Reti di Telecomunicazioni

Controllo del traffico

Autori

Queste slides sono state scritte da

MicheleMichelotto:

michele.michelotto@pd.infn.it

che ne detiene i diritti a tutti gli effetti

Copyright Notice

Network layer

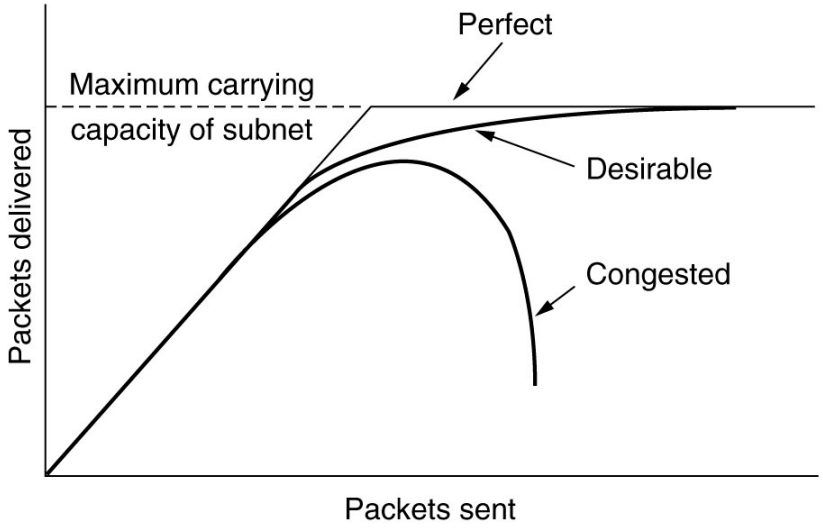
Introduzione
Layer: Modello OSI e TCP/IP
Physics Layer
Data Link Layer
MAC sublayer
Network Layer
Transport Layer
Application Layer

Problema: la congestione

- Per una subnet con ***n*** router ognuno con ***k*** vicini ho bisogno di memoria pari a ***nk.*** Quindi posso avere problemi di memoria e anche di tempo di calcolo
- Grandi problemi succedono se
  - il router dice di avere una linea che non ha
  - o non dichiara una linea che ha
  - o non forwarda correttamente i pacchetti
  - o li corrompe mandandoli
  - Infine il router può esaurire la memoria o sbagliare nel calcolare il routing.
- Quando la rete comincia ad avere decine o centinaia di migliaia di router la probabilità che questo accada diventa non trascurabile

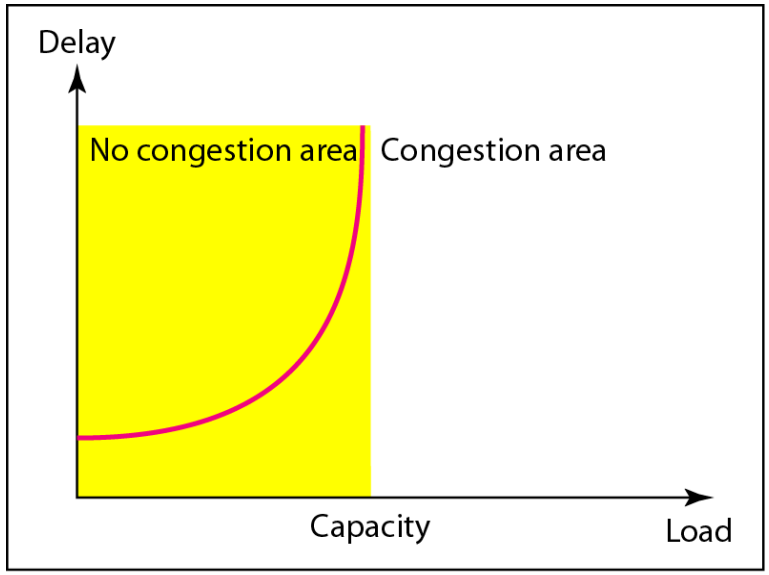
Gestione Congestioni

- Quando ci sono pochi pacchetti in ingresso alla rete, tutti vengono consegnati
  - A parte quelli che subiscono errori di trasmissione
  - Ho quello che mi aspetto, una risposta proporzionale
- Quando ho troppi pacchetti in una rete le prestazioni si degradano

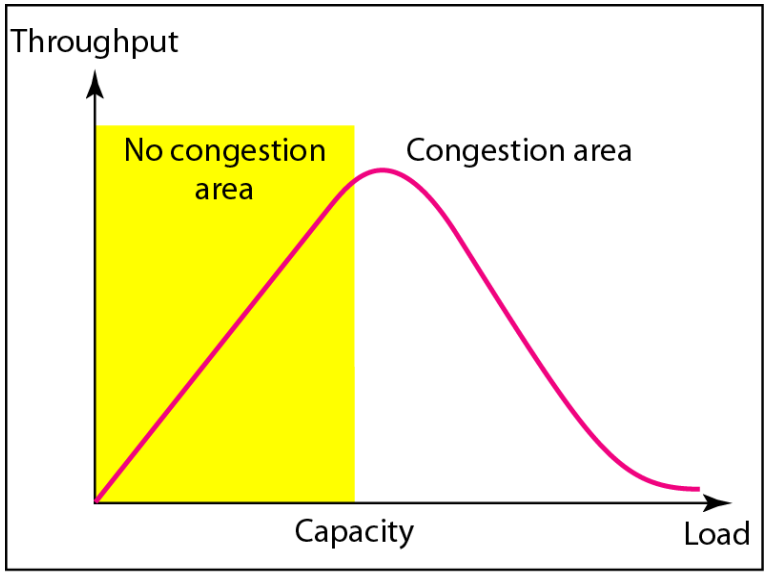


Delay e Throughput

- Abbiamo visto come all'avvicinarsi della capacità massima del sistema il delay tenda a infinito
- E come visto nella slide precedente il throughput descresce fino a quasi zero



a. Delay as a function of load



b. Throughput as a function of load

Cause di congestione

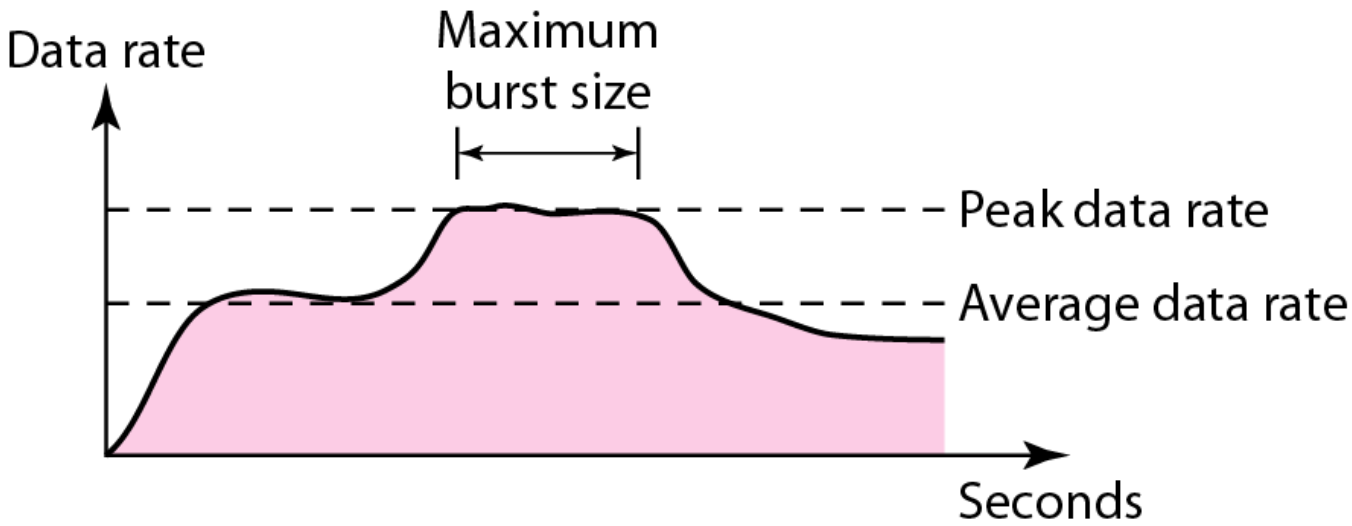
- Se arrivano flussi di pacchetti da diverse linee di ingresso, tutte dirette alla stessa linea di uscita, si forma una coda
  - Se non c'è memoria sufficiente nel router i pacchetti vanno persi
  - Aggiungendo memoria la situazione migliora ma per assurdo (Nagle 97) se il router avesse memoria infinita la congestione andrebbe in peggio invece che in meglio
  - Infatti quando i pacchetti riescono ad arrivare in testa alla coda sono già andati ripetutamente in timeout e quindi creati e spediti dei duplicati

Router lenti

- Processori lenti nei router causano congestione
  - Se la CPU è troppo lenta per gestire i pacchetti (accodare i buffer, aggiornare le tabelle) di nuovo si formano delle code, anche se in uscita ho delle linee veloci
  - Analogamente linee a bassa capacità causano congestione.

