Università di Genova
Facoltà di Ingegneria

11. Servizi Multimediali e Qualità del Servizio (QdS) su IP

Prof. Raffaele Bolla

Reti per servizi multimediali

Caratteristiche fondamentali

- Tipicamente richiedono il rispetto di un qualche vincolo sul ritardo (massimo, *jitter*, medio)
- Ma sono "tolleranti" alla perdita (perdite poco frequenti causano "solo" una leggera perdita di qualità)
- I requisiti, in questo senso, sono esattamente opposti a quelli richiesti dai servizi dati.
- I servizi multimediali sono spesso indicati anche col nome di "***continuous media***"

Classi di applicazioni *continuous media*:

- ***Streaming stored audio and video*** (video diffusivo non in tempo reale)
- ***Streaming live audio and video*** (video diffusivo in tempo reale)
- ***Real-time interactive audio and video*** (video interattivo in tempo reale)

6.2

Reti per servizi multimediali

Streaming stored audio and video

- Un *client* richiede un file audio/video ad un server e concatena/parallelizza la ricezione dalla rete con la visualizzazione.
- E' interattivo: l'utente può eseguire delle operazioni di controllo (operando in modo simile ad un video registratore : *pause*, *resume*, *fast forward*, *rewind*, etc.)
- Ritardo: dalla richiesta del *Client* alla partenza della visualizzazione possono passare da 1 a 10 s.

Streaming live audio and video

- Molto simile alle TV e radio attuali ma realizzato tramite internet.
- Non interattivo, solo "guarda e ascolta".

Interactive Real-Time :

- Conferenze audio/video.
- Vincoli sul ritardo più stringenti a causa dell'interazione in tempo reale.
- Video: accettabile < 150 msec
- Audio: buono < 150 msec, accettabile < 400 msec

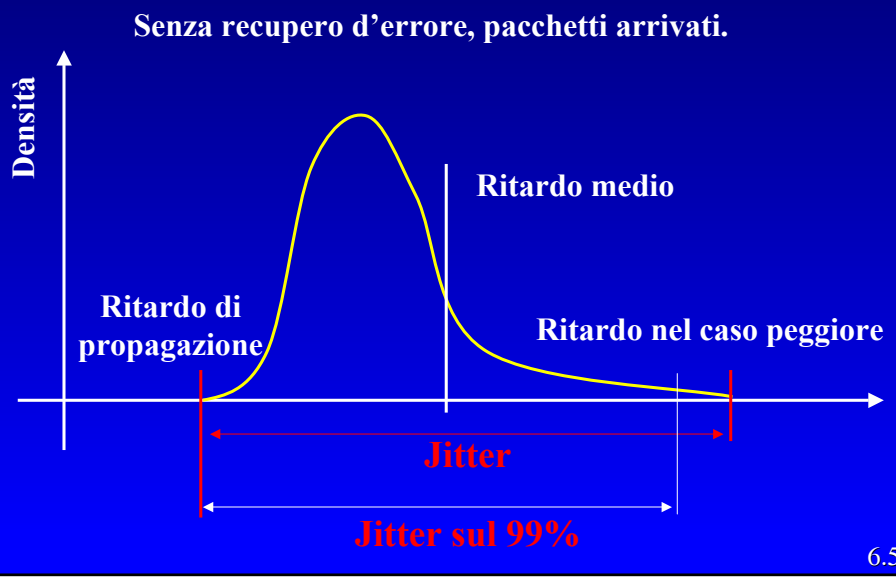
6.3

Il supporto di IP

- TCP/UDP/IP suite fornisce un servizio *best-effort* senza garanzie sul ritardo o sulla variabilità del ritardo.
  - Le applicazioni *streaming* con un ritardo iniziale di 5-10 sec sono oggi relativamente comuni, ma le prestazioni si deteriorano quando le linee tendono a congestionarsi.
  - Le applicazioni in tempo reale interattive hanno dei requisiti più stringenti sia sul ritardo che sul *jitter*, requisiti che non si riescono a rispettare in modo affidabile usando il TCP/IP.

6.4

Ritardo e *Jitter*



6.5

Servizi multimediali su IP

- In ogni modo si possono attuare alcune azioni per mitigare l'effetto del "*best-effort*":
  - Usare UDP evitando la fase di *slow-start* ed il recupero d'errore del TCP.
  - Usare un buffer ed un ritardo iniziale di visualizzazione a destinazione per compensare il *jitter*.
  - Introdurre dei *timestamp* così da permettere la riproduzione temporale corretta dei dati.
  - Adattare la compressione alla banda disponibile.
  - Inviare pacchetti ridondanti per mitigare le perdite.
- La realizzazione di servizi multimediali sarebbe certamente più agevole se la rete IP distinguesse almeno due classi di servizi (ad es. 1° classe e 2° classe).

6.6

Linee evolutive di Internet verso il supporto delle comunicazioni multimediali (QdS)

- Una prima filosofia non prevede particolari interventi architetturali:
  - Nessuna distinzione di classe e nessuna prenotazione di banda.
  - Quando la domanda sale fornire più banda in generale (aumentare le capacità delle linee o il loro numero).
  - Cercare di localizzare i contenuti sul bordo della rete (*Cache* e *server* nella rete d'accesso degli ISP) .
- **Un altro elemento sono le VPN (*Virtual Private Networks*)**

6.7

Linee evolutive di Internet verso il supporto delle comunicazioni multimediali (QdS)

Integrated Services

- Si modifica i protocolli Internet in modo che possano riservare banda *end-to-end*
  - Richiede un protocollo che riservi la banda.
  - Richiede un meccanismo di *scheduling* nei *router*.
  - Le applicazioni devono dichiarare le caratteristiche del proprio traffico.
  - Richiede del software nuovo e complesso sia nei *router* che negli *host*
  - Poco scalabile.

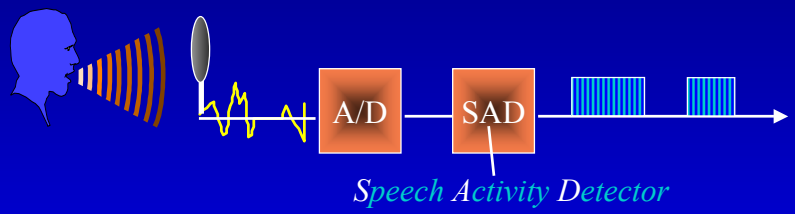
Differentiated Services

- Con "poche" modifiche alla struttura della rete si può fornire due (o tre) classi di servizio:
  - I datagram vengono contrassegnati come appartenenti ad una classe.
  - Gli utenti pagano di più per i pacchetti della 1° classe agli ISP.
  - Gli ISP pagano di più al *backbone* per i pacchetti di 1° classe.

6.8

Multimedia su *best-effort*:
Un esempio di servizio voce su IP

- Una sorgente vocale può essere modellata come un sistema a due stati: *parlo* e *silenzio*.

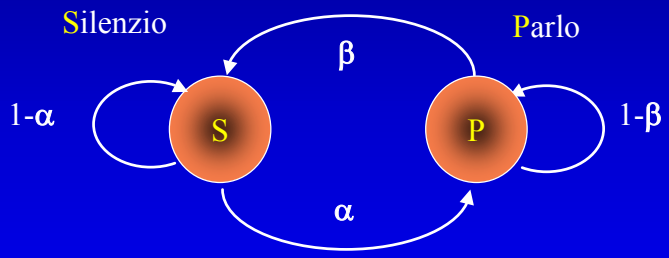


- Durata media del periodo attività (*talkspurt*) = 1.00 s
- Durata media del periodo di silenzio (*gap*) = 1.35 s
- Percentuale di attività ≈ 42 %

6.9

Multimedia su *best-effort*:
Un esempio di servizio voce su IP

- Il modello diventa del tipo:



6.10

Multimedia su *best-effort*: Un esempio di servizio voce su IP

- Nel caso in esempio, supponiamo che quando la sorgente è nello stato ***parlo*** l'applicazione generi dati con tasso pari a 64 Kb/s (8000 campioni/s, ognuno di 8 bit), nello stato *silenzio* ovviamente non genera dati.
- Questa significa che l'applicazione, ogni 20 ms genera un blocco di 160 Bytes.
- Al blocco aggiunge una intestazione ed il tutto viene inserito in un pacchetto UDP ed inviato in rete.
- Il ricevitore riceve questi pacchetti, ma no tutti perché alcuni vanno persi, e con ritardo variabile (eventualmente anche fuori sequenza).
- Il ricevitore deve decidere quando riprodurre un blocco e cosa fare se mancano dei blocchi.

6.11

Multimedia su *best-effort*: un esempio di telefonia su IP

- Sono due i problemi principali che vanno risolti
  - Perdite di pacchetti:
    - » Si potrebbe usare TCP, ma la ritrasmissione introduce ritardo e il controllo di flusso limita il tasso.
    - » Se si usa UDP, in ogni modo, bisogna inserire numeri di sequenza per accorgersi di eventuali perdite.
  - Ritardo
    - » Il ritardo *end to end* deve essere inferiore a 400 ms, altrimenti ne risente l'interatività.
    - » Il *jitter* in questo caso è dato dal fatto che i pacchetti sono trasmessi a 20 ms uno dall'altro ma possono arrivare al ricevitore sia più vicini che più lontani.
    - » Si dovrebbe comunque inserire dei *timestamp* per capire quando un blocco dati debba essere riprodotto.

6.12

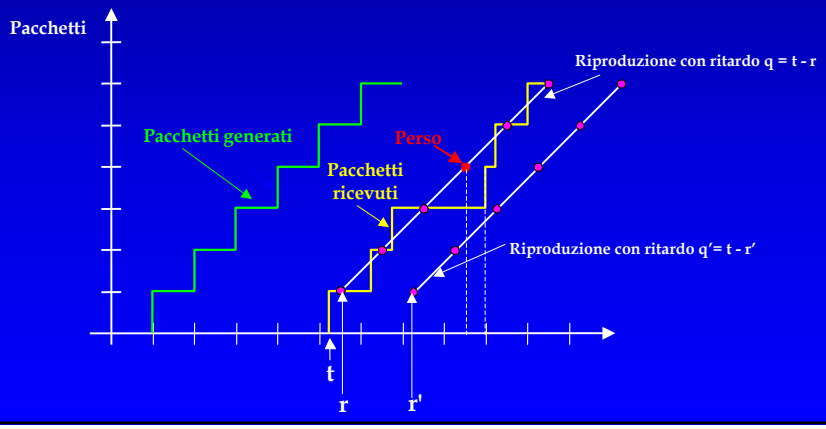
Compensazione del *jitter*

- Il *Jitter* viene in genere compensato utilizzando un buffer ed inserendo un ritardo q di riproduzione, ossia:
  - Se un blocco con un *timestamp* t è ricevuto entro t + q, viene riprodotto all'istante t + q
  - Altrimenti viene scartato
- Questa strategia non richiede numeri di sequenza per gestire le perdite
- Il problema è fissare q
  - q lunghi implicano meno perdite (per *jitter*);
  - q corti: l'interazione è più efficace.

