP compatible** content
- SIP is a protocol **text-based** and purely **client/server**

MESSAGES in SIP

SIP exchanges some fundamental messages, the only standardized ones, in HTML format

Few message types:

REQUEST messages

INVITE / ACK / CANCEL / BYE

Other Messages

REGISTER (contact information)

OPTIONS Request to servers information on capacity

Messages di RESPONSE

PROVISIONAL / FINAL

1xx provisional,

2xx success,

6xx failure

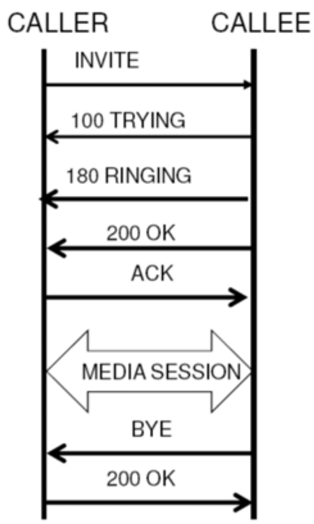
ALL SIP MESSAGES

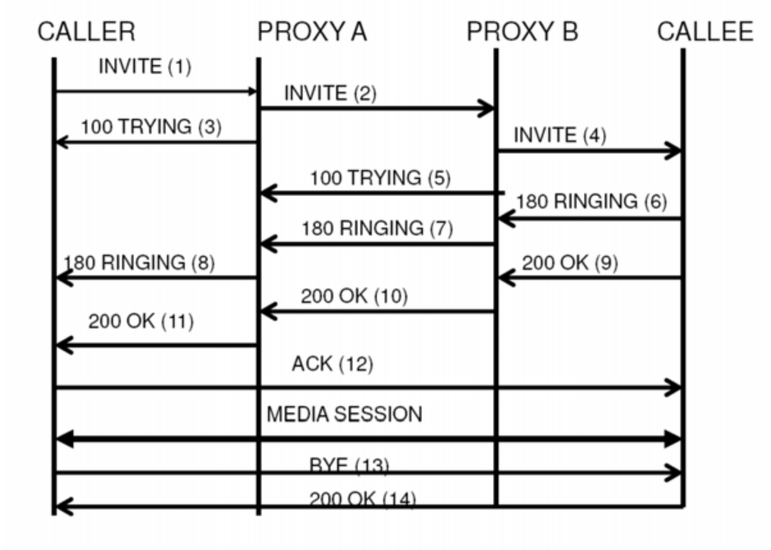
IETF	Nome metodo	Utilizzo
RFC 2976 (2000)	INFO	Per mandare dentro la sessione informazioni che non ne modificano lo stato
RFC 3261 (2002)	INVITE	Per instaurare una sessione, il client invita a partecipare il server
RFC 3261 (2002)	ACK	Il client conferma al server lo stabilimento di una sessione
RFC 3261 (2002)	BYE	La manda uno dei due partecipanti all'altro per terminare una sessione
RFC 3261 (2002)	CANCEL	La manda il client al server per cancellare la sua richiesta pendente
RFC 3261 (2002)	REGISTER	Mappa l'URI del client con la sua locazione corrente
RFC 3261 (2002)	OPTIONS	Richiesta fatta a un server per scoprire le sue capacità
RFC 3428 (2002)	MESSAGE	per trasmettere un messaggio istantaneo
RFC 3262 (2002)	PRACK	per confermare la ricezione di una risposta provvisoria
RFC 3311 (2002)	UPDATE	per modificare alcune caratteristiche della sessione
RFC 3265 (2002)	SUBSCRIBE	un UA si registra a un particolare evento di interesse
RFC 3265 (2002)	NOTIFY	Si notifica un UA di un nuovo evento a cui è interessato
RFC 3515 (2003)	REFER	per istruire un server sul trasferimento di una richiesta
RFC 3903 (2004)	PUBLISH	un UA pubblica un evento

SIP SCENARIOS

A base case
simple and direct

A more complex case with different mediators between client and server





SIP SCENARIOS

It is possible to provide different functional entities, from client and multimedia server, to other involved entities

User Agent

Endpoints that can act as user agent of clients (REQUEST) or servers(RESPONSE) to actuate the protocol

Proxy Server

Routers of application level that can keep the state of session transactions (otherwise stateless), that is the state of the requests sent from a client and response sent back to the client

Redirect Server

Servers capable of sending a client to a new alternative server

Registrar Service

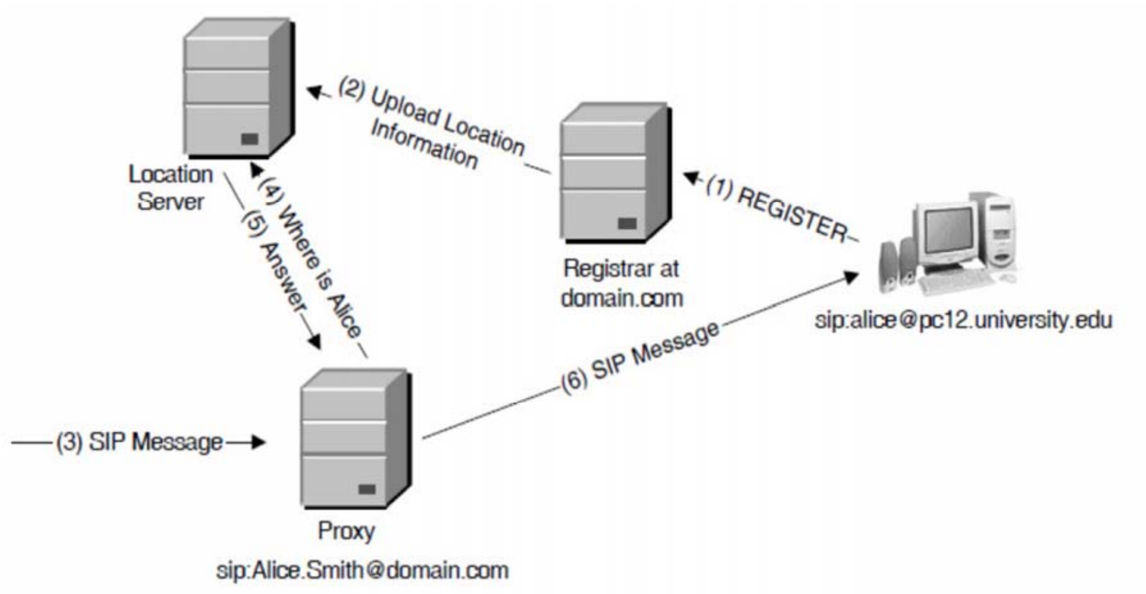
Service for user registration to the infrastructure

Location Service

Service to allow link of interested users to their location

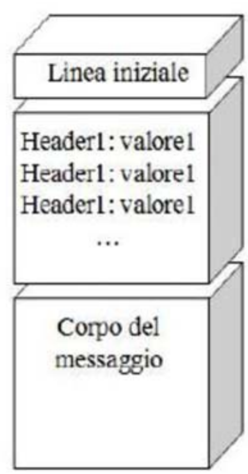
SCENARIO of SIP usage

A more articulated scenario
Proxy, **Location**, **Redirect** and **Registrar Service**