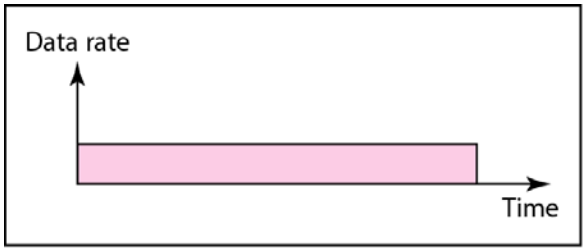
Descrizione del traffico

- Il traffico è caratterizzato non solo dal dalla velocità media (average data rate) ma anche da quello di picco
- E dalla larghezza del tempo dei picchi che sono raggiunti per brevi momenti
- Se le raffiche (**burst**) sono molto veloci (pochi millisecondi) possono essere ignorate ma se durano diversi secondi possono causare problemi

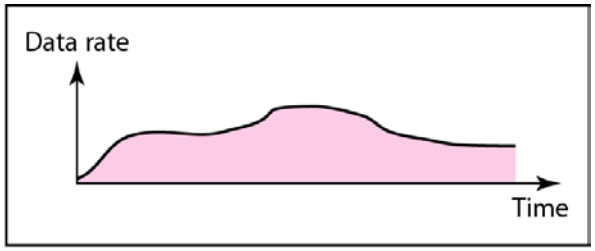


Profili di traffico

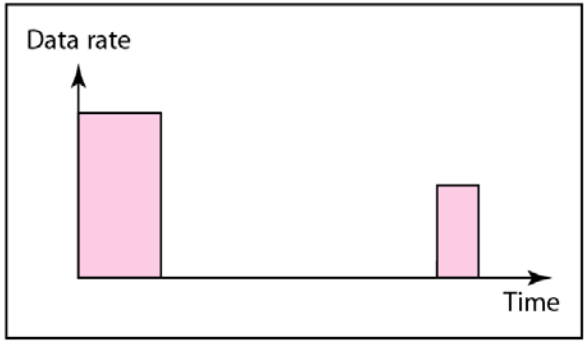
- Traffico costante. Velocità massima (di picco) coincide con quella media
- Traffico variabile. Cambiamenti graduali con brevi raffiche (burst) di dati. Più difficile da gestire del precedente
- Traffico a burst, tipico di LAN to LAN, da zero a 10 Mbps in pochi microsecondi. Il più difficile da gestire



a. Constant bit rate



b. Variable bit rate



c. Bursty

Soluzioni globali

- **Un upgrade delle linee ma non dei router** o viceversa **dei router ma non delle linee** aiuta per un po', ma spesso sposta solo il collo di bottiglia in un'altra parte del sistema
- Il problema reale di solito è uno sbilanciamento tra le diverse parti del sistema.
- Il problema persiste fino a quando tutte le componenti trovano un equilibrio

Congestion or flow control

- Differenza tra **controllo della congestione** e **flow control**
- Il **controllo della congestione** è legato alla capacità della rete di portare il traffico immesso, è un problema globale che coinvolge gli host, i router, la bufferizzazione all'interno dei router
- **Flow control** invece è legato al traffico punto punto tra un mittente ed un destinatario. Deve fare in modo che un mittente veloce non trasmetta dati più velocemente di quanto il ricevente li posso gestire
- Nel flow control spesso c'è un feedback dal ricevente al mittente per dire come vanno le cose all'altro capo

Confusi?

- La confusione viene dal fatto che gli algoritmi di controllo della congestione operano mandando indietro verso le sorgenti di traffico dei messaggi in cui chiedono di rallentare prima che la rete abbia problemi
- Quindi un host può ricevere un messaggio “RALLENTA!” sia perché il ricevente non riesce a gestire il ritmo di spedizione sia perché la rete non riesce a gestirlo

Esempio flow control

- Ho una rete ottica con 1000 Gbps di capacità su di un supercomputer che deve trasferire dati ad un PC a 1Gbps.
- Non c’è congestione, la rete in sè, non ha problemi, ma ho bisogno di flow control per impedire al supercomputer di schiantare il mio povero PC

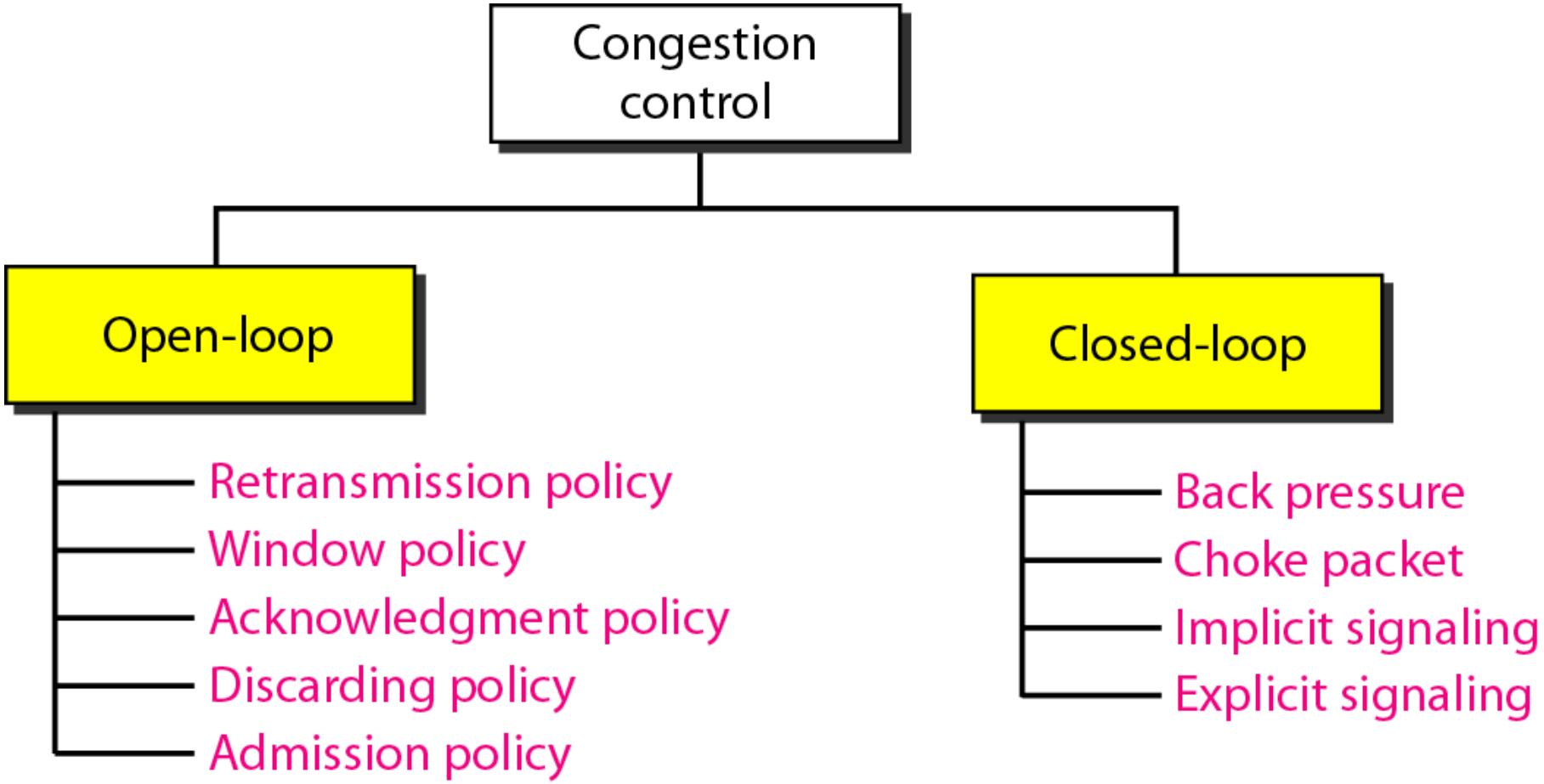
Esempio congestione

- Abbiamo una rete store-and-forward con linee a 1 Mbps e 1000 computer, metà dei quali vuole trasmettere file a 100 kbps all'altra metà
- Nessun mittente causa problemi al suo receiver data la lentezza del trasferimento ma il traffico totale imposto alla rete è superiore alla sua capacità

Due gruppi di soluzioni

- **Open Loop**: Tentano di risolvere il problema con una buona progettazione
  - Quindi cercando che il problema non si debba mai presentare, quando il sistema è in funzione non si tocca nulla.
  - Es: Decidendo quando accettare nuovo traffico, decidendo quando buttare i pacchetti e quali, prendendo decisioni di scheduling in varie parti della rete
- **Closed Loop**: Sono basati sul concetto di feedback
  - Monitorare il sistema per vedere dove e quando c'è congestione
  - Passare le informazioni dove si possono prendere provvedimenti
  - Modificare il funzionamento del sistema per correggere il problema