6.13

Compensazione del *jitter*

- La sorgente genera un pacchetto ogni 20 ms.
- Il primo pacchetto è ricevuto all'istante t.
- Nel primo caso (ritardo q) è riprodotto a partire dall'istante r.
- Nel secondo caso (ritardo q') è riprodotto a partire dall'istante r'.



6.14

Compensazione del *jitter*

- Il meccanismo di compensazione del jitter può essere reso più efficace variando in modo adattativo il ritardo di riproduzione.
- In particolare questo si realizza
  - Stimando il ritardo introdotto dalla rete (ed eventualmente la sua varianza).
  - Variando l'istante di inizio di ogni periodo di *parlato* (quando la sorgente è nello stato ***parlo***) , ossia comprimendo o allungando i periodi di silenzio (quelli durante i quali l sorgente è nello stato ***silenzio***).
  - Durante i periodi di *parlato* (*talk spurt*) la riproduzione avviene come nel caso di ritardo fisso.

6.15

Compensazione del *jitter*

- Indicando con
  - t_i = il *timestamp* dell'i-esimo pacchetto
  - r_i =l'istante in cui l'i-esimo pacchetto viene ricevuto
  - p_i =l'istante in cui l'i-esimo pacchetto viene riprodotto
  - d_i = il ritardo medio di rete stimato dopo la ricezione dell'i-esimo pacchetto
  - v_i = la deviazione media del ritardo dopo la ricezione dell'i-esimo pacchetto
- Si può scrivere:

Costante < 1 (ad es. 0,01)

$$d_i = (1-u)d_{i-1} + u(r_i - t_i)$$

$$v_i = (1-u)v_{i-1} + u\,|\,r_i - t_i - d_i\,|$$

- I valori d_i e v_i sono calcolati ogni pacchetto ricevuto, anche se vengono usati solo all'inizio di ogni periodo di *parlato*.

6.16

Compensazione del *jitter*

- Usando questa quantità si può iniziare a riprodurre un *talk-spurt* all'istante:

$$p_i = t_i + d_i + Kv_i$$

- Per ogni singolo pacchetto il ritardo dovrà essere:

$$q_i = p_i - t_i = d_i + Kv_i$$

- E quindi l'istante di riproduzione di un pacchetto r all'interno del *talk-spurt* diventa

$$p_r = t_r + q_i$$

- Per determinare l'inizio di un *talk-spurt* si deve verificare se la differenza fra due successivi *timestamp* con il numero di sequenza (che qui diventa necessario per non farsi ingannare dalle perdite di pacchetti) in ordine corretto è > di 20 msec.

6.17

Compensazione delle perdite

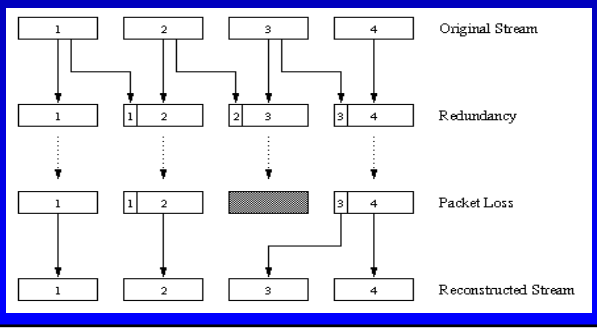
- ***Forward Error Correction* (FEC)**
  - Siccome la perdita significa il non arrivo del pacchetto, le tecniche FEC devono coinvolgere più di un pacchetto.
  - Possiamo fare due esempi di possibili strategie:
- **Strategia 1**
  - Ogni n blocchi, si costruisce un blocco ridondante ottenuto come XOR degli n precedenti e lo si invia in coda al gruppo
  - Se perdo un blocco solo degli n+1, all'istante di ricezione del blocco n+1 lo posso ricostruire
  - La banda aumenta di un fattore 1/n (ad es. 1/10 se n= 10)
  - Il ritardo fisso iniziale d deve essere tale da permettere la ricezione di almeno n+1 pacchetti (se n = 10, d > 200 msec).
  - Quindi aumentando n aumenta l'efficienza ma aumenta anche il ritardo e la probabilità di avere più di una perdita in una sequenza (situazione in cui questo meccanismo diventa inefficace).

6.18

Compensazione delle perdite

Strategia 2

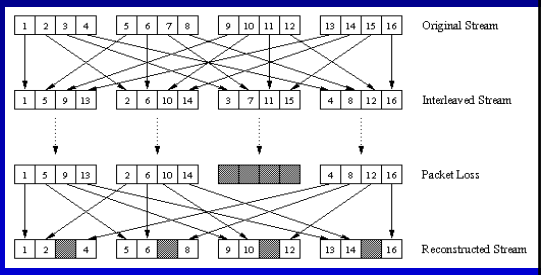
- Si comprime i dati usando due codifiche a risoluzioni diverse: una bassa (B) ed una alta (A) (ad es. un PCM a 64 Kbit/s e un GSM a 13 Kbit/s).
- Si costruisce il pacchetto n prendendo il blocco n del flusso codificato con A (160 Byte) e il blocco n-1 del flusso codificato con B (32,5 byte)
- Quando un pacchetto viene perso può essere ricostruito (con minore qualità) all'arrivo de pacchetto successivo
- Si può proteggere anche nei confronti di due perdite consecutive aggiungendo anche il blocco n-2 del flusso B.



6.19

Compensazione delle perdite

- *Interleaving*
  - I blocchi sono spezzato in unità più piccole (ad esempio in 4 unità da 40 byte).
  - Le unità vengono mescolate in pacchetti successivi che quindi contengono parti di più blocchi.
  - I blocchi originari vengono ricostruiti al ricevitore.
  - Se un pacchetto viene perso la perdita viene distribuita su n blocchi e quindi è più facilmente compensabile.



- Ricostruzione pacchetto/unità persa
  - Si usano dei dati che assomiglino ai precedenti (ripetizione o interpolazione)
  - Funziona abbastanza bene per perdite rare e pacchetti/unità piccole (4-40 msec)

6.20

Real-Time Protocol (RTP)

- Introduce le funzionalità che mancano a UDP per lo *streaming* di contenuti multimediali anche in multicast
- Sviluppato principalmente da H. Schulzrinne nel periodo 1992-1996.
- **`RFC 1889`** definisce il protocollo mentre **`RFC 1890`** fornisce il profilo base per audio/video conferenza.
- RTP risiede nei sistemi terminali (end system)
- I pacchetti RTP vengono incapsulati in segmenti UDP
- Per quanto riguarda l'architettura funzionale RTP è considerato una estensione del protocollo di trasporto
- Dal punto di vista dello sviluppatore è invece parte dell'applicazione, ossia deve essere inglobato nel software applicativo (per lasciare più controllo e flessibilità all'operazione di pacchettizzazione)

6.21

RTP

- In sostanza RTP aggiunge all'UDP le seguenti funzionalità:
  - Identificazione del *payload*
  - Numeri di sequenza
  - *timestamp*
  - Capacità di identificare le sorgenti per la sincronizzazione
- Ad esempio, nel caso di voce PCM a 64 Kbit/s, ad ogni blocco audio viene aggiunta l'intestazione RTP a formare un pacchetto RTP che viene quindi passato all'UDP.
- Il pacchetto RTP permette alla sorgente di riconoscere il tipo di codifica (che la sorgente può modificare potenzialmente pacchetto per pacchetto) e compensare il *jitter* tramite il *timestamp* ed i numeri di sequenza.

6.22

RTP: QdS

- RTP non fornisce nessun meccanismo per assicurare una QdS (ritardo o perdita).
- Il pacchetto RTP è visto solo dai nodi finali e non dai nodi di rete (router) che quindi non forniscono ai pacchetti RTP un trattamento preferenziale.
- Per poter fornire una QdS bisogna utilizzare dei meccanismi aggiuntivi (*Integrated* o *Differentiated Services*)

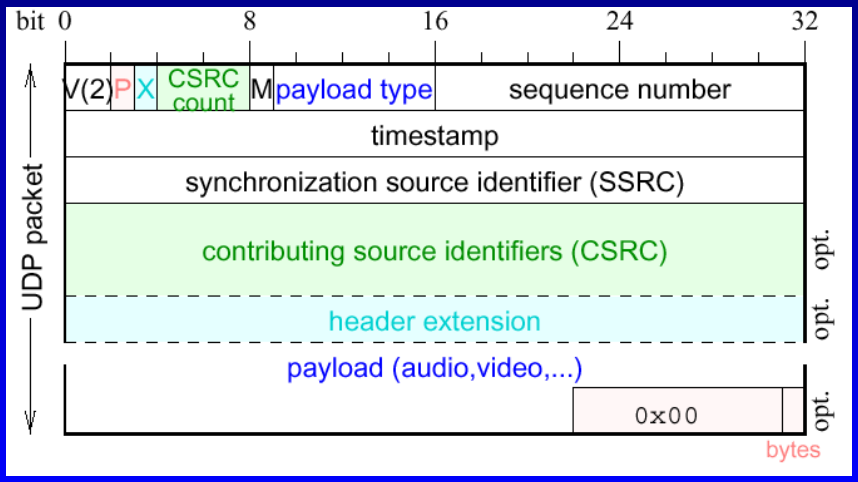
6.23

RTP: *Streams*

- RTP permette di assegnare un flusso (*stream*) indipendente ad ogni sorgente (telecamera, microfono, …).
- Alcuni standard di compressione (MPG1 e 2 ad esempio) permettono l'integrazione dell'audio e video durante il processo di decodifica; in questo caso ci sarà un solo *stream* per ciascuna direzione (comunicazioni unicast).
- Nel caso multicast ogni sorgente ha il proprio *stream* ma tutti gli *stream* usano lo stesso indirizzo multicast e lo stesso albero di distribuzione; la comunicazione tramite RTP all'interno di un gruppo multicast viene chiamata "sessione".