SIP MESSAGE STRUCTURE

The messages are structured as follows:
start-line, header, message body (optional)



For REQUEST messages
The request-line as start-line then
- name-method,
- protocol version,
- request URI

For REPLY messages
The status-line as start-line then
- protocol version,
- state code,
- explicative phase

The body can be present to contain further information on flow and service

SIP MESSAGES: INVITE

An INVITE example

INVITE sip:bob@biloxi.com SIP/2.0 (REQUEST LINE)
Via: SIP/2.0/UDP
pc33.atlanta.com; branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>; tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
…
Message body (optional): an SDP (Session Description Protocol) description to negotiate audio/video formats

SIP MESSAGE

A RESPONSE example to request OPTIONS

SIP/2.0 200 OK (STATUS LINE)
Via: SIP/2.0/UDP
pc33.atlanta.com; branch=z9hG4bKhjhs8ass877; received=192.0.2.4
To: <sip:carol@chicago.com>; tag=93810874
From: Alice <sip:alice@atlanta.com>; tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:carol@chicago.com>
Contact: <mailto:carol@chicago.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
… Message body (optional) …

QoS EXTENSIONS

Traffic Management

To provide a good service, a **traffic management** is **necessary**

In general, it is typically **provided** by **intermediate router nodes** that must handle the traffic itself (beyond **best-effort**)

Routers must manage queues and traffic

Scheduling and queue management

the routers must send packets while considering different flows, to maintain the QoS at any moment thoroughly the whole service

Routers must keep the **state** to differentiate flows

Queue managements activities are necessary

ROUTER INTERNET

The Routers move packets without differentiating ***queuing*** *or* ***scheduling*** *and without distinguishing between flows*

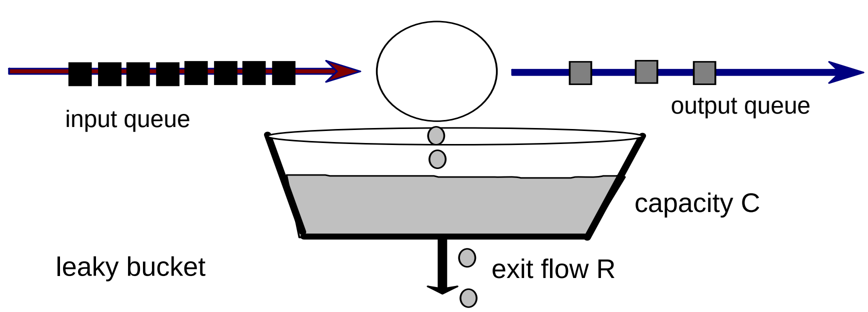
A router executes for every packet that is put into a **FIFO queue**:

1) Verification of the **destination**

2) Access to the **routing tables** to find output path

3) Select the **best output path** for the packet taking into account the **most suitable match** (maximum match length)

4) Forward the packet to the interface selected from the path

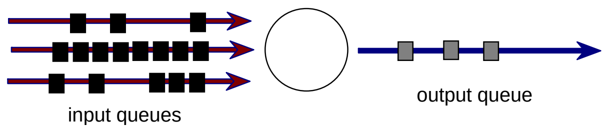


ROUTER INTERNET

Router in Internet best-effort

the router forwards datagrams without considering length or destination/source attributes

The normal work policy is **FIFO**, a unique queue for every flow of the router: that excludes any service **differentiation**



Simple policy and unified queues (a unique one)

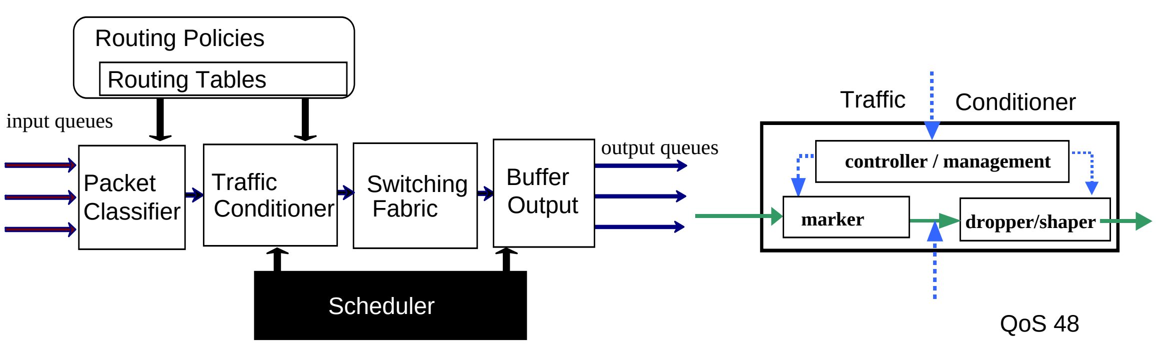
A packet (of any length) when in output can engage the router and block any other flow (delay)

We cannot reserve resources for flows that can have differentiated needs to meet their SLA and related QoS

QoS ROUTER

*The QoS Routers consider policies for **queuing** and **scheduling**, based on **packet,** either their **length** or **destination/source** (to differentiate flows)*

A router can also consider, apart from a packet **classifier** based on flow (source/destination and length), also a function for **traffic conditioning that can also decide to throw away or delay packets for some packets of some flows**

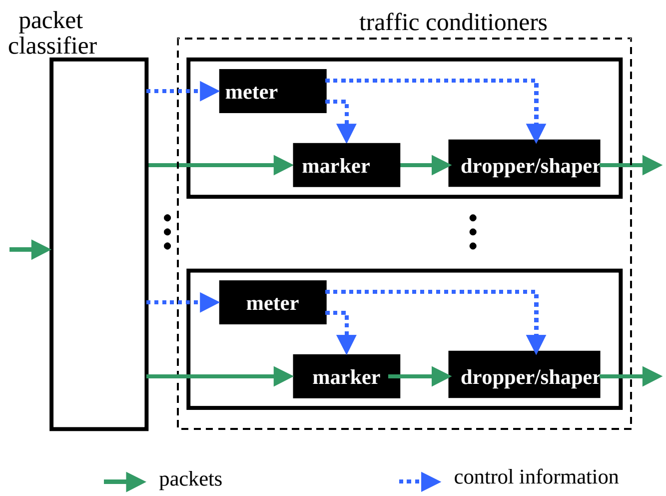


QoS EXTENSIONS

The main problem is how to intervene on **routing** to obtain **respect** (RFC1889) of some SLA of **byte stream flows**

Router organized on effective locality

Locality composed by internal nodes and border nodes



ROUTER SERVICE POLICIES

Router policies*: routers must pass messages*

The **first policy** that comes to mind is, when a packet arrives, to send it out without delay (**no management policy** or FIFO queuing)

That policy **do not insert delay** apart from the one imposed by other packets in output and it is defined **work conservative**