Classificazione

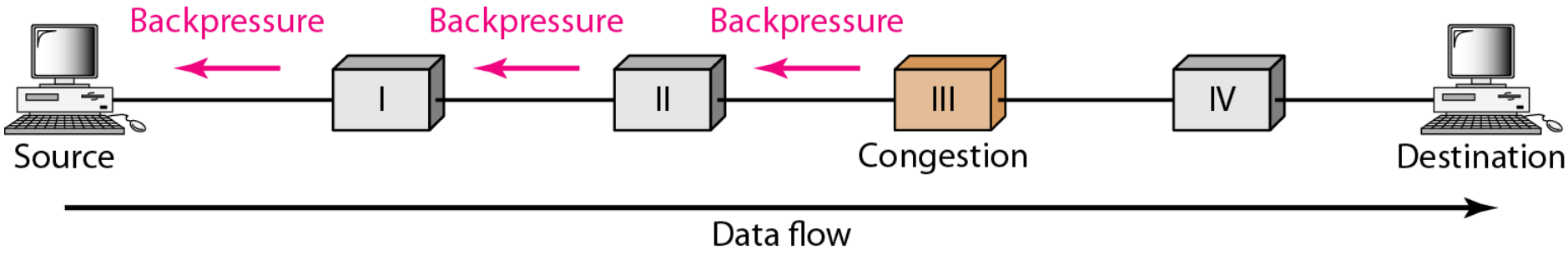


Closed Loop

- Cercano di eliminare la congestione dopo che si è verificata
- Back pressure policy – segnalazione all'indietro
- Choke packet – segnalazione al mittente
- Implicit Signaling
- Explicit Signaling

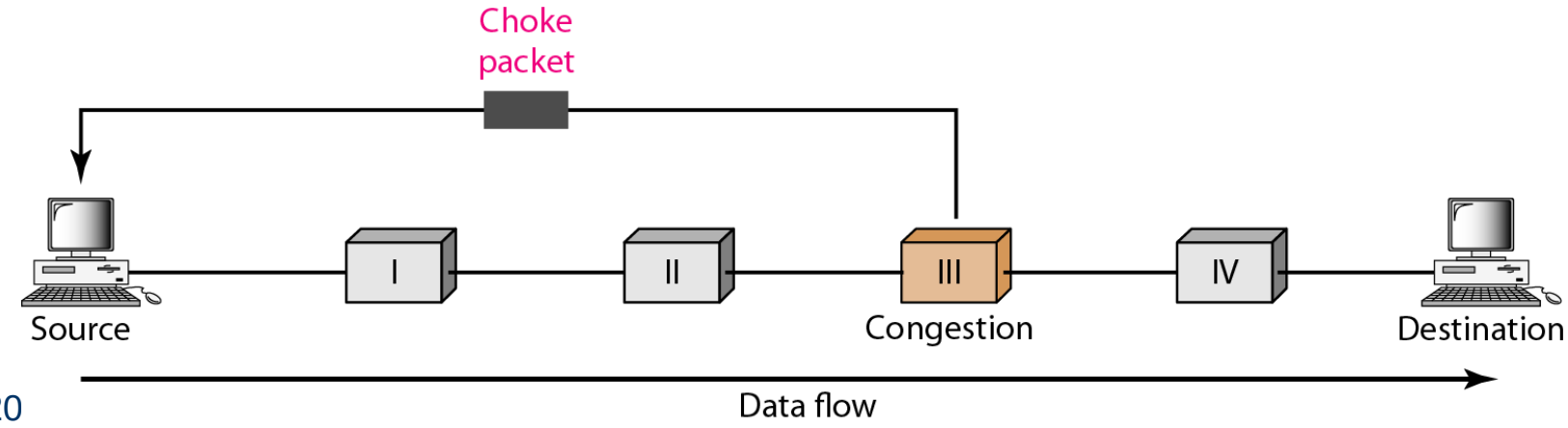
Back pressure

- Un nodo che sente la congestione smette di accettare i pacchetti dal nodo precedente
- Questo può causare congestione nello stesso e così via fino alla sorgente dei dati
- Può essere applicata solo se ho circuiti virtuali perché nelle reti datagram la nozione di router precedente potrebbe non avere senso
- In realtà nessuna rete usa questa rozza tecnica, ma è stata implementata nelle prime reti X.25



Segnalazione al mittente

- Il router congestionato informa direttamente il mittente
- Eventuali router intermedi non vengono notificati
- Es il protocollo ICMP, quando un router elimina dei pacchetti a livello network manda indietro al mittente un messaggio ICMP



Segnalazione implicita

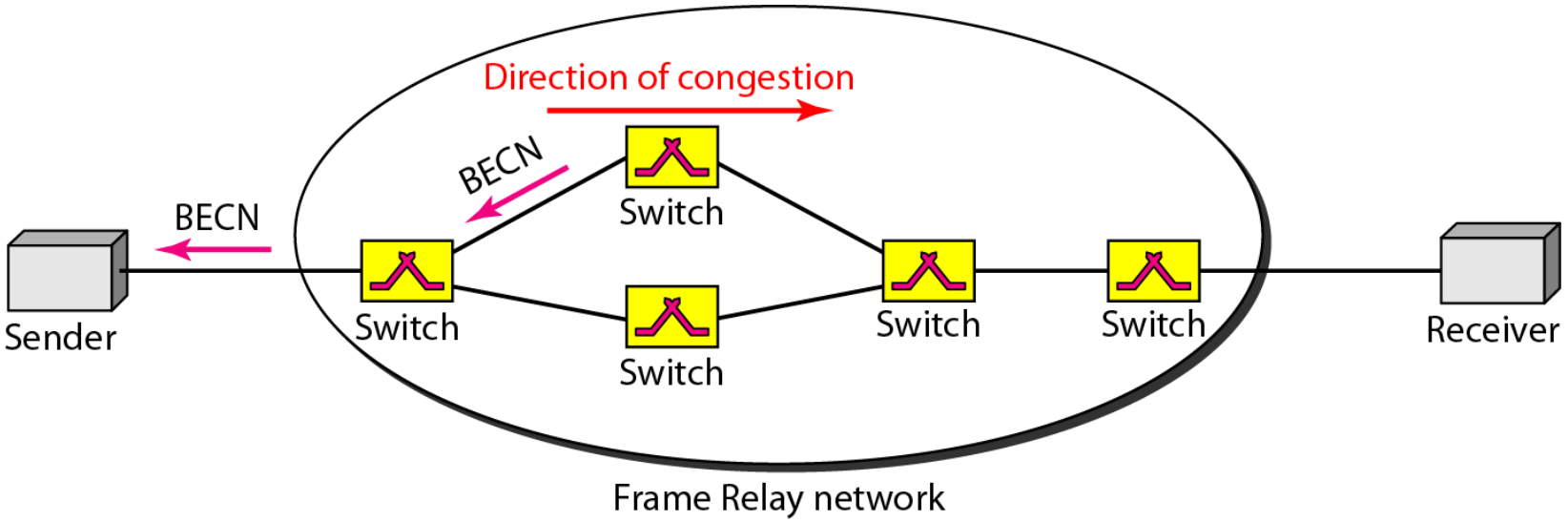
- Non c’è comunicazione tra nodo congestionato e il mittente.
- Il mittente si accorge che c’è congestione perché non riceve ACK per diversi pacchetti che ha mandato
- Anche il ritardo nell’arrivo di un ACK potrebbe essere indice di congestione
- Es: controllo della finestra di congestione TCP che vedremo nelle slides dedicate a TCP

Segnalazione Esplicita

- Con questa tecnica il nodo congestionato manda un segnale al mittente (backward signaling) o al destinatario (forward signaling)
- La **segnalazione all'indietro** è diversa dalla **back pressure** perché non viene mandato un pacchetto all'indietro ma semplicemente inserito un segnale dentro un pacchetto di dati che viaggia verso il mittente
- Vediamo l'esempio del Frame Relay