6.24

Il pacchetto RTP



6.25

Il pacchetto RTP

- ***Version*** (2 bit): è la versione, attualmente la 2
- ***eXtention*** (1bit): se a 1 indica la presenza di un *header extention*
- ***Payload Type*** (7 bits): indica il tipo di codifica usata per quel pacchetto (se la sorgente cambia codifica e tramite questo campo che il ricevitore se ne accorge)
  - Payload type 0: PCM mu-law, 64 Kbps
  - Payload type 3, GSM, 13 Kbps
  - Payload type 7, LPC, 2.4 Kbps
  - Payload type 26, Motion JPEG
  - Payload type 31. H.261
  - Payload type 33, MPEG2 video
- ***Sequence Number*** (16 bits): si incrementa di uno ogni pacchetto inviato (serve a riconoscere le perdite).

6.26

Il pacchetto RTP

- *Timestamp* (32 bit): riflette l'istante di campionamento del primo campione presente nel pacchetto ed è derivato dal *clock* del campionatore
  - Ad esempio, viene incrementato di uno ogni campione (per la voce +1 ogni 125 μs e quindi supponendo 8 bit per campione e 160 Bytes a pacchetto, verrebbe incrementato di 160 ogni pacchetto). Il valore viene incrementato anche quanto la sorgente è inattiva ed è riferito al flusso prima della codifica.
- *Sinchronization Source Identifier* (32 bit): Identifica la sorgente, non è un indirizzo IP ma un identificatore generato casualmente.

6.27

Il pacchetto RTP

- *Contributing Source* (CSRC) (32 bit): se un pacchetto contiene segnali di più sorgenti mescolati (tipicamente audio di più parlatori) questo campo permette di identificare le diverse sorgenti originarie (fino a 16).
- *CSRC Count* (4 bit): indica il numero di campi CSRC presenti nel pacchetto.
- Le estensioni trasportano informazioni ulteriori riguardanti ad esempio la codifica.

6.28

Real-Time Control Protocol (RTCP)

- Lavora insieme al RTP.
- Ogni partecipante ad una sessione RTP invia periodicamente pacchetti di controllo RTCP a tutti gli altri partecipanti (multicast).
- Ogni pacchetto contiene informazioni statistiche del ricevitore/trasmettitore utili alle applicazioni.
- Fra le informazioni inviate si trovano, i pacchetti inviati/ricevuti, la percentuale delle perdite, il *jitter*,…
- Questa retroazione può essere usata per verificare le prestazioni e per diagnostica; eventualmente la si può usare per modificare le caratteristiche della trasmissione (parametri della o tipo di compressione, risoluzione, …).

6.29

RTCP

- Tipi di pacchetti ed informazioni inviate
  - ***Receiver report***: frazione di pacchetti persi, ultimo numero di sequenza, jitter di interarrivo medio
  - ***Sender Report***: SSRC dello *stream* RTP, tempo corrente, numero di pacchetti inviati, numero di bytes inviati, *timestamp* dell'ultimo pacchetto RTP inviato.
  - ***Source description***: e-mail del mittente, nome del mittente, SSRC dello stream. Questo pacchetto lega i dati del mittente con l'SSRC.
- Anche i pacchetti RTCP vengono trasportati tramite UDP ma usano una porta diversa rispetto RTP
- Un singolo pacchetto UDP può trasportare più pacchetti RTCP (eventualmente compressi).

6.30

RTCP

- RTCP può essere usato per sincronizzare diversi *stream* all'interno di una sessione
- Si consideri una videoconferenza in cui ciascuna sorgente genera uno *stream* RTP per il video e uno per l'audio.
- I *timestamp* dei pacchetti RTP sono legati ai *clock* dei singoli campionatori, ma non hanno un riferimento temporale comune.
- I *sender report* contengono un riferimento assoluto di tempo e il *timestamp* dell'ultimo pacchetto, per cui permettono di ricostruire un sincronismo, ad esempio, fra voce e video.

6.31

RTCP

- Nel caso multicast, il traffico generato dall'RTCP tenderebbe a crescere linearmente col numero dei partecipanti alla sessione.
- Lo standard prevede invece che il traffico RTCP debba essere limitato al 5% della banda disponibili per la sessione.
- Questo 5% (ad esempio su 2 Mb/s sarebbero 100 Kbit/s) il 75% sono riservati ai ricevitori, 25% alle sorgenti
- L'intervallo di trasmissione per i pacchetti RTCP viene calcolato come:

Per le sorgenti

$$T = (\text{dim pacchetto}) * (\text{num. sorgenti}) / (0{,}25 * 0{,}05 * \text{Banda})$$

Per i ricevitori

$$T = (\text{dim pacchetto}) * (\text{num. ricevitori}) / (0{,}75 * 0{,}05 * \text{Banda})$$

6.32

RTP - RTCP

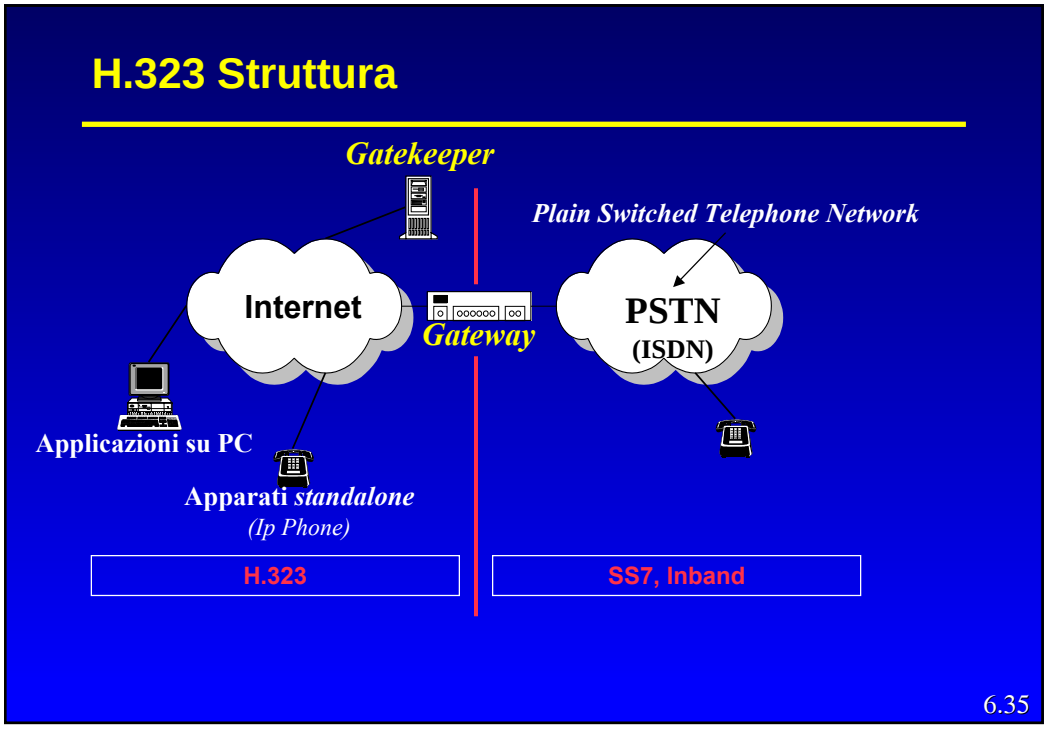
- Alcuni esempi di applicazioni che usano RTP-RTCP sono
  - Strumenti di Mbone: vic/vat/rat
  - Real Media (real audio, real video)
  - MS Netmeeting
  - *Apple Quicktime*
  - H.323
- Attualmente le varie applicazioni non sono sempre in grado di comunicare fra loro
  - Per incompatibilità slegate da RTP
  - Per piccoli scostamenti dallo standard

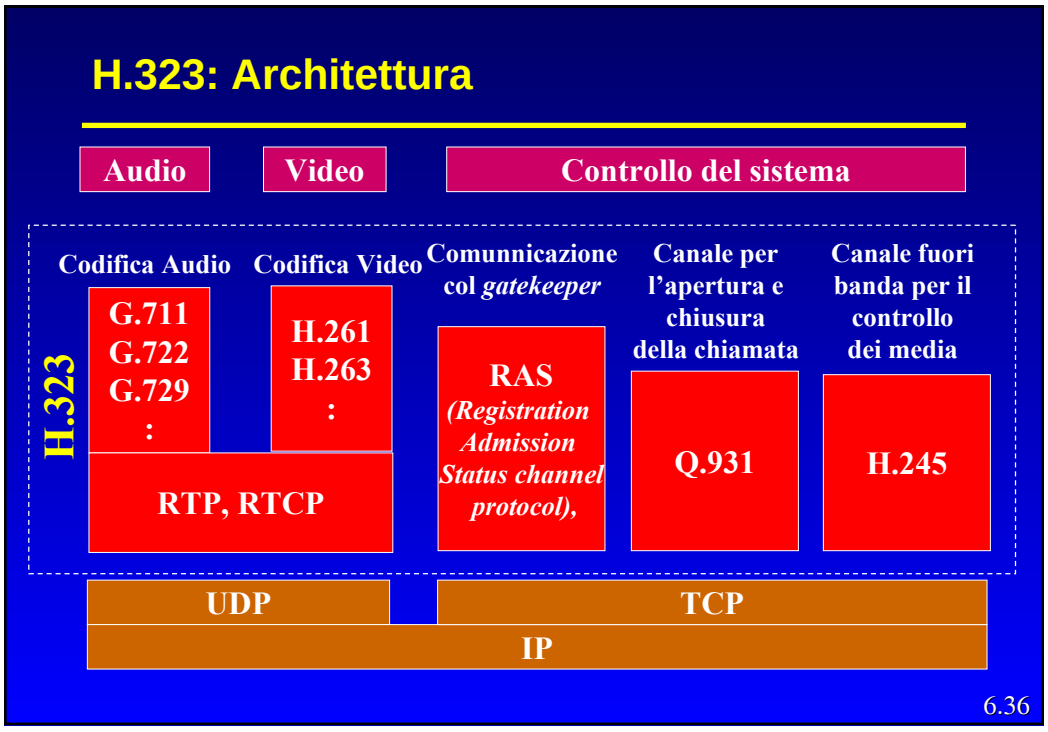
6.33

H.323

- U.323 sono un insieme di protocolli strutturati in una architettura per servizi di video/audio conferenze tramite IP
- Il target sono i servizi interattivi in tempo reale
- Nasce sotto l'egida dell'IUT
- Le specifiche includono:
  - Come i nodi terminali fanno e ricevono le chiamate.
  - Come negoziano le codifiche.
  - Come i flussi vengono incapsulati.
  - Come avviene la sicronizzazione.
  - Come i terminali colloquiano con elementi di coordinamento (*Gatekeeper*).
  - Come i terminali possono interagire con la rete telefonica classica.