Router best-effort are work conservative

Law of conservation of Kleinrock

A router (router work-conservative) cannot be idle if there are packets on the output port (**delays cannot be introduced on the traffic in any way**)

A router can decide to work according to a policy ***work conservative*** or ***not respecting it, for QoS sake***

When we consider QoS, routers can also **introduce delays** not to penalize some flows: a packet with less priority can be delayed even if there are no other more priority packets (*but they can arrive at any time*)

KLEINROCK LAW

Conservation Law of Kleinrock (for work-conservative router): the router cannot be idle if there are packets to send in output

*If there are **n flows** with λ_n traffic for every flow, and if a flow n has a service a mean time μ_n, then the use is $\rho_n = \lambda_n \mu_n$ where*

ρ_n represents the mean use for that flow, while

q_n represents the mean waiting time for the flow n

The Kleinrock Law for **work-conservative scheduler** requires that

$$\Sigma \rho_n q_n = \text{Constant}$$

that is

it is possible to give a **lower delay** or a **higher bandwidth** to a flow, **only if we can increase the delay of another one** or if we can **reduce the bandwidth of another one**

Even respecting the law, in high load situations a router with limited and defined performances cannot favor a flow without damaging another one by limiting its support to it

Let us recall that Router for **QoS are also not conservative**

MODELS for QoS ROUTERS

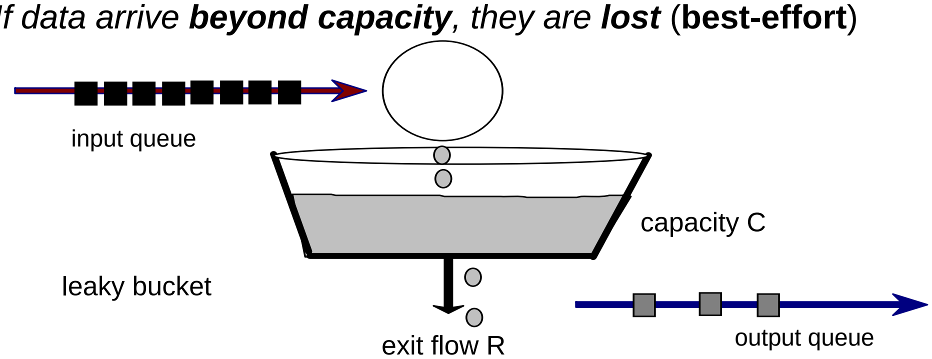
Router with traffic characterization

the router must know the flows and possible service capacity *(router capacity) and must **have access** to traffic **management***

The LEAKY BUCKET models a **router ACTIVELY shaping services** with the strategy of **limiting output flows (a bucket per flow/ all flows)**

We can control flows through **capacity**:

> *If data arrive **too quickly** beyond admissible output flow, they are **delayed*** (**best-effort**)
>
> *If data arrive **beyond capacity**, they are **lost*** (**best-effort**)



LEAKY BUCKET

LEAKY BUCKET for traffic characterization

r maximum output flow, **R** mean input flow

LEAKY BUCKET switch off packet bursts

A packet is queued only if there is place in the bucket (it is *discarded otherwise*) depending on bucket capacity **C**

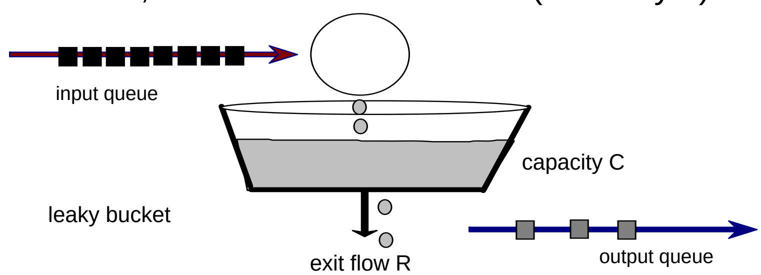
The packets can exit with a **maximum speed r** that limits the allowed input flow **R** (**r < R**)

If 100Mbyte in 300msec and if the output flow is 33Mb/sec, *the leaky regulate the flow to an admissible one*

If 150Mbyte in 300msec, we can start to lose data (≅ 50Mbyte)

c = 100Mbyte

r = 33Mb/sec



TOKEN BUCKET (flow history)

TOKEN BUCKET is a **traffic modeling keeping into account flows history via tokens:** the bucket collects tokens used as authorization to allow packet passing

The tokens are generated uniformly by time for any flow

TOKEN BUCKET allows packet bursts

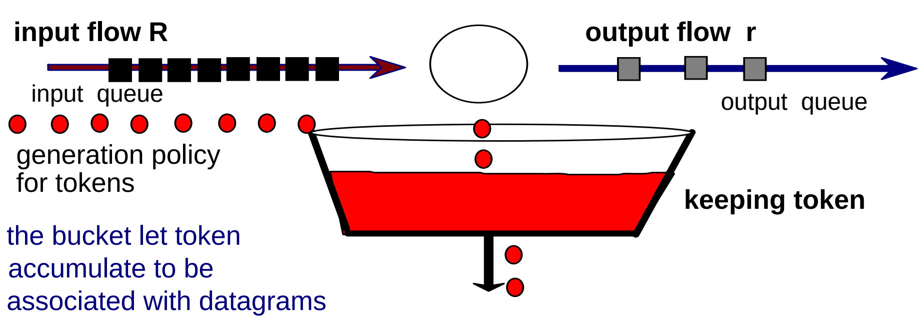
Data beyond capacity are not lost but only delayed

If data arrive too quickly beyond the admissible output flow, they can exit when enough corresponding tokens are accumulated

If the bucket is **empty** → not pass and wait

If the bucket is **full** → tokens can be associated with packets to pass

If **partially full** → something can pass, others have to wait

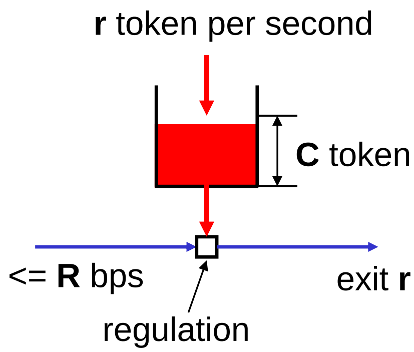


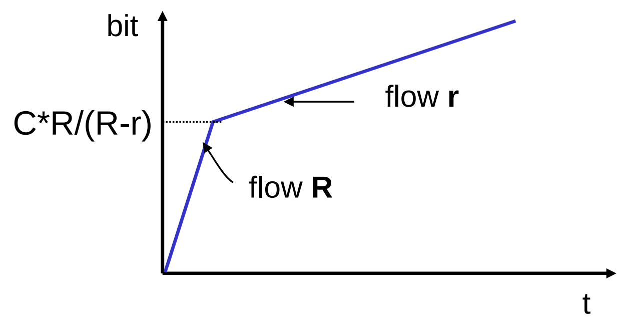
TOKEN BUCKET for QoS

The TOKEN BUCKET models router service with variations and different policies

It makes a flow wait for an appropriate **number of tokens** and **sends** the packet **keeping into account packet dimensions**