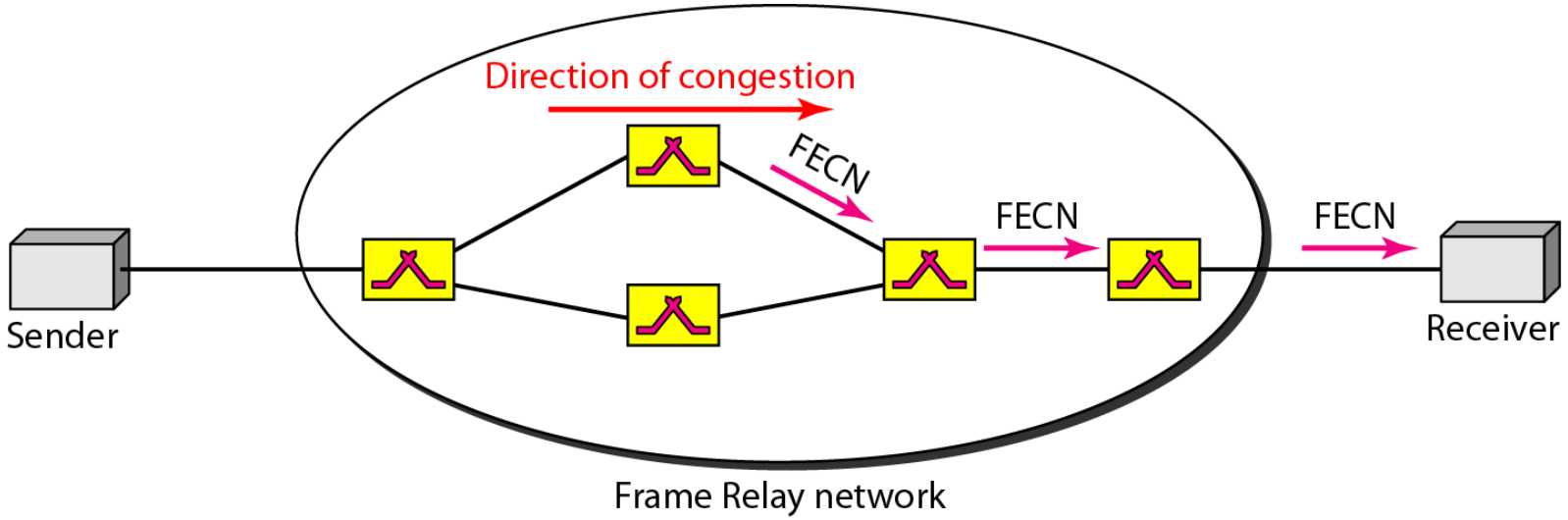
Frame Relay BECN

- **BECN** backward explicit congestion notification
  - Viene utilizzato nei frame che vanno verso il mittente, altrimenti se non ce ne sono uso una connessione predefinita con identificativo di circuito virtuale 1023, il mittente reagisce diminuendo il rate di spedizione

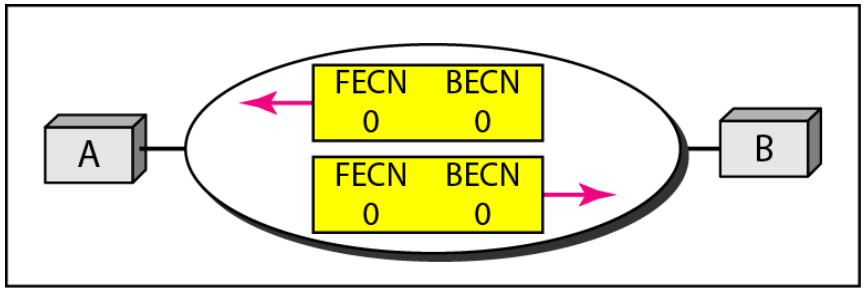


Frame Relay FECN

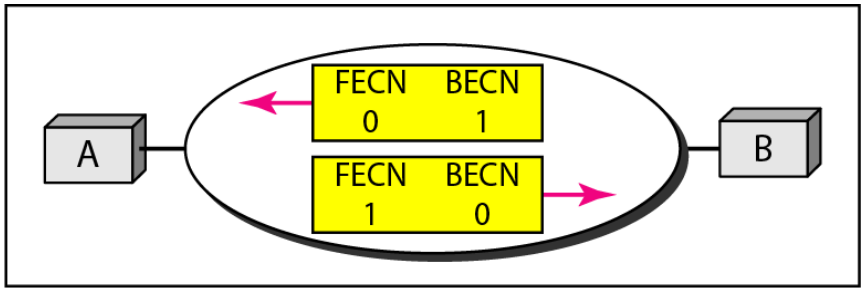
- **FECN** forward explicit congestion notification
  - Qui si possono usare direttamente i frame che viaggiano verso il destinatario. Potrebbe sembrare inutile perché non è colpa del destinatario, in realtà il destinatario potrebbe poi comunicare a livello di protocolli più alti e quindi utilizzare il controllo di flusso di quei protocolli per ritardare



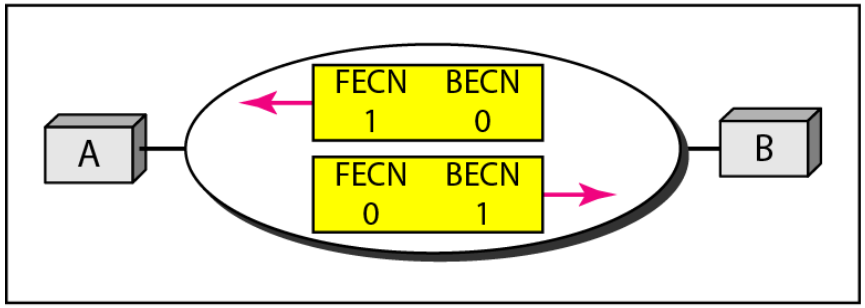
4 tipi di congestione FR



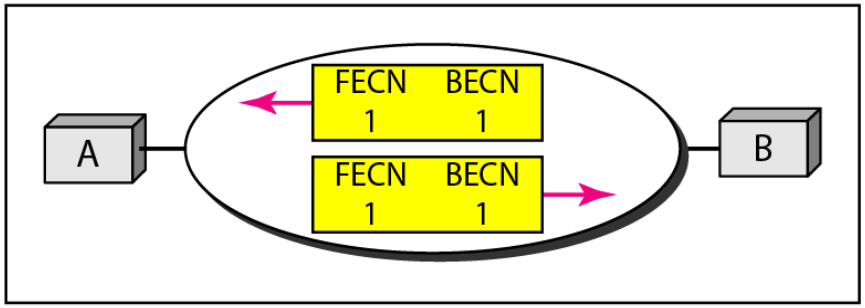
a. No congestion



b. Congestion in the direction A–B



c. Congestion in the direction B–A



d. Congestion in both directions

Monitoraggio

- Si possono usare diverse metriche per monitorare la rete
  - Percentuale di pacchetti che vengono scartati perché non c'è sufficiente spazio nei buffer
  - Lunghezza media delle code
  - Numero di pacchetti che vanno in time out e che devono essere ritrasmessi
  - Delay medio dei pacchetti e deviazione standard del delay
- Numeri in aumento indicano congestione in crescita

Informare chi di dovere

- Trasferire le informazioni da dove vengono rivelate al punto in cui qualcosa può essere fatto
  - Il modo più ovvio è che il router che rileva la congestione manda un pacchetto alla/e sorgente/i del traffico, annunciando il problema.
  - Questi pacchetti extra però aggiungono più carico alla rete proprio nel momento meno adatto

Alternative

- Un bit o un campo viene riservato in ogni pacchetto affinché il router lo possa riempire quando la congestione supera una certa soglia, avvertendo in questo modo i router vicini
- Oppure il router periodicamente invia dei pacchetti sonda che si informano esplicitamente della congestione. Queste informazioni poi sono usate per ruotare il traffico girando intorno alle zone più trafficate

Importanza del timing

- In tutti gli schemi di feedback c'è la speranza che la conoscenza della congestione induca gli host a prendere le opportune contromisure
- Ma per questo è importante che la scala temporale sia scelta con attenzione
  - Se ogni volta che arrivano due pacchetti di fila il router chiede uno stop e ogni volta che il router è idle da 20 μs chiede pacchetti, il sistema oscilla tra stop e go e non si stabilizza mai
  - Se invece il router aspetta sempre 30 minuti prima di mandare un feedback i meccanismi di controllo saranno sempre troppo lenti per produrre un risultato

Soluzione del problema

- La congestione è segno di carico (temporaneamente) maggiore delle risorse disponibili (di una parte del sistema)
- O aumento le risorse
  - Per esempio ho una linea punto a punto satura, ne tiro su un’altra in dial-up, oppure aumento la potenza di trasmissione di un link via satellite, o splitto il traffico su diverse route invece di usare solo quella ottimale
- O diminuisco il carico
  - Posso negare il servizio a qualche utente oppure degradare il servizio per qualche o tutti gli utenti oppure chiedere agli utenti di rischedulare le richieste in un modo più predicibile

Policies di prevenzione

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Policies di data link

- **Retrasmission policy**: il mittente non deve avere un timeout troppo corto per cui ritrasmette troppo spesso tutti i pacchetti per i quali non ha ricevuto ack
- **Out of order caching policy**: Se il ricevente scarta tutti i pacchetti fuori sequenza questi dovranno essere ritrasmessi, mentre potrebbe chiedere alla fine solo quelli che non sono arrivati e riordinare
- **Ack policy**: Fare l'ack di ogni pacchetto che arriva o approfittare del traffico nell'altro senso per fare piggybacking
- **Flow control policy**: Un flow control stretto, per esempio con finestre piccole, riduce il data rate e quindi combatte la congestione

Retrasmission

- **Retrasmission policy**:
  - La ritrasmissione è necessaria quando ci sono problemi con trasmissioni precedenti. La ritrasmissione però genera ulteriore traffico e quindi incrementa la congestione.
  - Il mittente non deve avere un timeout troppo corto per evitare di ritrasmettere pacchetti che sono solo in ritardo

Out of order caching

- **Out of order caching policy**:
  - Se il ricevente scarta tutti i pacchetti fuori sequenza nella sliding windows questi dovranno essere ritrasmessi
  - Invece potrebbe chiedere con il pacchetto di solo quelli che non sono arrivati e riordinarli poi quando arrivano
  - **Selective Ack**