6.34


H.323 Struttura
Gatekeeper
Plain Switched Telephone Network
Internet
Gateway
PSTN
(ISDN)
Applicazioni su PC
Apparati standalone
(Ip Phone)
H.323
SS7, Inband
6.35


H.323: Architettura
Audio
Video
Controllo del sistema
Codifica Audio
Codifica Video
Comunnicazione col gatekeeper
Canale per l'apertura e chiusura della chiamata
Canale fuori banda per il controllo dei media
H.323
G.711
G.722
G.729
:
H.261
H.263
:
RAS
(Registration Admission Status channel protocol),
Q.931
H.245
RTP, RTCP
UDP
TCP
IP
6.36

H.323: Codifiche

Audio

- I terminali devo supportare lo standard di compressione voce G.711 a 56-64 Kbps
- Consigliato il supporto del G.723.1 che opera a 5,3 - 6,3 Kbps
- Opzionalmente si possono usare G722, G728 e G729

Video

- Le capacità video sono opzionali
- Se capace di trattare video, un terminale deve supportare formati QCIF H.261 (176 x 144 pixel)
- Opzionalmente può supportare H.263 e dimensioni CIF (352 x 288), 4CIF (704 x 576) e 16CIF (1408 x 1152).

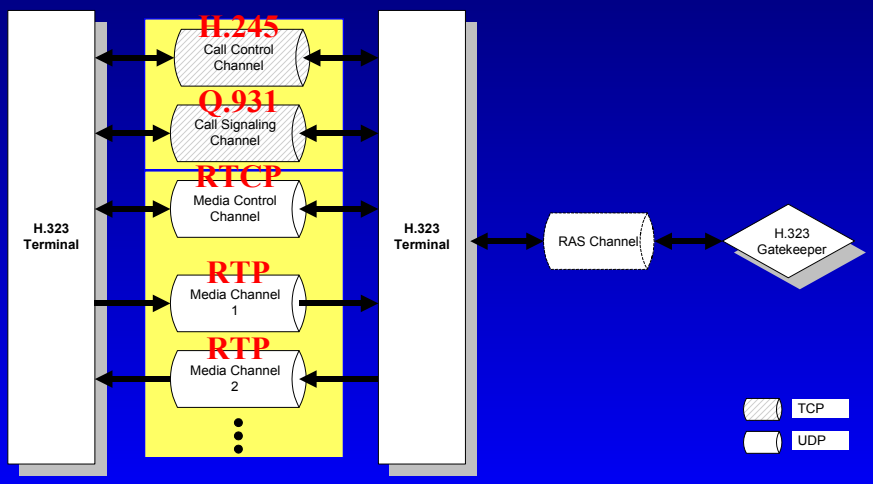
6.37

H.323 Il canale di controllo H.245

- Una sessione H.323 può contenere canali multipli per differenti tipi di media.
- Si ha comunque sempre un unico canale di controllo H.245 che si appoggia su TCP.
- I compiti principali sono
  - Aprire e chiudere i canali relativi ai media.
  - Permettere la negoziazione delle funzionalità (per esempio le codifiche).
- In sostanza H.245 ha, per la sessione di conferenza multimediale, la stessa funzione che RTCP ha per lo *stream* del singolo media.

6.38

H.323 I flussi di informazione



6.39

H.323 *Gatekeeper*

- E' un elemento opzionale del sistema
- Le sue funzioni principali sono:
  - Traduzione di indirizzi IP (come un DNS, ossia tradurre il nome di una persona nel corrispondente indirizzo IP)
  - Gestire la banda, principalmente attraverso un controllo d'accesso
- In genere si immagina che sia presente un *gatekeeper* per area ed il funzionamento dovrebbe essere il seguente:
  - Quando attivata, una applicazione H.323 si deve registrare presso il *gatekeeper* (se esiste), usando il RAS;
  - Nel momento in cui deve fare una chiamata l'applicazione chiede il permesso al GK;
  - Se gli viene accordato il permesso invia un indirizzo email, un nome, un numero telefonico e il GK lo traduce (interagendo eventualmente con altri GK) in un indirizzo IP e lo rimanda all'applicazione.

6.40

Controllo di comunicazioni multimediali

- H.323 non contiene tutti i protocolli e non è l'unica architettura di di riferimento per il controllo di applicazioni multimediali
- Esistono altri protocolli, fra i quali si possono citare per esempio:
  - ***Real Time Streaming Protocol*** (**RTSP**): serve a controllare in remoto una sorgente *stream* (come se fosse un audio/video registratore)
  - ***Session Initiation Protocol*** (SIP)

6.41

Reti integrate nei servizi

- Uno degli obiettivi delle reti di telecomunicazioni nell'ultimo decennio è stato l'integrazione dei servizi: creare una rete in grado di supportare ogni tipo di servizio (previsto e prevedibile)
- Un rete integrata permetterebbe una elevata efficienza e renderebbe appetibile l'offerta dinamica di nuovi servizi di tlc ma richiede una tecnologia molto flessibile

Commutazione di pacchetto

6.42

IP e ATM

- Inizialmente gli standard (ITU-T) e la ricerca hanno proposto ATM (Asynchronous Transfer Mode)
  - Commutazione pacchetto veloce
  - Orientato alla connessione
  - Pensato principalmente per servizi *real-time* (video-voce)
  - ATM non si trova molto "a suo agio" con il traffico dati, al contrario di IP che è invece stato pensato solo per i dati.
- Con Internet, il TCP/IP si è imposto sia come tecnologia che come "infrastruttura" di rete, quindi è sembrato "naturale" proporre **Internet** anche come **rete integrata universale**

6.43

QoS

- Per poter fornire supporto a servizi differenti, bisogna poter assicurare una **Qualità** ossia rispettare dei requisiti diversi in termini
  - Perdita massima ↓
  - Ritardo massimo ↓
  - *Jitter* ↓
  - *Throughtput* ↑
- In corrispondenza di flussi di traffico differenti per
  - Tasso di generazione
  - Statistiche di generazione (CBR, VBR,…)

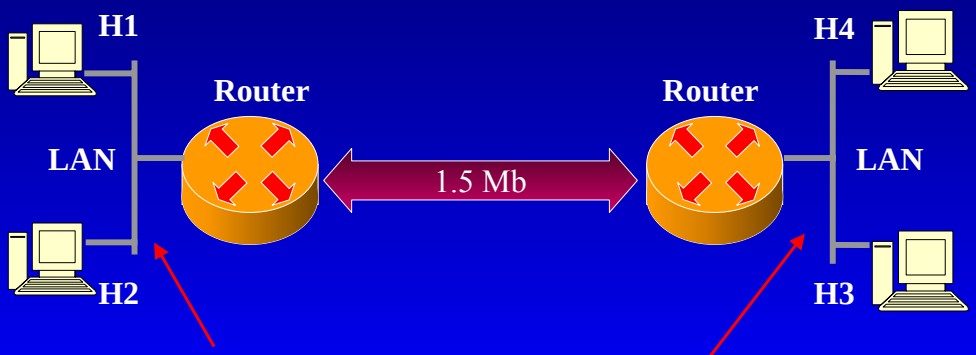
6.44

QoS su reti a pacchetto

- Internet o, in generale, l'architettura TCP/IP è nata ed è attualmente strutturata per fornire solo servizi ***best-effort***, ossia con
  - Perdita di pacchetto
  - Ritardi non superiormente limitati
  - Ritardi variabili (*Jitter*)
- Per poter utilizzare Internet come rete universale, bisogna poter assicurare un grado di QoS ad applicazioni o utenti che ne facciano richiesta, e quindi bisogna introdotti dei **nuovi componenti architetturali**.

6.45

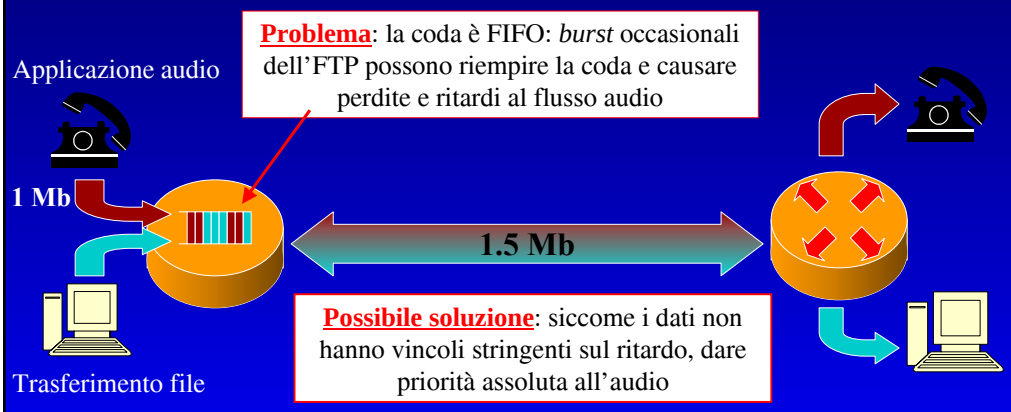
Scenario di riferimento



Si supponga che la LAN non sia un elemento critico (nel senso che ha una velocità molto più alta di quella presente fra i due *router*)

6.46

Scenario 1



Problema: la coda è FIFO: *burst* occasionali dell'FTP possono riempire la coda e causare perdite e ritardi al flusso audio

Possibile soluzione: siccome i dati non hanno vincoli stringenti sul ritardo, dare priorità assoluta all'audio

Per poter servire differentemente i due traffici i *router* devono poter distinguere i pacchetti di classi diverse, ossia è necessario un "***packet marking***" per classe di traffico

6.47

Scenario 1

- Se l'utente del trasferimento file ha pagato molto per avere un servizio di "prima classe", mentre l'utente audio ha pagato poco per avere un servizio a basso prezzo, non è detto che la priorità dell'audio debba essere assoluta.
- Ci sono tre considerazioni da fare:
  - Il *Marking* esplicito serve solo a distinguere i pacchetti ma non determina necessariamente il tipo di servizio che riceverà un pacchetto.
  - Il modo con cui il *router* classifica (*classification*) i pacchetti può basarsi su diversi criteri (sorgente, destinazione, stato della linea) di cui l'identificazione esplicita è solo uno degli elementi
  - Il modo in cui i diversi pacchetti vengono classificati e poi trattati dipende dalla "politica" di servizio.