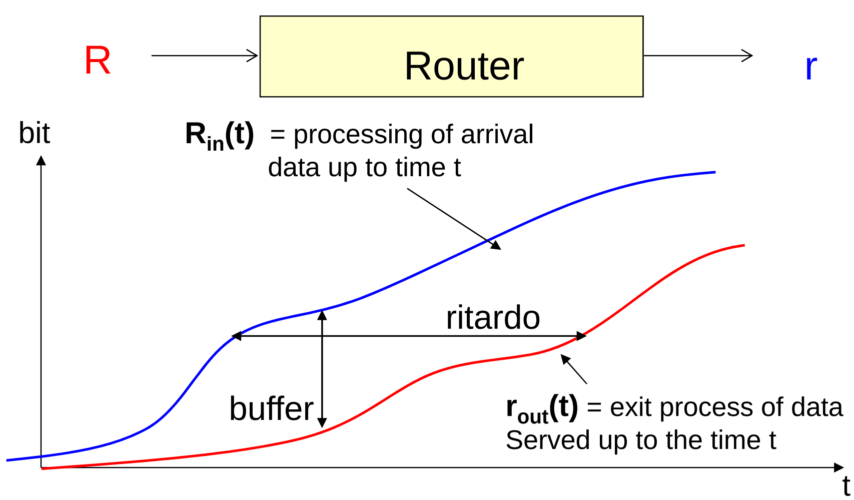
The packet is not discarded if it is not possible to pass it for token loss, but only delayed waiting for token production





SERVICE - QoS

thee TOKEN BUCKET imposes constraints on flows, in the sense of delay, considering **flows** and **requested buffered resources**, before make the packets get out of the router



Often the two kind of bucket policies are used in series

POLICIES for QoS ROUTERS

The Internet Routers - First Come First Serve or FIFO – work respecting Kleinrock conservation law

instead, to give priority to a flow, you have to penalize others flows, by giving less resources to them, with many different policies

Scheduling and **Queuing** must respect some properties

Implementation facility

to make easier the router design and toward real feasibility

Fairness and **Protection**

during the same operation situation any flow must be penalized the same as all others

Performance limits

to define constraints on correct flows operation

Admission Control

decision on admission before the distribution

GENERAL FAIR POLICIES: MAX-MIN

PRINCIPLE - Max-Min Fairness

General strategy to meet the **fairness property**, often implemented with a **policy easy to implement**

Max-Min share → requests of different resources by different flows must be considered in order of growing request (**first the ones that require less and then the ones with higher requests, in order**)

C global max capacity of resources
X_n resources request by flow-n $X_1 < X_2 < X_3 < \dots X_i < \dots X_{N-1} < X_N$
m_n previously allocated resources with success to flow-n
M_n available resources for flow-n

$m_n = \min(X_n, M_n)$ and $M_n = \dfrac{C - \sum_{i=1}^{n-1} m_i}{N - n + 1}$

It is also possible to consider different weights for different flows

GENERALIZED PROCESSOR SCHEDULING

The **Max-Min model** deals and let pass first the flow that has less demanding requests, then the others ordered by weight requests…

Of course, scaling down only in lack of resources

Generalized Processor Scheduling (GPS)
Fluid traffic model

This policy answers to service requests only one at time and in a very fair Round Robin order

At every round it is served only a bit for flow that is forwarded to the output queue

It is possible to prove that this policy is optimum for service scheduling

Unfortunately → **GPS** it is **NOT** implementable in reality

It is possible to serve **only packets** and **not bits** (overhead)

It is necessary to design approximations to it and **easy to be implemented**

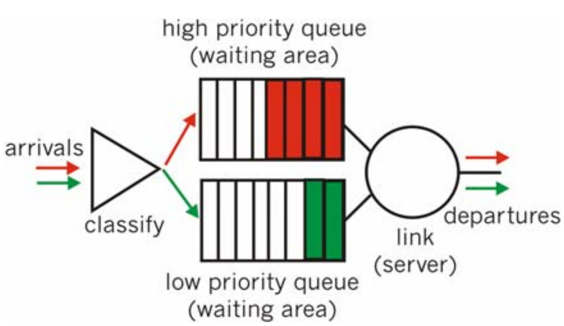
REAL SCHEDULING POLICIES

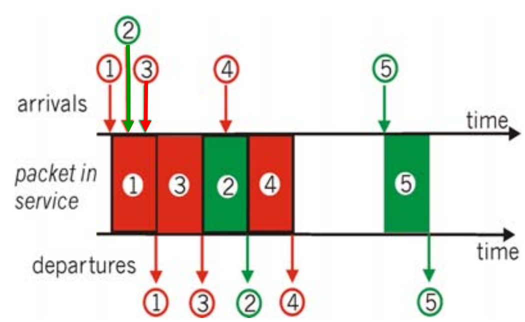
Strategies alternative to FIFO or FCFS

Queue Scheduling forms (*typically **not work-conservative***) to avoid that an **uncontrolled excessive flow** can **congest** the entire traffic situation and **all the flows**

Scheduling with Priority

Queuing policy and scheduling easy to implement… but
A flow with high priority can cause *starvation* of low priority flow

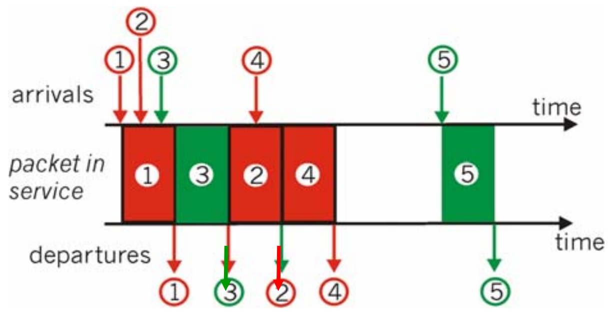




ROUND ROBIN POLICIES

Round Robin

flows served by round-robin scheduling policy (if any traffic)
It can serves repeatedly the traffic of a flow, if it is the only one



Weighted Round Robin

Flows are served by using round-robin but in *proportion to a weight assigned to every flow*

every queue is visited for every round a number of times equal to the weight and for a number of packet that depends to the weight

The normalized weight is more difficult in case of short flows

OTHER ROUND ROBIN VARIATIONS

Deficit Round Robin

Every flow has a **state value** (**deficit** at zero)
When arriving the top of the queue, the packet is **extracted if it is less than *a threshold length***, otherwise it wait for a number of rounds associated with its length and the deficit threshold (augmenting the deficit of a specific amount for every visit)

The packets beyond threshold pass after an appropriate number of wait turns, proportional to their length

It works well when there are a limited number of flows and with small packets on average

There are many different Round Robin variations with different performances and various algorithm with various costs

But *most have visits and extractions in order…*

FAIR SCHEDULING: FAIR QUEUING

Fair Queuing and its variations

GPS principle, as it were per-bit

Messages are not sent 1 bit at time, but using tag for the message end in every queue to select comparatively the packet to output first