Ack policies

- Se il destinatario non manda l'ACK per ogni pacchetto potrebbe rallentare il mittente e automaticamente prevenire la congestione
- Vari approcci
  - Posso spedire l'ACK solo se ho un pacchetto di dati da spedire per cui uso piggybacking
  - Oppure solo se è scaduto un timeout
  - Si potrebbe anche decidere di mandare un ACK solo se sono arrivati almeno N pacchetti
  - Oltre a rallentare la sorgente viene generato anche meno traffico aggiuntivo alla rete

Policies di network link

- **Scelta tra Virtual Circuit o Datagram:**
  - Molti algoritmi di “admission policy” funzionano solo in reti con circuiti virtuali
- **Packet queueing and service policy**:
  - i router hanno un coda per ogni linea di ingresso o per ogni linea di uscita o entrambi. E come le gestisco? Round Robin o con priorità?
- **Discard policy**:
  - Quali pacchetti devo droppare in caso non ci sia più posto. Es algoritmo RED per drop anticipato di pacchetti. In una trasmissione audio si possono eliminare pacchetti senza penalizzare eccessivamente la trasmissione

Policies di network link

- **Routing algorithm**: Evito congestione mandando il traffico su diverse linee se ho un buon algoritmo, mando altro traffico in una linea satura se ne ho uno cattivo
- **Packet lifetime management**: Quanto a lungo vive un pacchetto prima di essere scartato? Se vive troppo mi sta troppo in giro a intasare la rete, se vive troppo poco potrebbe andare in timeout prima di raggiungere la destinazione causando ritrasmissioni

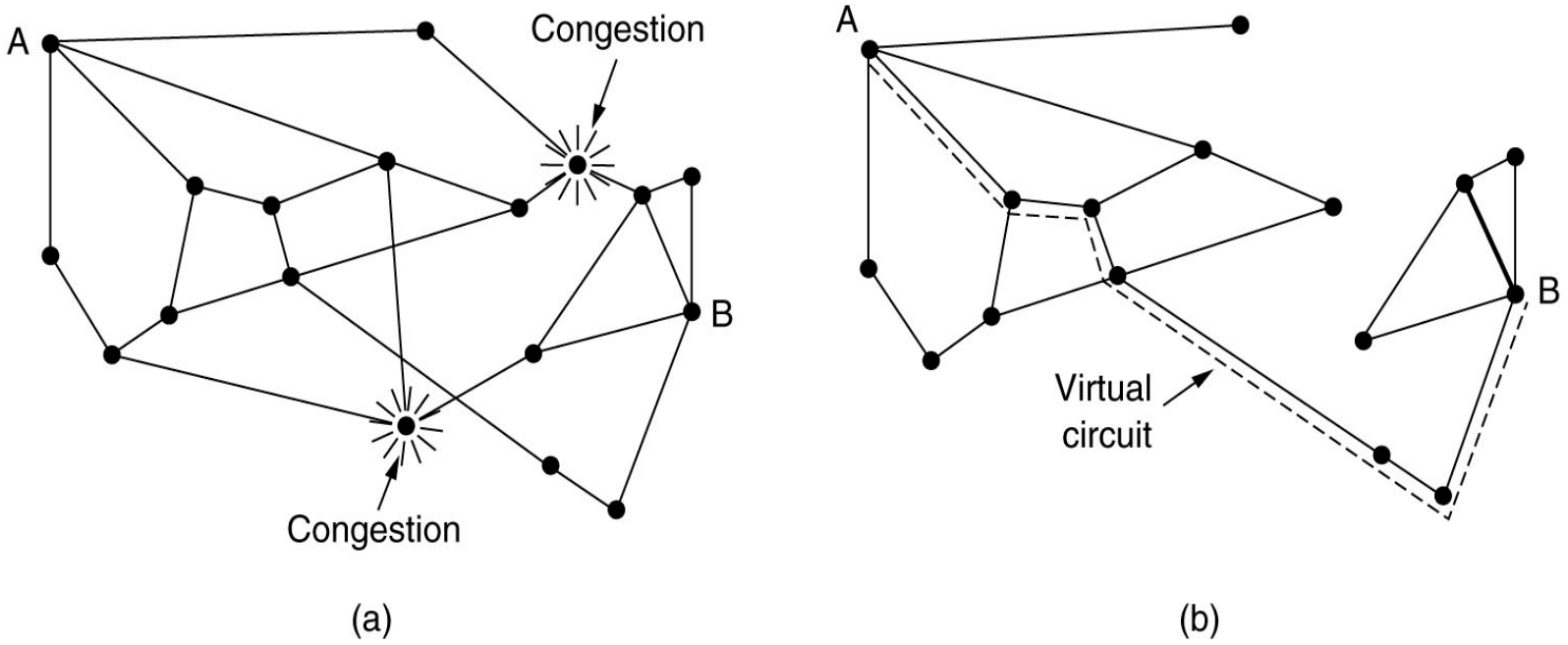
Policy di trasporto

- A livello di trasporto ho gli stessi problemi del data link ma in più ho maggiori problemi nel determinare i timeout, visto che il tempo di transito nella rete è meno prevedibile del transito di un filo tra due router
- Se il timeout è troppo corto vengono mandati pacchetti extra non necessari
- Se è troppo lungo la congestione viene ridotta ma peggiora il tempo di risposta quando un pacchetto viene perso

Congestione con VC

- Controllo di congestione con reti a Virtual Circuit:
  - **Admission Control.** Quando vedo segni di congestione non creo nuovi circuiti virtuali. Rude ma semplice. Se voglio telefonare e uno switch è sovraccarico non mi da un tono di libero
  - Creo nuovi circuiti non usando le aree congestionate

Evito le congestioni



- Nella figura (a) ci sono due punti di congestione
- Mi ridisegno la rete eliminando quei link e su (b) mi creo il mio circuito virtuale

Admission Policy

- Quando creo il circuito negozio le condizioni tra l'host e la subnet
  - Volume e forma del traffico (costante, bursty), QoS
  - La rete quindi prenota le risorse necessarie lungo il percorso quando il VC viene stabilito (tabelle e spazio buffer nei router, banda nelle linee)
  - La prenotazione si può fare di routine oppure solo quando la rete è congestionata
  - Lo svantaggio di farla sempre è lo spreco di risorse. Per esempio se ho 6 VC da 1 Mbps che passano per un link fisico a 6 Mbps, la linea è dichiarata piena anche se raramente tutti i 6 circuiti trasmetteranno a piena capacità allo stesso momento

Warning bit

- DECNET oppure Frame Relay segnalano lo stato di allerta settando un **bit** nell'header del pacchetto
- Quando il pacchetto arriva a destinazione a livello di trasporto viene copiato questo **bit** nel frame di **ack** verso la sorgente dei pacchetti, la quale riduce il traffico
- Finché il router sta in allarme, continua a tenere settato il bit e quindi la sorgente continua a ricevere gli **ack** con il bit di allarme e continua a ridurre il rate di trasmissione
- Quando invece arrivano pochi **ack** con il bit settato aumento il rate di trasmissione
- Dal momento che tutti i router nel path possono settare il bit, il traffico aumento solo se nessun router ha problemi di carico

Choke packet

- In questo caso il router manda direttamente un pacchetto (**choke packet**) all'host mittente usando l'indirizzo trovato nel pacchetto dati e setta un bit nel pacchetto dati, in questo modo questo non genererà altri **choke packet** lungo il cammino
- Quando la sorgente vede questo **choke packet** riduce il traffico di qualche percento.
- Altri pacchetti sono in viaggio per quella destinazione e genereranno altri **choke packet** ma la sorgente per un certo periodo fisso li deve ignorare
- Passato il periodo fisso la sorgente sente se arrivano altri **choke** per quella destinazione. Se ne arriva uno la linea è ancora congestionata e quindi riduce il flusso e si rimette ad aspettare il periodo fisso
- Se invece non arrivano **choke packet** la sorgente aumenta il flusso

Come ridurre il flusso?