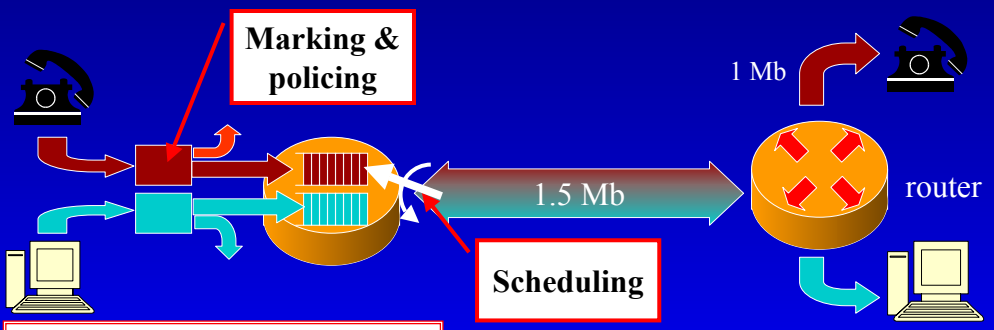
Non basta fare il marking per assicurare una QoS

Occorrono anche altri meccanismi

6.48

Scenario 2

L'applicazione audio (volontariamente o per errore) comincia a generare più di 1 Mbps; se arriva a generare ≥ 1.5 Mbps, l'FTP non passa più



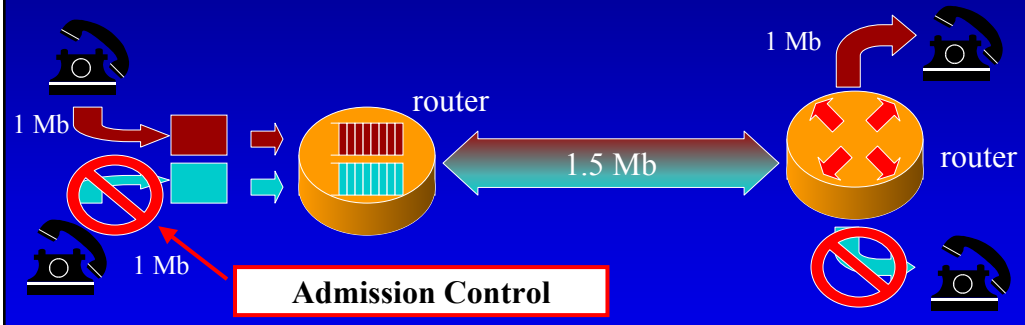
Bisogna poter fornire un **certo grado di indipendenza fra i flussi**, per evitare che flussi "sbagliati" influenzino negativamente le prestazioni degli altri

Si osservi però che l'isolamento fra flussi deve essere ottenuto mantenendo il **massimo grado di efficienza** nell'uso delle risorse (banda e buffer)

6.49

Scenario 3

Se si attivano due applicazioni audio da 1 Mbps, qualunque *classification* o *policing* o partizione (*scheduling*) non permette di mantenere la QoS



Bisogna che le applicazioni dichiarino i propri flussi e la richiesta di QoS e che subiscano un "**Controllo di Accesso**"

6.50

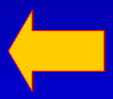
Elementi

- In sostanza gli elementi che devono essere considerati per fornire una QoS su IP sono:
  - **Classificazione dei pacchetti (*Packet Classification*)**
  - **Isolamento dei flussi: *Scheduling* e *Policing***
  - **Efficace utilizzazione delle risorse (Allocazione dinamica delle risorse)**
  - **Controllo di ammissione (*Call Admission*)**

6.51

IP-QoS

- L'IETF ha proposto due approcci diversi (ma non necessariamente mutualmente esclusivi):
  - **Integrated Services**
  - **Differentiated Services**



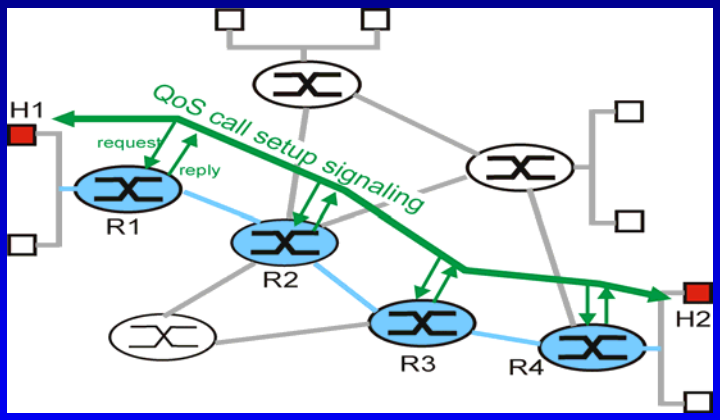
6.52

Integrated Services (IntServ)

- In sostanza, si porta la filosofia ATM in IP:
  - Tramite cinque parametri (indirizzo destinazione, ind. sorgente, protocollo, porta sorgente, porta destinazione) i pacchetti vengono identificati "su base connessione o flusso" *(marking)*.
  - Quando un flusso chiede di essere attivato la sorgente deve dichiarare:
    - » La QdS richiesta (definiti da un insieme di dichiarazioni dette **R-spec**)
    - » Le caratteristiche del traffico (descritte nel **T-spec**)
  - un protocollo (RSVP) di segnalazione trasporta i T-Spec e R-spec e chiede ai *router* lungo il percorso verso la destinazione di onorarli (riservando la banda); se questo non è possibile ne informa la sorgente che non attiva il flusso (*Call Admission*).
  - Si ha quindi una sorta di "Circuito Virtuale" che richiede la presenza di uno stato per flusso nei *router*.

6.53

Integrated Services (IntServ)



6.54

Integrated Services (IntServ)

- I router devono assicurare banda e QoS a ciascun flusso (tramite uno *scheduling* fra i flussi)
- All'ingresso della rete sul flusso di ogni connessione è applicato un "*policing*" (ed eventualmente uno "*shaping*") per mantenerli sempre conformi al dichiarato.
- Sono state definite due classi generali di servizio per le quali riservare banda:
  - » ***Guaranteed Quality Service***: a cui si assicura banda, ritardo massimo limitato e perdita nulla
  - » ***Controlled Load Network Service***: a cui si assicura le stesse prestazioni del best-effort in caso di rete poco carica

6.55

Integrated Services (IntServ)

- Il protocollo principale legato a questo approccio è il ***ReSerVation Protocol*** (**RSVP**).
- Si tratta, in sostanza, di un protocollo di segnalazione che permette ai *router* (ed eventualmente agli *host*) di interagire per attivare e configurare flussi con QdS assicurata.
- Non definisce meccanismi da applicare per assicurare banda e ritardo, e neppure come calcolare quanta banda riservare ai diversi flussi e come decidere se accettare o meno un nuovo flusso.
- Non è un protocollo d'instradamento e non influenza l'instradamento.

6.56

Integrated Services (IntServ)
RSVP

- E' definito nell'RFC 2205
- Ha le seguenti caratteristiche principali:
  - E' pensato per operare anche in presenza di multicast.
  - E' di tipo simplex (opera la segnalazione solo per una direzione).
  - La prenotazione è guidata dal/dai ricevitori;
  - Utilizza i "***soft state***".
  - Permette di "aggregare" secondo modalità diverse le richieste di banda delle sorgenti.
  - Opera direttamente sopra IP.

6.57

Integrated Services (IntServ)
RSVP

- RSVP opera tramite due messaggi:
  - **Path**
  - **Resv**
- Un trasmettitore che vuole iniziare a trasmettere un flusso invia un messaggio di Path al gruppo, il quale fluisce nell'albero di distribuzione e fa si che ogni *router* attraversato memorizzi il salto (hop) inverso verso la sorgente.
- Ogni ricevitore risponde con un messaggio di Resv che contiene le sue richieste e che fluisce in senso inverso fino alla sorgente.
- Ogni *router* che riceve il Resv verifica la disponibilità di risorse e (se disponibili) le riserva, creando un *softstate* per il flusso e facendo proseguire il Resv verso la sorgente (conosce l'*hop* grazie al messaggio di Path precedente).

6.58

Integrated Services (IntServ)
RSVP

- Se il *router* è nel punto di incontro di più rami dell'albero di distribuzione del multicast, opera effettuando una "fusione" delle richieste.
- Anche le richieste legate a più sorgenti di uno stesso gruppo possono venir "fuse" secondo uno di tre stili di prenotazione:
  - Filtro aperto: unica prenotazione da condividere da parte di tutte le sorgenti.
  - Filtro fisso: elenco di sorgenti e corrispondente prenotazione per ciascuna
  - Condivisione esplicita: elenco di sorgenti e unica prenotazione da condividere solo fra queste.

6.59

Integrated Services (IntServ)
Conclusioni

- Il distinguere ciascun flusso ha il vantaggio di permettere una precisa allocazione delle risorse ma ha il grosso limite di essere **poco scalabile**.
- I *router* delle dorsali, che commutano molti flussi e gestiscono capacità molto elevate, potrebbero non riuscire a gestire una QoS per flusso in modo efficiente.
- Per cui, l'approccio IntServ appare realistico all'interno di "aree" di rete ristrette e/o applicato insieme a delle tecnologie di supporto per la QoS (ATM o MLSP)

6.60

Differentiated Services

- Il vincolo principale a cui ha cercato di sottostare questo approccio è permettere una agevole scalabilità
- Con questo proposito si è introdotto (rispetto agli *IntServ*):
  - L'aggregazioni dei flussi in classi
  - La differenziazione delle funzionalità dei *router*
  - La riduzione del traffico di segnalazione
  - La riduzione delle variabili di stato nei *router*

6.61

Differentiated Services

- Il campo ToS (*Type of Service*) dell'IP viene rinominato **DS *field*** ed usato per identificare delle "**classi di servizio**".
- Ad ogni classe viene legato un
  Per Hop Behaviour (PHB)
  che stabilisce il tipo di trattamento che ad ogni *router* deve dare ai pacchetti appartenenti ad una certa classe.