(the one that would complete first the service and in the fastest way, if was a per-bit service)

A packet of a size N in a flow can be output only after visiting all other queues N times, by examining all flows 1 bit at time

FAIR QUEUING is the more suitable policy and also simple to implement, and it is typically available on all routers even low-cost

Some of its variations

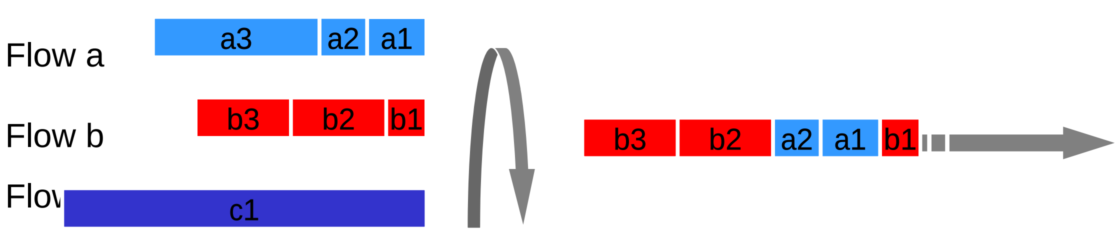
Weighted Fair Queuing with different weight associated to flows

FAIR QUEUING SCHEDULING

FAIR Queuing

Scheduling considers different queue and their messages

The **firsts** to exit are the **packets “that ends first”** and that do obstruct the others less (scenario with flows at the same level), that is we select packets with a size that engages less the router in output



It is possible to weight differently flows (with weights like in **Weighted WFQ**)

SCHEDULING: problem

Fair Queuing is a fair policy that most favors an optimal usage of router resources, by respecting mutual flows constraints

But there is a general problem in QoS routing

NOT KNOWING in **ADVANCE** the **INCOMING TRAFFIC**

the problem is that, when we decide to output a packet we do not know what is arriving on incoming flows that share the same resources

Solution: when possible we insert the state inside the system (forecasting the flow arrival)

In case of **application traffic by flow**, it is possible to estimate, before the real provisioning, the possible average resource load and to consider to forecast resource usage

(through user specification and costs)

Scalability ??? and costs ??? ☹

CONGESTION PREVENTION

One of the more unpleasant situations in ***best-effort systems***

Congestion where no one can work correctly anymore

Often dealt with **simple and reactive policies**

In Internet traditionally best-effort

only **reactive actions** are possible

to discard only excess packets (*silently*) or

to send indication to limit traffic (*choke packets*)

Inside the new QoS Internet with various strategies

it is possible to **undertake also** **preventive actions**

For example the *use of transmission window* on a channel

or other strategies that *prevent dangerous situations*

PROACTIVE POLITICIES: RED

RANDOM EARLY DETECTION (or RED)

a queue for every flow, and *queues with equal priority*

Congestion **prevention** with a ***random discarding*** of packets in every queue, even **much earlier than the congestion situation**

There are many variations: all are based on preventive polcieis packets are randomly discarded, more and more with the growing of the packet queues

RED define the min, max, and mean length of the queue

If queue < **minimum threshold** no action

If queue > **maximum threshold** all new packets discarded

Otherwise discard with probability proportional to queue length

The RED policy has success in preventing congestion

INTERNET SERVICES and NEW REQUIREMENTS

Differentiated service specifications more or less tight

best-effort **appropriate to elastic services like internet services**

No guaranteed throughput, also possible delays, no duplications control or action order guarantees

controlled load **similar to best-effort with low load, but with limitations on delay (with possible overrun)**

elastic services and tolerant real-time

guaranteed load **tight limit to delay and maximum guarantee on flows**

real-time non tolerant services

SERVICES and NEW REQUIREMENTS

IP ⇨ **best-effort**
TCP ⇨ **elastic with ordering warranties, unicity, flow control**
OSI ⇨ **QoS obtained at any level**
Naturally, quality warranties have a cost

Internet is in transition from a low-cost and low-performance infrastructure to infrastructure with differentiated cost and corresponding granted and agreed performances

Integrated Services — **working at single flow level (RFC2210)**

Differentiated Services — **joining and classifying flows for different qualities (RFC 2475)**

http://www.rfc-editor.org/

NEW PROTOCOLS

Protocols evolution ⇨ Integrated Services

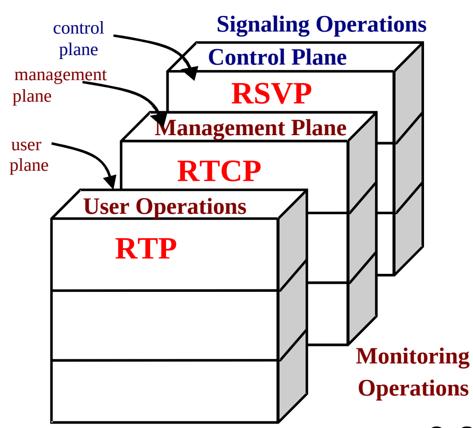
New protocols to adapt Internet to obtain a better control on operations and resources, compatibly with IP best-effort properties

In general, the analysis is done per flow and per hop without considering scalability too much

RSVP ⇨ **Resource Reservation Setup Protocol**
(RFC 2205) **signaling** protocol
To plan resources on intermediate nodes

RTCP ⇨ **Real-Time Control Protocol**
Dynamic management and control
keep negotiated QoS

RTP ⇨ **Real-Time Protocol**
(RFC 1889) general operational messages with reliable sending of frames by using UDP datagrams



Integrated Services for QoS

Integrated Services INTSERV (RFC2210)

QoS support for **flows management** at the application level

The idea behind **integrated services** is to **define** and **keep** a certain **level of service** for every **specific flow** within either an administration domain or also in a global scenario best effort, but with QoS, **working at the application level**

An application requires an agreed **service level (SLA)** specified by using an appropriate **interface** and a **management protocol for any required flow**

The protocols allow to manage and ascertain that the service can be offered (**admission** control) and organize to provide it

The **suite** (it is a set of three protocols) do not implement directly **local actions** and those that grant the **SLA respect** must be obtained at lower level in other appropriate ways (not at the control level)

So, local low levels *(of network) in INTSERV must handle the local actions (reservations, …)*

Integrated Services for QoS

Integrated Services or IntServ – Basic principle

During the service of different flows, the view must be changed

Flows are considered one at a time **(for SLA)**

For every flow, IntServ must consider **not only the endpoints** but also **the whole path** to identify the whole route and to implement the actins for providing resources for the virtual channel

The path becomes an **active path, by working hop-by-hop, and involving several nodes**

In general the service is enabled by

- one active initiator (receiver or client) and

- one service provider (provider)

- Many intermediate nodes that must be connected by the active path, adapted to the service supply via those identified and selected mediators

RSVP - Reservation Protocol

The RSVP Reservation Protocol specifies how to communicate between neighbor nodes to enable the reservation of needed resources to guarantee an agreed SLA (in a complete separate way from the current Internet traffic)