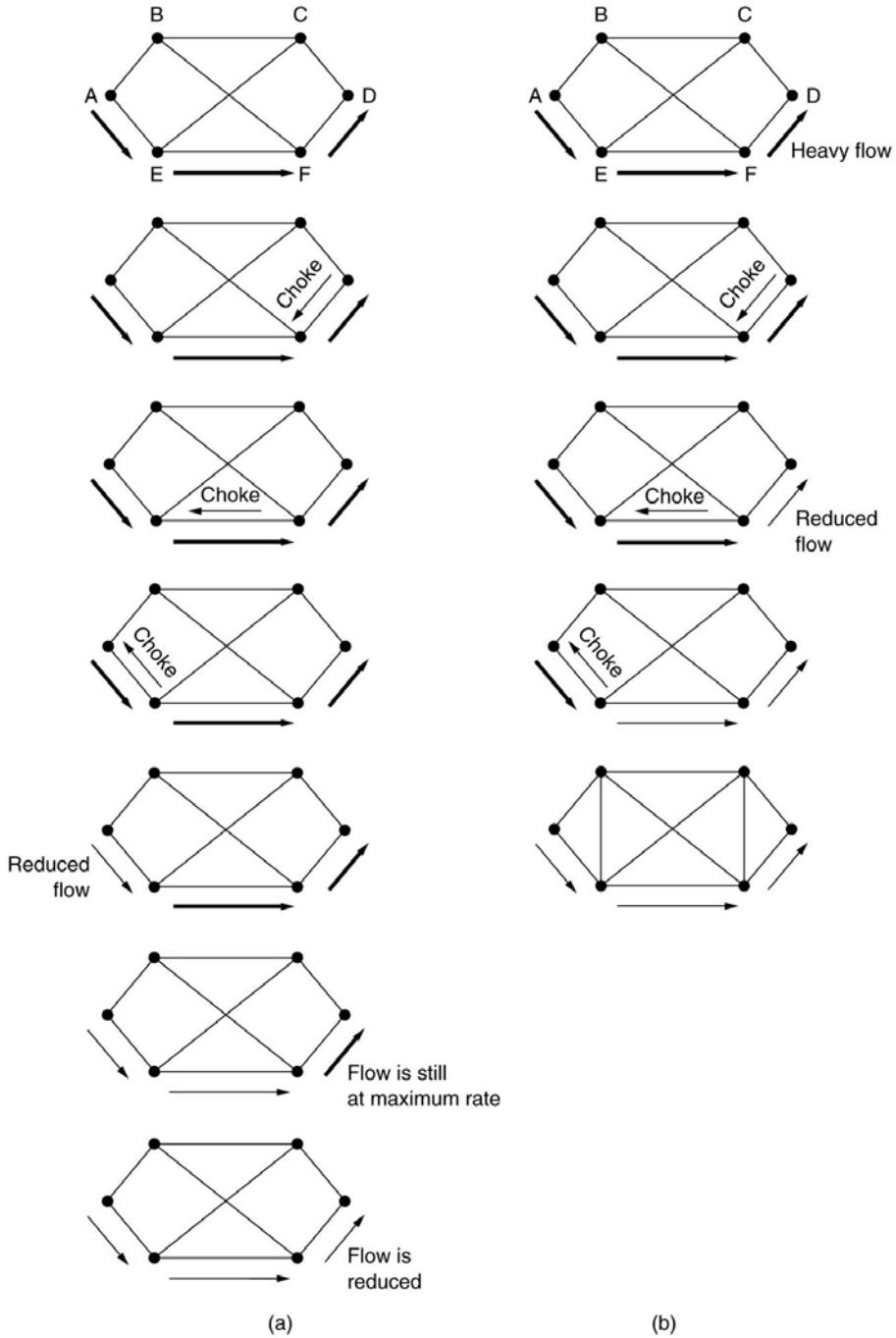
- Una sorgente di traffico riduce il rate di spedizione cambiando alcuni parametri, per esempio la window size
- Per esempio il primo choke causa una riduzione al 50% del rate precedente, il prossimo al 25% e via così
- Gli incrementi invece sono fatti in piccoli incrementi per prevenire che avvengano subito nuove congestioni
- Una variante di questo algoritmo prevede che ci siano diverse soglie, una di pre-allarme, un allarme medio e uno grave o un ultimatum
- Un'altra variante prevede che si misurino la lunghezza della coda o l'utilizzazione del buffer come trigger

Hop by hop choke packet

- Ad alte velocità o lunghe distanze il **choke packet** non funziona bene.
  - Es sorgente a San Francisco, dest. a New York, link a 155 Mbps.
  - Un choke packet ci mette 30 ms a tornare da NY a SF, nel frattempo sono partiti da SF altri 4.6 Mbit
  - Anche se SF immediatamente blocca ci sono questi 4.6 Mbit in transito che devono essere digeriti
  - Il router che sente la congestione avverte con un choke packet il precedente che quindi comincia rallenta, comincia ad accodare pacchetti.
  - Questo da un po' di respiro al router congestionato ma mette nei guai il router precedente, almeno fino a quando il choke packet non raggiunge il router ancora precedente
  - Alla fine il choke packet arriva alla sorgente e il flusso rallenta veramente

Choke

- L’effetto del hop to hop è quindi di fornire un miglioramento immediato nel punto di congestione al prezzo di usare più spazio di buffer nei router a monte



Load shedding

- Se nessuno di questi metodi funziona il router ricorre alle maniere forti
  - **Load shedding**, modo elegante per dire che droppano i pacchetti che non riescono a gestire
  - Termine che viene dall'usanza delle compagnie elettriche di togliere la corrente ad alcune aree per evitare lo shutdown di tutta la rete
  - Soluzione rozza che ovviamente funziona. Butto via a caso i pacchetti in eccesso
  - **ma i pacchetti non sono tutti uguali nella realtà…**

Wine and Milk

- Se sto facendo un file transfer i pacchetti vecchi valgono più di quelli nuovi
  - Esempio se droppo il pacchetto numero 6 e lascio passare da 7 a 10 avrò un buco che potrebbe costringere il ricevente a richiedere tutti i pacchetti da 6 a 10 (se il ricevente scarta tutti i pacchetti fuori sequenza)
  - Quindi se ho 12 pacchetti e droppo il 6 poi dovrò ritrasmettere tutti quelli da 6 a 12
  - Invece se droppo il 10 dovrò ritrasmettere solo da 10 a 12
- Invece con traffico multimediale è vero il contrario, un pacchetto nuovo “vale” di più di uno vecchio
- La prima policy (vecchio meglio) di nuovo si chiama **wine** mentre la seconda si chiama **milk**

Se il mittente coopera

- In alcune applicazioni alcuni pacchetti sono più importanti di altri
  - In traffico video compresso vengono trasmessi frame interi e poi le differenze tra il nuovo frame e il precedente
  - Il frame intero quindi è più importante, se va perso è inutile che io riceva bene tutti i delta
- Per implementare una policy di dropping intelligente le applicazioni devono marcare i pacchetti così da permettere classi di priorità
  - Allora i router in caso di congestione droppano per primi i pacchetti a bassa priorità

Incentivi

- Ma i miei pacchetti sono TUTTI importanti!
  - Se non c'è un incentivo tutti marcherebbero i propri pacchetti come molto importanti
  - Che incentivo potrei mettere?
  - L'incentivo potrebbe essere in denaro, i pacchetti a bassa priorità costano meno
  - Oppure ad un processo viene concesso di mandare pacchetti ad alta priorità quando il carico è leggero ma quando il traffico aumenta questi sono scartati. Questo incoraggia gli utenti a non mandare pacchetti ad alta priorità

Incentivi

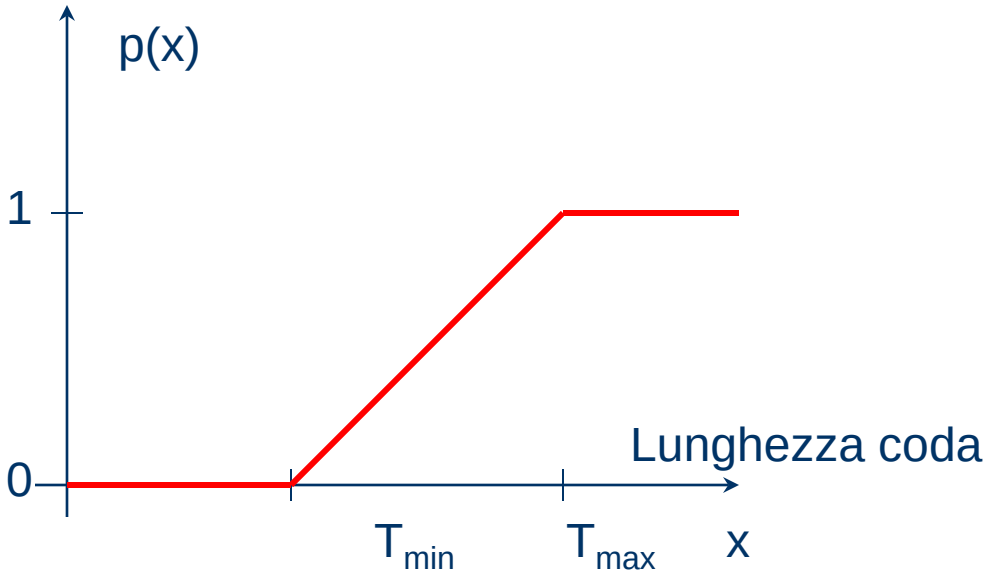
- Infine si può permettere all'host di eccedere alcuni limiti specificati negli accordi negoziati per il VC (per es. usare un bandwidth maggiore di quella permessa) ma con la condizione che tutto il traffico in eccesso è marcato come bassa priorità
- Questa è un'ottima strategia perché permette una efficiente uso di risorse scarse permettendo agli host di usarle se nessuno le sta usando ma senza concedere il diritto di usare le risorse quando i tempi diventano difficili

Random Early Detection

- Prima si riesce ad intervenire in caso di congestione e meglio è
  - Si potrebbe cominciare a scartare i pacchetti prima che i buffer siano veramente pieni
  - Un algoritmo molto popolare si chiama RED (Floyd Jacobson, 1993)
  - In alcuni protocolli di trasporto come TCP la rete rallenta anche se ci sono dei pacchetti persi. Infatti funzionano su reti wired molto affidabili e se si perdono pacchetti è perché i buffer sono troppo pieni e non per errori di trasmissione