6.62

PHB

- ***Expedited Forwarding*** (EF)
  - E' pensato per costruire servizi *end-to-end* a
    - » bassa perdita
    - » bassa latenza
    - » basso *Jitter*
    - » banda assicurata
      (VLAN, Videoconferenza, voice over IP, …)
  - Ai due estremi appare come una "linea dedicata virtuale".
  - Impone, in sostanza, un servizio prioritario ed una allocazione di risorse indipendente dalle altri classi

6.63

PHB

- ***Assured Forwarding*** (AF)
  - Ha l'obiettivo di permettere un servizio accettabile anche in condizioni di congestione della rete.
  - Al suo interno si possono distinguere più sotto classi (AF1-AF4).
  - I pacchetti possono essere "marcati" con tre diversi colori (verde, giallo e rosso) che implicano diversi livelli di precedenza in caso di scarto (*drop*).

6.64

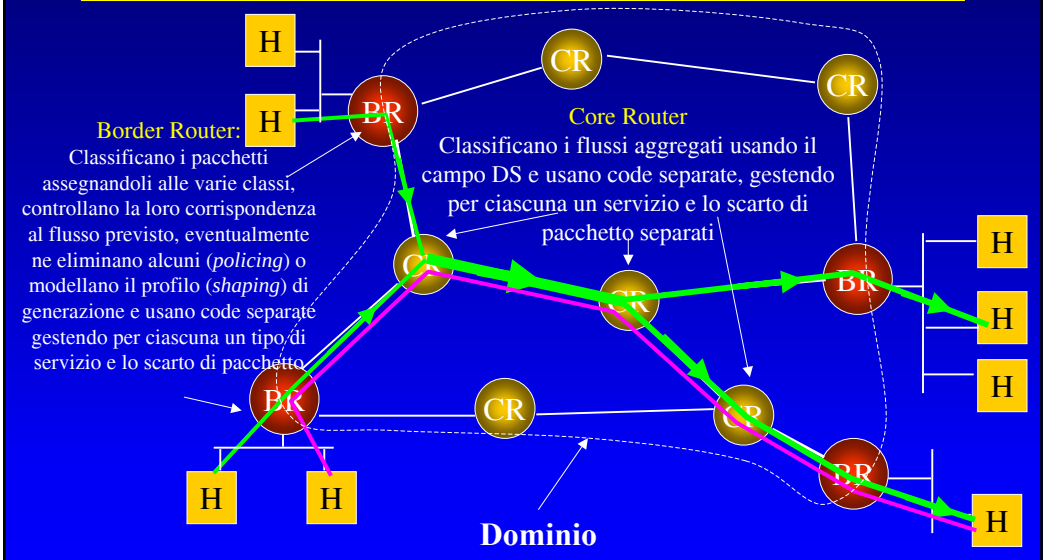
PHB

- *Default Forwarding* (DF)
  - Sono i pacchetti non esplicitamente marcati, ossia l'attuale traffico *best effort*
  - Questa classe, i n linea di principio, potrebbe venir trattata come un AF con la priorità più bassa in assoluto o considerata a parte.

6.65

Differentiated Services

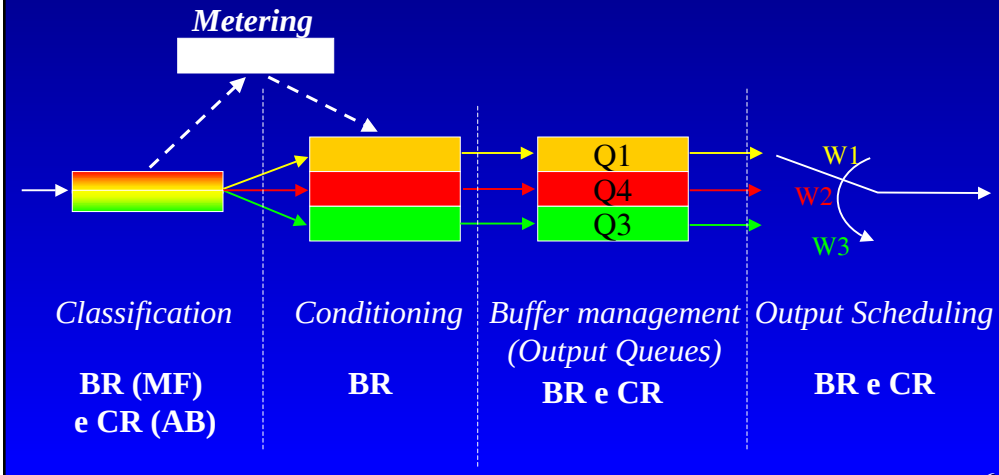
Schema di principio



6.66

Router Components

- Componenti dei *router Diffserv*



6.67

Classification

- La classificazione serve a decidere come trattare i pacchetti.
- Tipi di classificazioni
  - ***Behaviour Aggregate*** (BA): il *router* usa solo campo DS per decidere come trattare il pacchetto.
  - ***Multi- Field*** (MF): il *router* usa cinque campi dell'intestazione IP/TCP (indirizzo sorgente/destinazione, porta sorgente/destinazione e IP *protocol number)*
  - Ci possono essere anche altri tipi di classificazione (basate, per esempio, sugli indirizzi MAC).

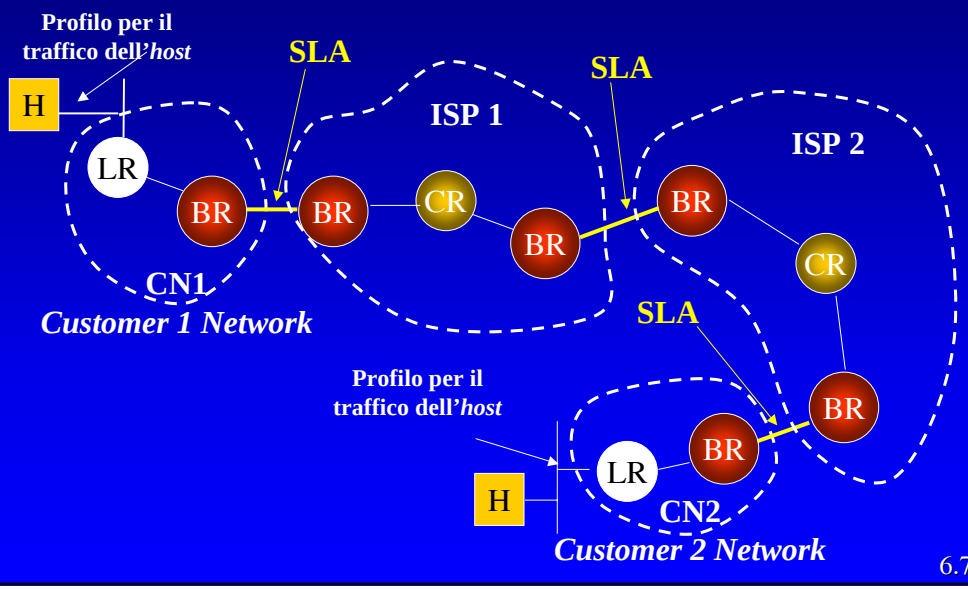
6.68

Service Level Agreement (SLA)

- In corrispondenza dell'attivazione di servizi differenziati, deve venir stabilito un "profilo di traffico" a cui il flusso generato deve sottostare.
- Questi profili sono definiti tramite ***Service Level Agreement*** (**SLA**), che definiscono le classi supportate e il livello di traffico con le sue caratteristiche.
- Ogni area di confine, fra *host* e "*Leaf Router*" (LR), fra rete utente e ISP, fra domini diversi e diversi ISP, devono essere definiti dei Profili di Traffico o degli SLA che stabiliscano, per ogni classe, le caratteristiche del flusso.

6.69

SLA



6.70

SLA

- Gli SLA possono essere
  - statici (definiti per lunghe durate, mesi/anni).
  - dinamici, cioè attivati su richiesta, all'occorrenza.
- Per poter gestire gli SLA dinamici occorre un protocollo di segnalazione (RSVP?).
- Gli SLA dinamici devono ovviamente essere soggetti ad un controllo di accesso.

6.71

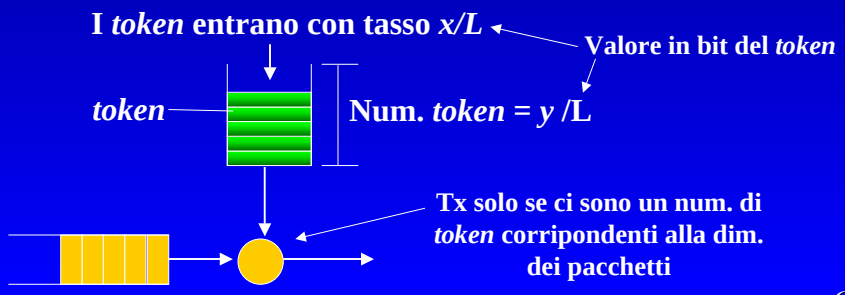
Metering e Conditioning

- Il *Metering* serve a verificare che i flussi in ingresso sia conformi agli SLA o ai profili definiti.
  - Agisce effettuando delle misure sui tassi medi e di picco
- Il *Conditioning* serve a intervenire sul traffico quando questo non risulta conforme al previsto
- Può operare le seguenti azioni
  - Riclassificare i pacchetti non conformi cambiando il valore del DS all'interno della classe o cambiandoli classe (ad es. portandoli alla classe *best-effort*)
  - Applicare degli *shaper*, ad esempio un *Leaky Buket*, per rimodellare il flusso
  - Eventualmente scartare dei pacchetti

6.72

Shaping- Policing

- *Linear bound arrival process*:
  (num. bit trasmessi in t) ≤ $xt+y$
  x = tasso a lungo termine
  y = lungh. massima *burst* (deviazione massima)
- Implementazione: ***Leaky Bucket***