The **R**e**S**er**V**ation **P**rotocol handles the management via the **desired traffic information sent to the receiver** (***in the direction of the flow provisioning***) elaborated by its initiative from all **nodes of the active path** to obtain the service itself and from the **receiver** of a service (***in the direction from the receiver to the sender***) in the second phase

Protocol before service and out of band
(not competing with user data)

Exchanged messages: Path, Resv, ResvTear, PathTear, ResvErr, PathErr,

The negotiation of the FlowSpec (Flow Specifications) via

- **TSpec** (traffic description) sent from the receiver through the network
- **AdSpec** (optional) the sender confirm to the receiver the reservation

RSVP enable resource reservation in an unidirectional way (sender to receiver)

RSVP PRINCIPLES

The RSVP Reservation Protocol is the protocol for the **active path identification** for **one flow to grant** its **QoS and SLA,** but **does not reserve** resources (but it enables its reservations)

RSVP is at the **application** level, **out of band,** and **before service provisioning**
based on **two phase propagations**:
- First, **announce messages** go from providers with offers toward potential receivers
- Second, the receiver **propagate inversely its intention** of creating an active path, typically requiring reservations

The **state allowed is soft**, and it is maintained for a limited time
Many optimizations are possible, such as **shared paths**, **multiple providers**, …

In general, the implementation is free of choosing **wide decisions** (overhead?), since RSVP works before the real provisioning, so it impacts less on service execution

RSVP - Reservation Protocol

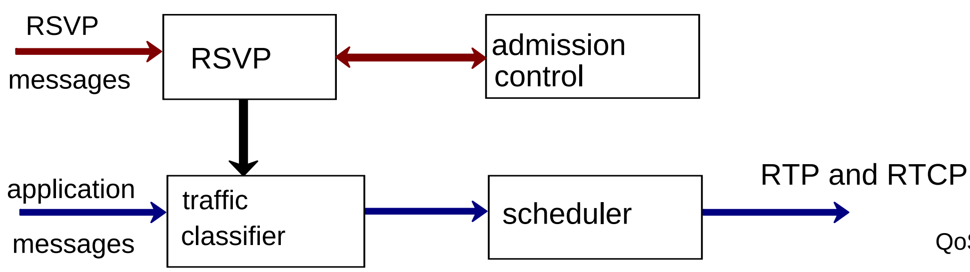
RSVP (RFC 2205) part of INTSERV (soft state and two steps)

RSVP is a **static (out of bands) two phase protocol**, with **soft state,** where the **receiver** requests to enable **resource reservation** for the whole service duration

in an independent way from possible multicast or unicast routing

in a permanent way but for a period (**soft-state** to be refresh)

It is also possible to reserve resources in a **shared way (sharing between different flows)** or **fixed** (no shared with possible optimizations for sharing, such as having several providers)

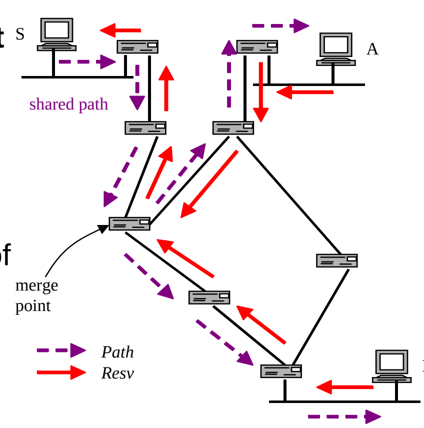


RSVP - Message Protocol

RSVP is a protocol in two phases with messages **Path** and **Resv (the first from sender, the second from provider)**

1) messages **Path** arrive from servers in **broadcast**
sender: message **Path** and
2) receivers send **Resv** to define final paths
receiver: message **Resv** -
TSpec (+ Rspec also in **broadcast**)
3) **refresh the soft-state** by using other message of
Path and **Resv**

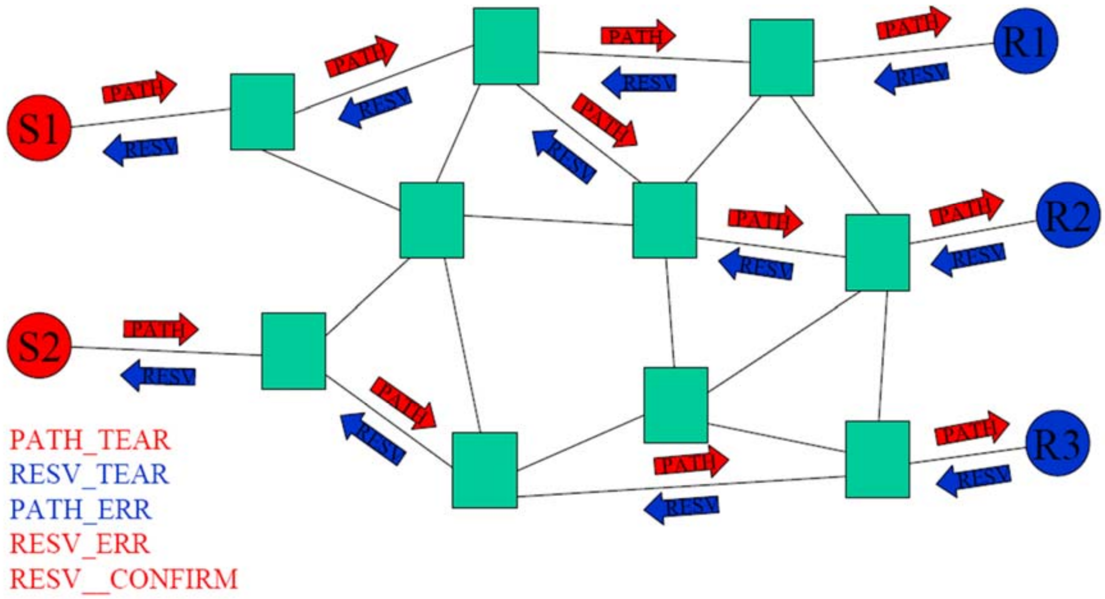
It is possible to answer with *PathTear* or time-out
sender: **PathTear**
receiver: **ResvTear**



- **Broadcast** use when needed but it has **high cost**
- Nodes keep the **soft-state** until next reservation
- Paths and resources are reserved in a **private** or **shared** way

Reservation Protocol Propagation

RSVP: the first PATH phase **propagate from sender** and the second phase **RESV messages** in the opposite way



RSVP - Reservation Protocol

RSVP leaves responsibility of **reserving resources to the application level tools,** before provisioning

*For the **two step protocol,** a reservation can block another one producing an **ResvErr***
*The state must be kept for **every receiver** and **traffic** is produced for **every state refresh***
*It is possible to share **resources in multicast** and compatible service levels for **different receivers** must be provided*

Events to consider and reconsider

In case of **router failure**, the **QoS** can also downgrade to best-effort → in this case it is necessary to **renegotiate QoS during provisioning!!**