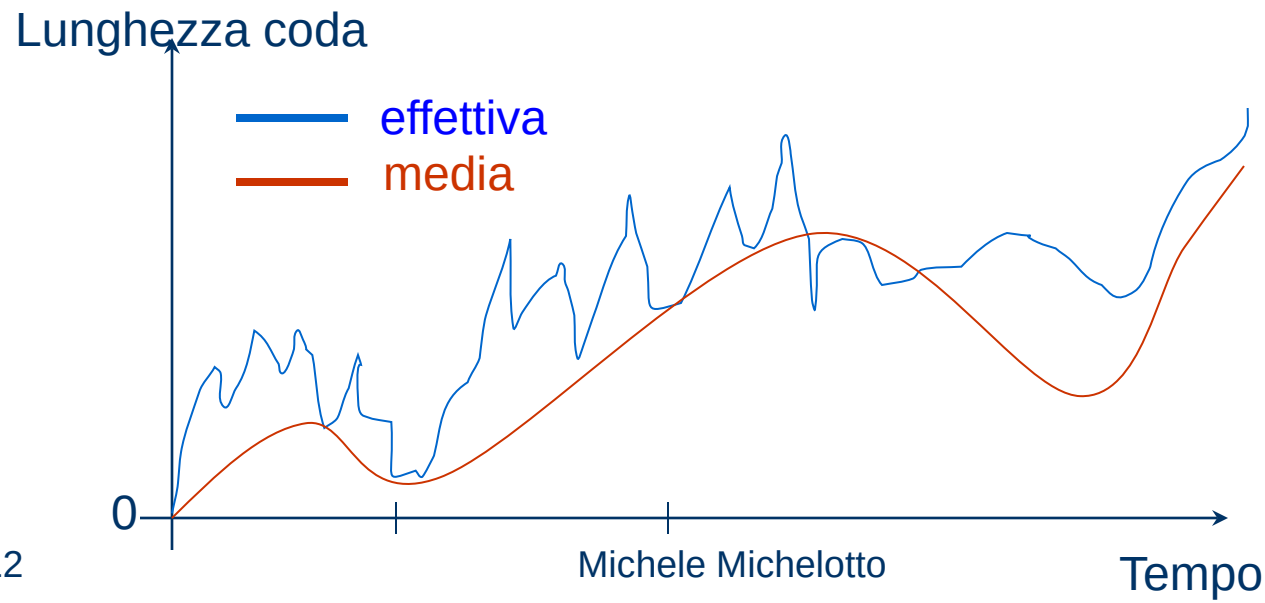
Algoritmo RED

- RED (Random Early Detection)
  - usa due soglie $T_{min} < T_{max} < T$ il numero massimo di pacchetti che possono stare nelle code di entrata
  - Quando arriva un pacchetto vedo se i pacchetti in coda sono $x < T_{min}$ allora lo lascio passare
  - Se $T_{min} < x < T_{max}$ allora viene eliminato con probabilità p compresa tra 0 e 1. Conviene mettere $T_{max}=T$ perché quando $x=T$ c'è congestione e tutti i pacchetti sono scartati
  - Una scelta semplice per p è che $p(T_{min})=0$ e $p(T_{max})=1$ e all'interno distribuzione lineare



Algoritmo RED

- Questo è molto sensato perché più aumenta la congestione e più probabilmente viene scartato il pacchetto
- In questo modo però in caso di brevi burst potrei scartare pacchetti che il router potrebe tranquillamente gestire
- Per evitare questo non si considera il numero di pacchetti effettivamente presenti in coda ma si mantiene la media (running average) y del numero di pacchetti in coda



Congestione e reti datagram

- In generale ogni router misura l'utilizzazione delle sue linee di input (RED), di output e altre risorse.
- Supponiamo che ad linea si possa associare una misura di carico *u* che va da 0.0 a 1.0
- Una stima del carico si può ricavare dall'utilizzazione presente *f* (0 oppure 1) e aggiornando $u_n = a u_{n-1} + (1-a) f$
- *a* determina quando velocemente il router dimentica la storia recente
  - con a vicino a 1 do molto peso alla storia passata
  - con a vicino a 0 peso molto la misura recente
- Quando *u* supera una certa soglia la linea va in allarme e ogni nuovo pacchetto viene controllato per vedere se deve andare in quella linea e se ci deve andare viene intrapresa qualche azione correttiva

Agire prima (early)

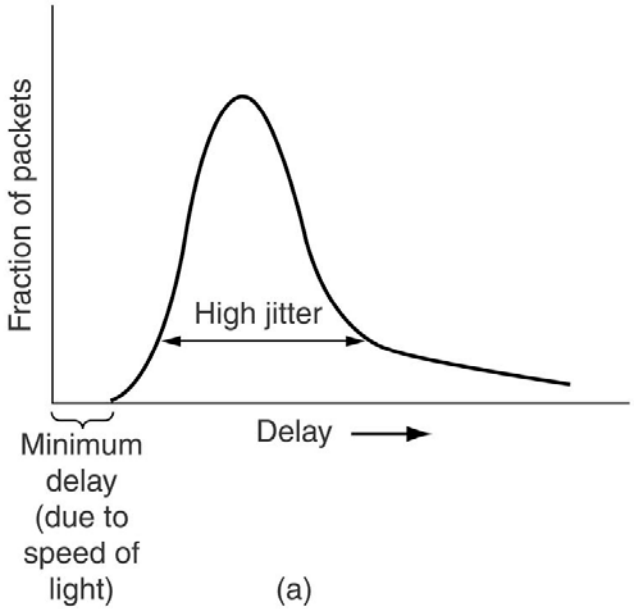
- I router si calcolano di continuo una media istantanea (running average) della lunghezza delle loro code
- Quando la media supera una certa soglia si dichiara la linea congestionata e si prendono provvedimenti
- Come fa il router ad avvertire la sorgente? Invece di mandare choke packet che causerebbero ulteriori traffico, semplicemente il router scarta il pacchetto e non dice nulla (come infatti fanno i router che implementano RED)

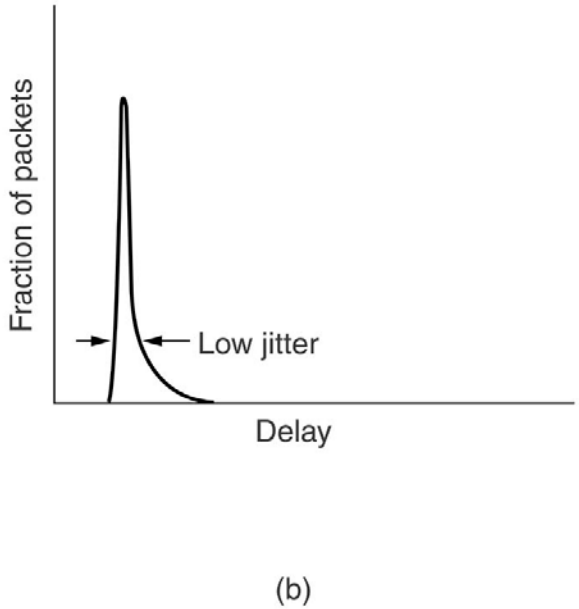
Agire prima

- La sorgente si accorge che che non è arrivato l'ack che si aspettava, ma sapendo che una perdita di pacchetto con ogni probabilità è dovuta da congestione, risponde abbassando il rate
- Questa forma implicita di feedback funziona solo quando la sorgente reagisce abbassando il rate in caso di perdita di pacchetti; in reti wireless in cui di solito le perdite sono dovute a rumore nel link aereo questo approccio non funziona

Jitter

- Quando ho traffico audio o video in **streaming**, non importa se i pacchetti ci mettono 20 ms o 30 ms per essere consegnati.
- Importa che il tempo di transito sia costante, cioè che la variazione standard dei tempi di arrivo detta **jitter** sia piccola