6.73

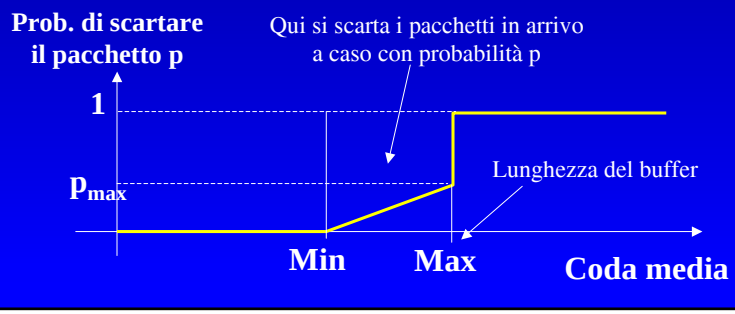
Buffer managenment

- In un caso semplice, la perdita di pacchetti, avviene in modo "naturale" in corrispondenza di un sovraccarico temporaneo, per riempimento della coda: i pacchetti che arrivano quando la coda è piena vengono scartati (*dopped*).
- Questo modo di procedere non è, generalmente, il più efficace perché da luogo a "*burst*" di pacchetti consecutivi persi che hanno due effetti:
  - Peggiorano la qualità di flussi non controllati
  - Tendono a sincronizzare i controlli di flusso del TCP dando luogo a comportamenti oscillatori poco efficienti

6.74

Random Early Detection (RED)

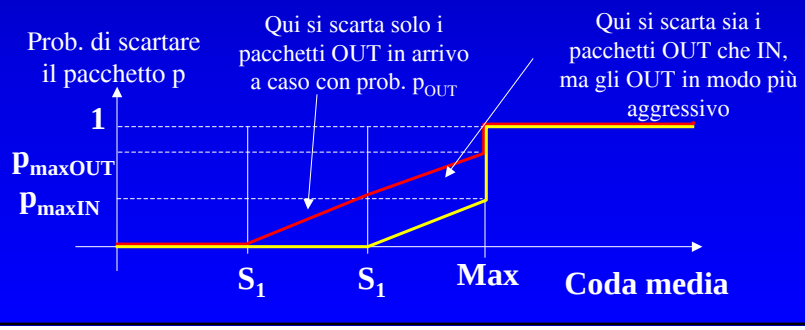
- Un modo per porre rimedio a questo problema è cercare di rendere le perdite "casuali".
- Un metodo per farlo prende il nome RED (*Random Early Detection*); al di sopra di una certa soglia sulla coda media (media esponenziale) introduce una probabilità di perdita che varia linearmente con il valore della coda media



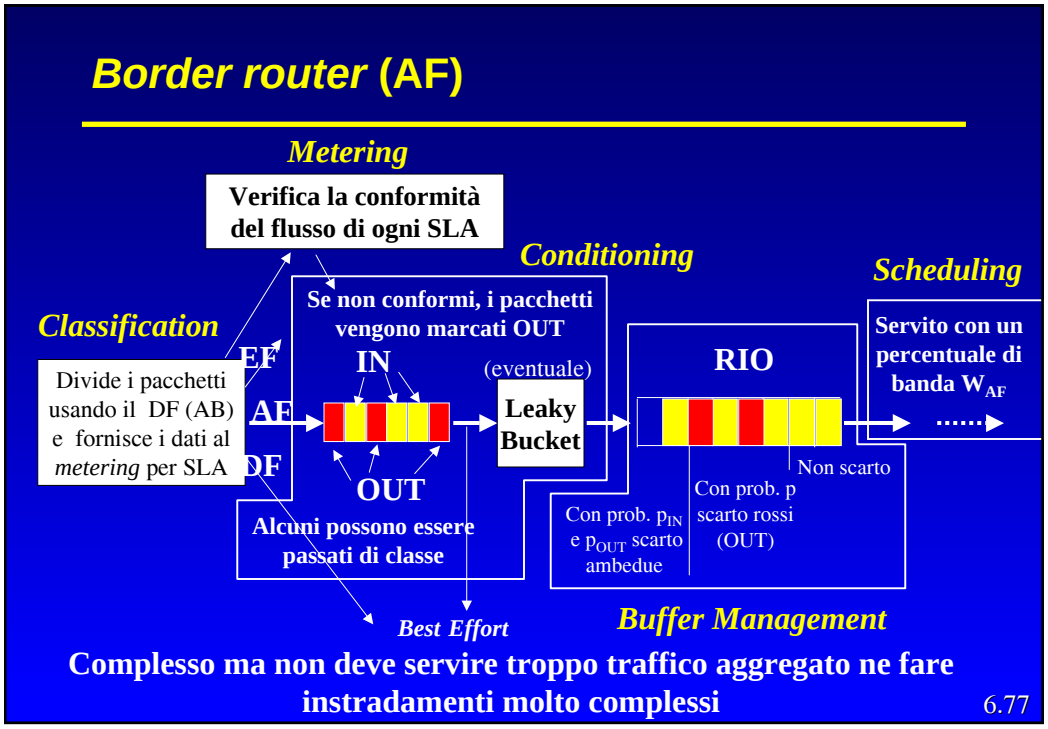
6.75

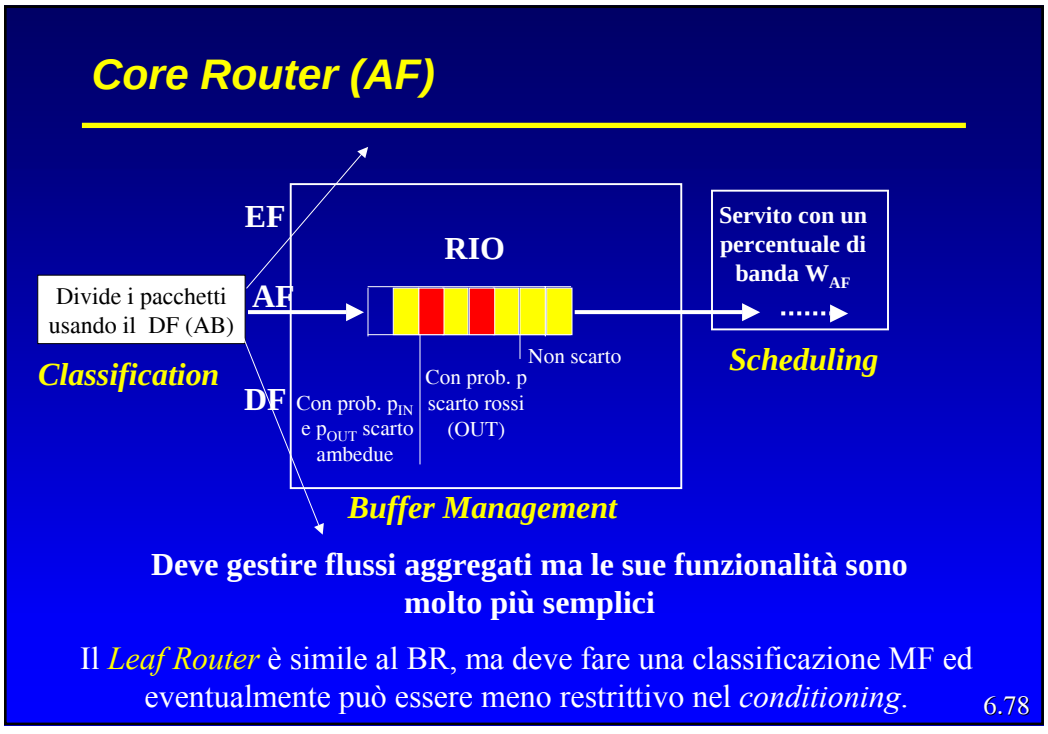
RIO

- Una seconda tecnica più sofisticata si chiama *Random early detection with In and Out* (RIO).
  - In questo caso si suppone la presenza di due classi di pacchetti: IN e OUT (gialli e rossi).
  - Si applicano due RED separati per ciascuna classe:



6.76


Border router (AF)
Metering
Verifica la conformità del flusso di ogni SLA
Conditioning
Scheduling
Classification
Se non conformi, i pacchetti vengono marcati OUT
Servito con un percentuale di banda W_{AF}
Divide i pacchetti usando il DF (AB) e fornisce i dati al metering per SLA
EF
AF
DF
IN
(eventuale)
Leaky Bucket
RIO
OUT
Non scarto
Con prob. p_{IN} e p_{OUT} scarto ambedue
Con prob. p scarto rossi (OUT)
Alcuni possono essere passati di classe
Best Effort
Buffer Management
Complesso ma non deve servire troppo traffico aggregato ne fare instradamenti molto complessi
6.77


Core Router (AF)
EF
RIO
Servito con un percentuale di banda W_{AF}
Divide i pacchetti usando il DF (AB)
AF
Classification
Scheduling
DF
Non scarto
Con prob. p scarto rossi (OUT)
Con prob. p_{IN} e p_{OUT} scarto ambedue
Buffer Management
Deve gestire flussi aggregati ma le sue funzionalità sono molto più semplici
Il Leaf Router è simile al BR, ma deve fare una classificazione MF ed eventualmente può essere meno restrittivo nel conditioning.
6.78

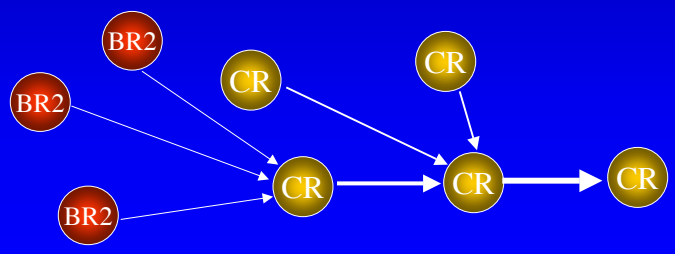
Expedited Forwarding (EF)

- Per i EF il discorso è simile al AF con alcune differenze:
  - Il traffico è sottoposto a *shaping* sia ai LR che ai BR
  - Il traffico in eccesso (sul tasso di picco) non è cambiato di classe ma scartato
  - Lo *scheduler* serve con priorità elevata il traffico EF, indipendentemente dal carico generato dagli altri traffici.
  - Il traffico EF è soggetto a controllo di accesso (gli SLA possono essere sia statici che dinamici).
  - La quantità di traffico EF ammesso nelle rete deve essere una percentuale ridotta (10%) della capacità totale del traffico in rete.