Applications and routers must know that RSVP is in use and there can be problems with legacy applications
At the moment it is recommended only for *limited local networks* and *not for global environments*

RSVP SUMMARY

RSVP: single-hop Protocol inside **INTSERV** suite (from one node to one potential neighbor) **on the active path** (to be identified)

- RSVP has the only objective to **signal information** to reserve necessary resources for QoS
- RSVP is oriented to operations **on receiver initiative**
- RSVP produce **state on every node of the path** that is established from the sender to receiver during phase two
- RSVP defines a **non permanent state** of the active path
- RSVP can allow **sharing of active paths in various forms**
- RSVP can work together with **any routing protocols,** either unicast or multicast during routing activity
- RSVP it is not a **routing protocol** but must be compatible with them (IPv4 and IPv6 for example)

OTHER INTSERV PROTOCOLS

Support protocols for QoS at application level (inside band) during activity (RFC1889)

Considering **UDP as the transport protocol** (we exclude TCP(?)), two **protocols for data communication at the level of single flow** are defined and used **(single-hop protocols)**

RTP → **R**eal-time **T**ransport **P**rotocol *port UDP even*

RTCP → **R**eal-time **T**ransport **C**ontrol **P**rotocol *port UDP odd*

that can perform a **QoS control** during **service provisioning** making available information at the application level (*obviously they provide QoS information but cannot grant SLA*)… so that the **application level can work to produce the necessary actions for SLA granting**

For flow transmission, and for the whole duration

RTP define messages for traffic marking, time, and application based

The RTP messages go **in band with progressive numbers** and **associating time together with the flow of data**

RTCP connection management messages

RTP for QoS

Support protocols to QoS at the application level

- The **information flows** are sent from the sender to the receiver through an **application** connection managed hop-by-hop
- Every packet (**flow frame**) are identified with **progressive number** tags and can also be **identified by different routers classifiers**
- It is possible to provide indications on the **transit time** in any path hop between sender and receiver
- In case of missing packets, the suggestion is to **not** retransmit, but to **interpolate** previous packets
- Different applications can take advantage of **differentiated formats** of the **packet data** part to insert specific requests

RTP - Real-time Transport Protocol

Real-time Transport Protocol (from sender to receiver)

Active role for both **sender** and **mixers** that can actively work on the protocol, by inserting passage traces (in the ***sender – receiver direction***)

The mediators can intervene on the message with timestamps, to add information to give information toward SLA monitoring

RTP

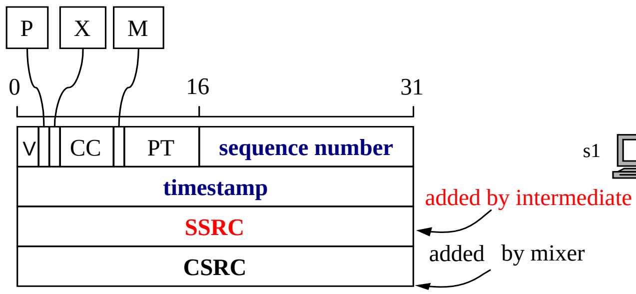
Intermediate nodes (as additional sources) can insert **information on application messages** that can be used to **further qualify** the information delivery and propagate information on **possible delays**

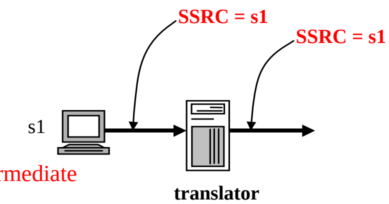
The **active path** become a **set of sources** for every node in the active path

It is possible to consider **shared paths** that therefore can produce more complex graphs with nodes belonging to more paths (and mixers)

RTP - Real-Time Transport Protocol

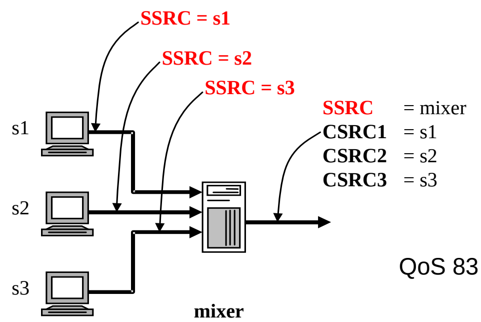
Real-Time Protocol - several source nodes, primaries, within the path (synchronization source) and also for sharing (contributing source)





V	2-bit, version number (=2)
P	1-bit, padding
X	1-bit, signal an extension di header
CC	4-bit, numero di CSRC (CSRC count)
M	1-bit, marker specific per profile
PT	7-bits, payload type, specific of profile
SSRC	synchronisation source
CSRC	contributing source

timestamping in units defined by profile/flow



REAL-TIME TRASPORT CONTROL PROTOCOL

Real-time Transport Control Protocol RTCP (bidirectional)

Provides global and concise information of **flow control** at the application level, with *synthetic information on the service execution*

Control messages are sent together with the traffic (in band, obviously competing with application)

The RTCP messages *travel in both directions* and allow to propagate information to every participant, related to normal operations and to exceptional events

Objective is to 'quickly' propagate the knowledge about the current situation, to give anyone the information to intervene

QoS per flow

information on packets: *loss, delays, jitter*

information of end system: *user*

information on application: *specification on applicative flow*

Real-time Transport Control Protocol

RTCP Protocol (strictly associated with RTP) for flows management with QoS and only to transport control information for the current flow engaged by RTP

While the flow is provisioning, RTCP can provide synthetic information on flows parameters, like delay, bandwidth, jitter, etc.

Objective: **possible correction**

Use of typed messages

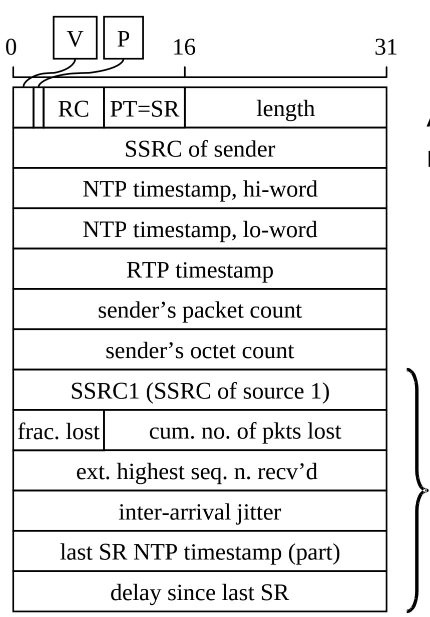
RR / SR	Receiver / Sender Report
SDES	Source Description
BYE	Session Abort
APP	Application specific

The RTCP protocol is bound to execute by using the **same resources** (bandwidth) of **RTP**
RTCP is limited in **intrusion**, limiting its **percentage usage in bandwidth** (5% - 10% of RTP)

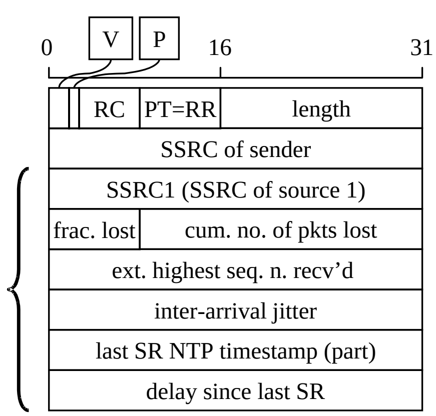
RTCP

Messages of **RR** and **SR** type (**R**eceiver / **S**ender **R**eport)

RI5



Anche più istanze ripetute in un report



RI5 immagine con testo in italiano
Raffaele Ianniello, 4/29/2016

RTPC - Real-time Transport Control Protocol

Messages of SDES type, logic description of a flow

Source **DES**cription as ASCII strings with type information

- CNAME: canonical identifier (mandatory)
- NAME: user name
- EMAIL: user address
- PHONE: user number
- LOC: user location, application specific
- TOOL: name of application/tool
- NOTE: transient messages from user
- PRIV: application-specific/experimental use

RTCP MESSAGE FORMATS

Messages of type BYE

BYE specify the abandoning of an RTP session

An **SSRC** (or SSRC and CSRC list if mixer) send this message, …

providing a suggestion for its abandoning reasons

Free messages of type APP

APPlication allows to pass application-specific packets

An **SSRC** specifies ASCII strings 'for name of element' as data application dependent

In summary… for INTSERV, an application flow,

- **prepares the path and enable resource reservations with RSVP (static phase) before provisioning**
- **during provisioning the flow is associated with RTP (and RTCP)**
- **in case of problems, also a new path negotiation can occur, even locally to mediators (via RSVP)**

RTSP - Real-Time Streaming Protocol

Streaming Protocols: **Real Time Streaming Protocol** (RFC 2326)

Integration of a **Web-based streaming** transported up to a final client (such as in **RealPlayer**)

RTSP starts after downloading the *file specification from the server*

The player communicates with the server via **UDP or TCP,** trying to obtain a better provisioning and adaptation, by exploiting only a **local buffer strategy**

The receiver *does not wait the entire file* (all the frames) to provide the stream, but keeps a *buffer* to always contain some frames

- if UDP, wait 2-5 seconds than start to show
- If TCP, a larger buffer is used

Policies pull and push on the server with **watermark techniques** to better synchronize (if under the threshold, it starts pulling requests)

Also **interleaving technique** are used to deal with packet loss

DIFFSERV (Differentiated Services)

Differentiated Services (DIFFSERV RFC 2474, 2475, …)

DiffServs **differentiate flows in classes** easy to be managed and handled together
They achieve **greater scalability**, by supporting low-level differentiation, i.e., they work at low level in the OSI standards

Differentiated services are very domain specific to user community. Now IETF group are defining different standards for DiffServs

Diffservs require a little user involvement and they are easier to use than IntServs, so they are suitable for legacy applications

The packages are **marked** at the **network layer** (not at the application level): routers recognize and process data in an aggregate and direct form

DiffServs do not work for each information flow separately, but aggregating network level classes of flows

DIFFSERV (Differentiated Services)

Different service class are used: for example

* **Gold** **70% bandwidth**
* **Silver** **20% bandwidth**
* **Bronze** **10% bandwidth**

also

* **premium** (low delay)
* **assured** (high speed, low packet loss)

The classification is done when the packet enters, based on packet content

Service Level Agreement (SLA) based on classification

Policy of service arranged between user and server, and service provided by the network with policies ensured by routers

A flow is classified and then the support is automatic, after inserting it in its proper class

DIFFSERV (Differentiated Services)

Service classes in RFC3246 **expedited forwarding**

Expedited forwarding vs. Regular forwarding

Routers must keep at least **two differentiated queue** and guarantee the delivery of **expedited packets** in every hop (Per-Hop Behavior)

In the case Expedited: PHB **low loss**, **low delay**, **low jitter**

A point-to-point connection is created like shared line between endpoint

Service Level Agreement (SLA) (80 –20)

Packets must receive at least a **Weighted Fair Queuing**

Service classes RFC2579 **assured forwarding**

Four **priority classes** with **three service levels** in case of **congestion** (low, medium, high)

The different packets must be labeled and processed in a differentiated strategy

Differentiated Services Mechanisms

DIFFSERV can use different mechanisms to differentiate services for different protocols

the easier to use and the most spread seems to be the **byte** called **DS (Differentiated Service)** inside the packet header (Type of Service - Service type, or **ToS**, in IPv4)

packet marking inside DS byte

- **IPv4 ToS byte**
- **IPv6 traffic-class byte**

Traffic classifiers based on

- multi-field (MF): **DS** byte + other **fields**
- Aggregations of behavior (BA): only **DS**

DS codepoint depending on the application scenario

Any flow is classified at its input and inserted in the right queue

Per-hop behavior (PHB) granted when joining at the managed network

QoS EXTENSIONS (RFC1889)

Necessity of traffic profile measuring

use of profiles: **in-profile**, **out-of-profile**

to decide how to handle the traffic

also **re-marking** (new DS codepoint, Differentiated Service) to influence / reconditioning the traffic

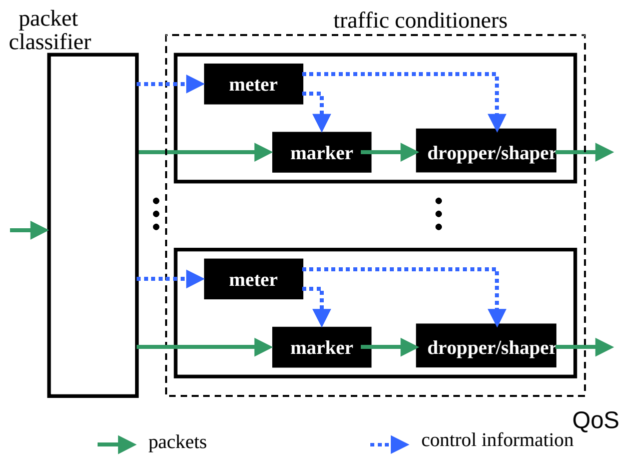
Possibility to have

Shaper

Dropper

on packets

and actively intervene on current traffic



DIFFSERV MULTIFIELD

The traffic classifiers work selecting packets by using **information inside headers**, in the widest possible way (keeping into account any available information)

It is possible to consider

- the **external ports**,
- the **kind of protocol**,
- the **kind of reservation**,
- ...

DIFFSERV present still limits compared with what is possible to achieve with **RSVP** and **integrated services** and are still experimented in small limited areas

Often **joined usage of the two approaches together**

in connected areas that integrate (IntServ and DiffServ)

New DIFFSERV Proposals

INTSERV and DIFFSERV together

Currently, both efforts cooperate to put together both **differentiated** and **integrated** protocols

Even if **differentiated services** seems to be **more scalable** and can **provide performances also within legacy services**, the integrated can grant some specific QoS to some flows

Obviously, routers must provide those new integrated approach