6.79

Expedited Forwarding (EF)

- Con questi presupposti, dato che in genere tutti i *link* in un *router* sono *full-duplex* e quindi la capacità in ingresso è uguale a quella d'uscita, se il flusso EF è minore del 10% ed è prioritario subisce sempre pochissima perdita e ritardo.
- In realtà non necessariamente detto che questo sia vero (perché sia così bisogna usare degli algoritmi di *routing* per la QoS o fare un management specifico)



6.80

Allocazione di Banda - BB

- Sebbene gli elementi architettura siano stati definiti, in realtà non è stato definito l'elemento che determina l'allocazione delle risorse ed eventualmente effettua il Controllo d'accesso.
- Tale elemento viene in genere indicato col nome di

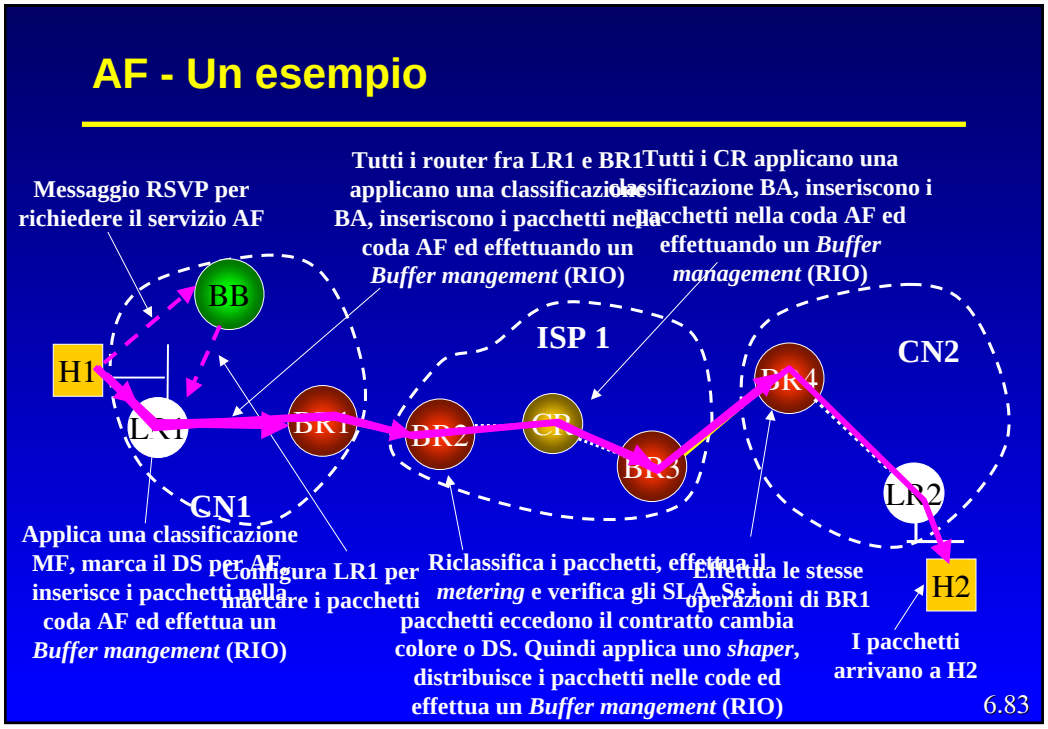
  Bandwidth Broker
- Si tratta di una entità logica che risiede in ogni dominio (di utente o di ISP).
- Nella rete dell'utente, il BB interagisce con l'host all'attivazione del sevizio e configura il LR e tutti i *router* intermedi per permettere la realizzazione del servizio

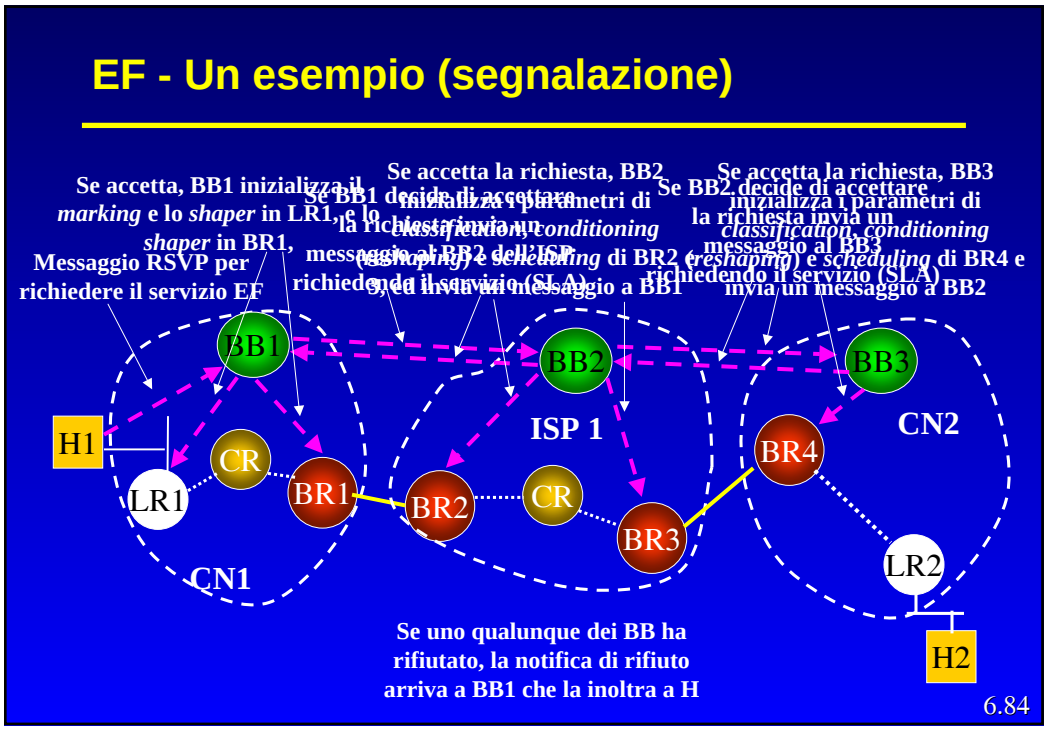
6.81

Allocazione di Banda - BB

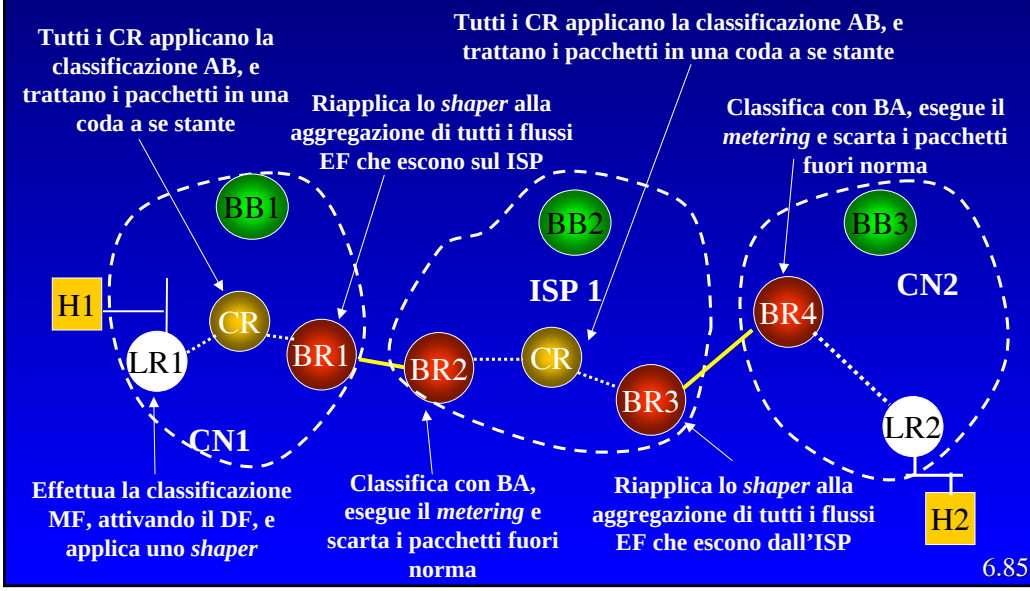
- All'interno degli ISP, se gli SLA sono statici, non è strettamente necessario un BB, in quanto le risorse possono essere allocate via *management*
- Se ci sono SLA dinamici, i BB sono necessari per configurare i *router* (BR e CR) e devono essere in grado di colloquiare fra loro anche fra domini diversi.
- Sebbene definiti architetturalmente, non sono definite (e neppure banali) le politiche attraverso le quali le risorse devono essere allocate.

6.82


AF - Un esempio
Messaggio RSVP per richiedere il servizio AF
Tutti i router fra LR1 e BR1 applicano una classificazione BA, inseriscono i pacchetti nella coda AF ed effettuando un *Buffer mangement* (RIO)
Tutti i CR applicano una classificazione BA, inseriscono i pacchetti nella coda AF ed effettuando un *Buffer management* (RIO)
BB
H1
LR1
BR1
BR2
CR
BR3
BR4
LR2
H2
ISP 1
CN1
CN2
Applica una classificazione MF, marca il DS per AF, inserisce i pacchetti nella coda AF ed effettua un *Buffer mangement* (RIO)
Configura LR1 per marcare i pacchetti
Riclassifica i pacchetti, effettua il *metering* e verifica gli SLA. Se i pacchetti eccedono il contratto cambia colore o DS. Quindi applica uno *shaper*, distribuisce i pacchetti nelle code ed effettua un *Buffer mangement* (RIO)
Effettua le stesse operazioni di BR1
I pacchetti arrivano a H2
6.83


EF - Un esempio (segnalazione)
Se accetta, BB1 inizializza il *marking* e lo *shaper* in LR1, e lo *shaper* in BR1, e invia un messaggio a BB2
Se accetta la richiesta, BB2 inizializza i parametri di *classification, conditioning* (*reshaping*) e *scheduling* di BR2 e invia un messaggio a BB1
Se BB2 decide di accettare la richiesta invia un messaggio al BB3 richiedendo il servizio (SLA)
Se accetta la richiesta, BB3 inizializza i parametri di *classification, conditioning* (*reshaping*) e *scheduling* di BR4 e invia un messaggio a BB2
Messaggio RSVP per richiedere il servizio EF
Se BB1 decide di accettare la richiesta invia un messaggio al BB2 dell'ISP richiedendo il servizio (SLA)
BB1
BB2
BB3
H1
CR
LR1
BR1
BR2
CR
BR3
BR4
LR2
H2
ISP 1
CN1
CN2
Se uno qualunque dei BB ha rifiutato, la notifica di rifiuto arriva a BB1 che la inoltra a H
6.84

EF - Un esempio (fase dati)



Altri elementi importanti

- *MultiProtocol Label Switching* (MPLS)
- QoS *Routing* (*Constraint based routing*)
- *Traffic Engineering*
- ATM

6.86