Controllo del jitter

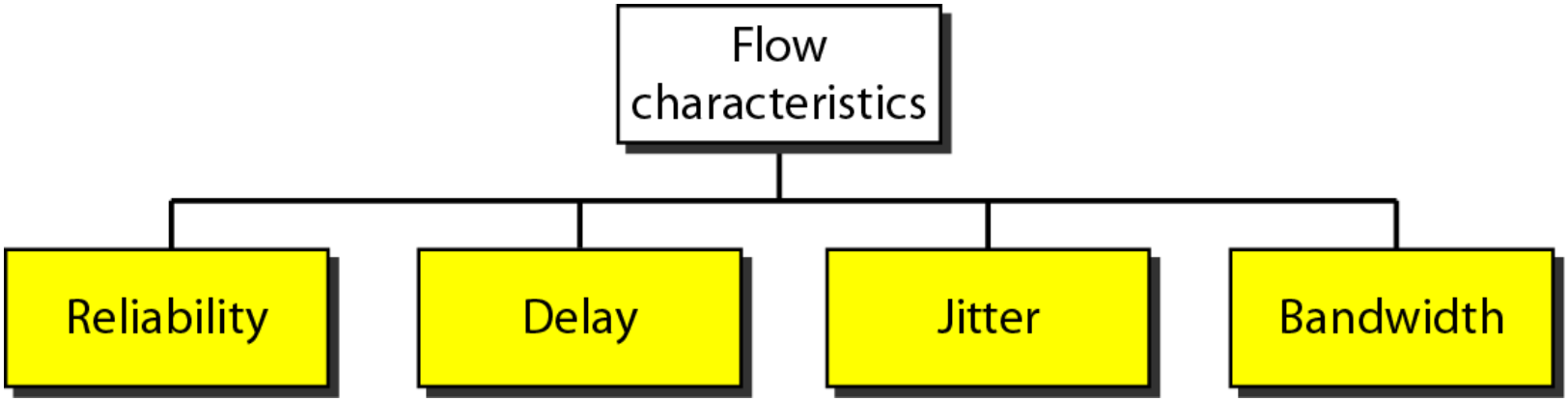
- Il jitter può essere limitato calcolando il tempo di transito atteso per ogni hop
  - Quando il pacchetto arriva ad un router, il router controlla se è in anticipo o in ritardo sulla tabella di marcia
  - Questa informazione viene messa nel pacchetto e aggiornata all'hop successivo
  - Se il pacchetto è in anticipo viene ritardato leggermente, mentre se è in ritardo viene mandato fuori con la massima celerità
  - Anzi se ho tanti pacchetti in attesa in una coda dovrei sempre prendere quello più in ritardo

Controllo del jitter

- In alcune applicazioni, come video on demand, il jitter viene eliminato con un grande buffer presso il ricevente, invece che usare la rete come buffer
- Quindi ho un **jitter** basso ma un **grande delay iniziale**
- Invece altre applicazioni interattive come VoIP o videoconferenza non posso usare questo trucco perché devo tenere non solo il **jitter** basso ma anche il **delay** di ogni scambio

Quality of Service

- Un flusso di dati è caratterizzato da quattro parametri principali
  - affidabilità, ritardo, varianza del ritardo e banda passante
  - reliability, delay, jitter e bandwidth
- Questi parametri determinano la Quality of Service (QoS) che il flusso richiede
- Diverse applicazioni pongono richieste diverse sui quattro parametri



Applicazioni e QoS

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

- Quanto sono stringenti i parametri di QoS per una certa applicazione?
- Poco (Low), Tanto (High) o una via di mezzo (Medium)

QoS e ATM

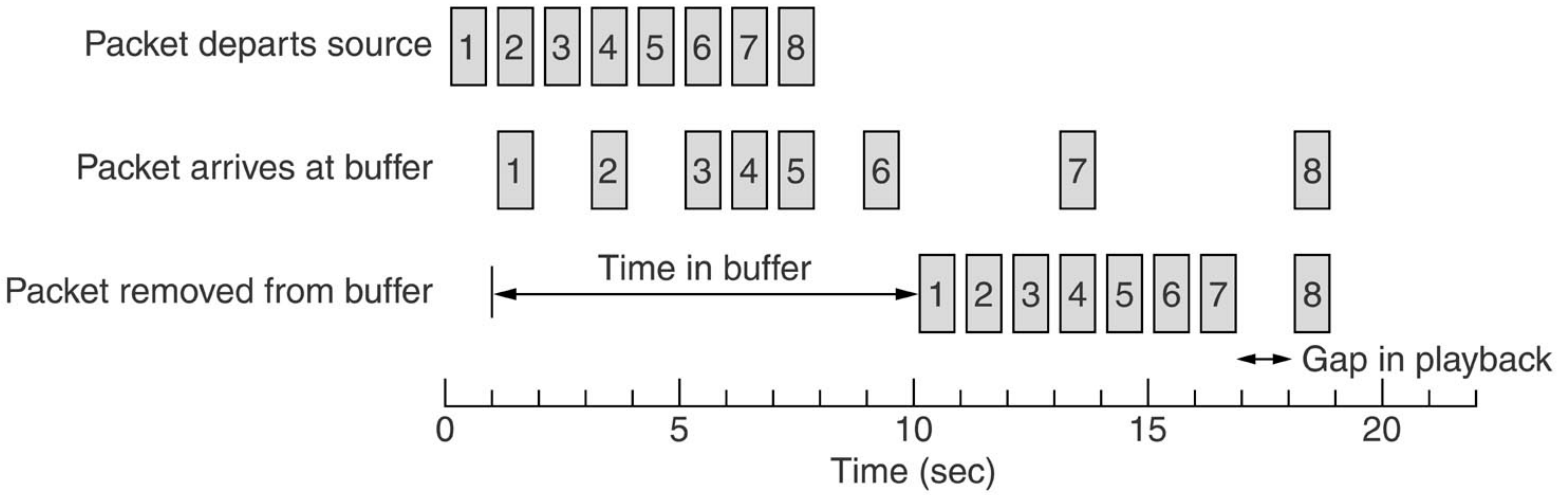
- Le reti ATM classificano i flussi in quattro categorie rispetto alle richieste di QoS
  1. Constant bit rate (telefonia): simula un filo con banda e delay costante
  2. Real time variable bit rate (videoconferenza): variable bit rate dal momento che la compressione video genera frame di dimensioni variabili
  3. Non real time variable bit rate (video su internet)
  4. Available bit rate (file transfer): per applicazioni come ftp e email non sensibili a delay e jitter

Sovradimensionamento

- **Overprovisioning**.
  - Reti con router, spazio buffer, e banda in eccesso in modo che il pacchetto non trovi mai traffico. Funziona bene ma è potenzialmente costoso
  - Con il tempo i progettisti capiscono quanto è sufficiente sovradimensionare la rete
- Quando le reti costano poco è la soluzione ideale.
  - Esempio in LAN si è passati in una decina di anni da reti shared a 10 Mps a reti switch a 100 Mbps o 1Gbps. La rete verso gli uffici in un edificio è ampiamente sovradimensionata
  - In WAN si è passati da link a 256kbps/2Mbps a reti con link a 1 Gbps / 10 Gbps o dark fibre.

Buffering

- Si possono bufferizzare i flussi alla destinazione. Questo non influenza bandwidth o reliability, aumenta il **delay** ma minimizza il **jitter**
- Tecnica da usare in audio o video on demand in cui il problema principale è proprio il **jitter**
- Come si vede dalla figura il buffering ha compattato tutti i pacchetti a parte il numero 8 per cui si sente una fastidiosa interruzione
- Tutti i siti web commerciali che fanno streaming audio o video hanno media players con un buffer di circa 10 secondi



Traffic shaping

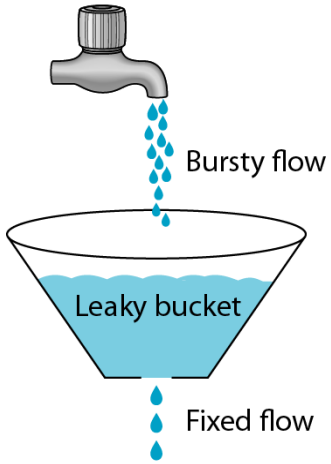
- Il buffering non va bene se i pacchetti non partono uniformemente, oppure se ho traffico interattivo come la videoconferenza
- **Traffic shaping** cerca di rendere il traffico omogeneo dal lato mittente, regolandone il flusso e la burstiness
- Quando si stabilisce la connessione l'utente e la rete si accordano su di un certo pattern di traffico (shape).
- A volte si dice che c'è un **S**ervice **L**evel **A**greement
- Finché l'utente sta nei parametri concordati, la rete promette di consegnare i pacchetti senza ritardi

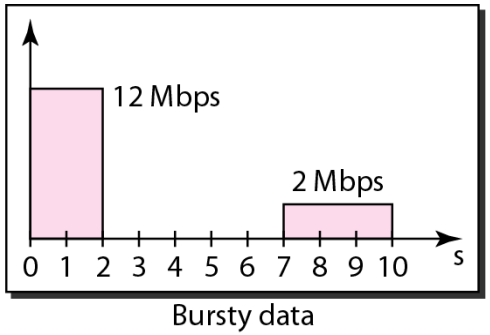
Leaky bucket

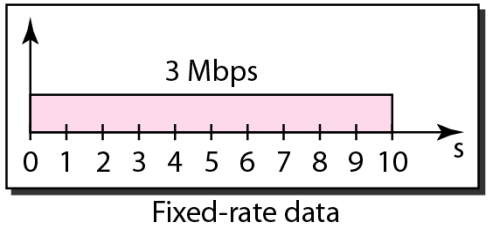
- Come un secchio con un buco sul fondo. Non importa quanta acqua entra, il flusso sul fondo è costante e va a zero quando il secchio è vuoto
- Ogni nodo è collegato alla rete da un'interfaccia con un secchio bucato, una coda interna finita. Se il pacchetto arriva quando la coda è piena viene droppato, altrimenti esce a rate costante
- Può essere implementato in hardware o dal sistema operativo
- Produce un flusso di pacchetti costante piallando i burst e riducendo le probabilità di congestione

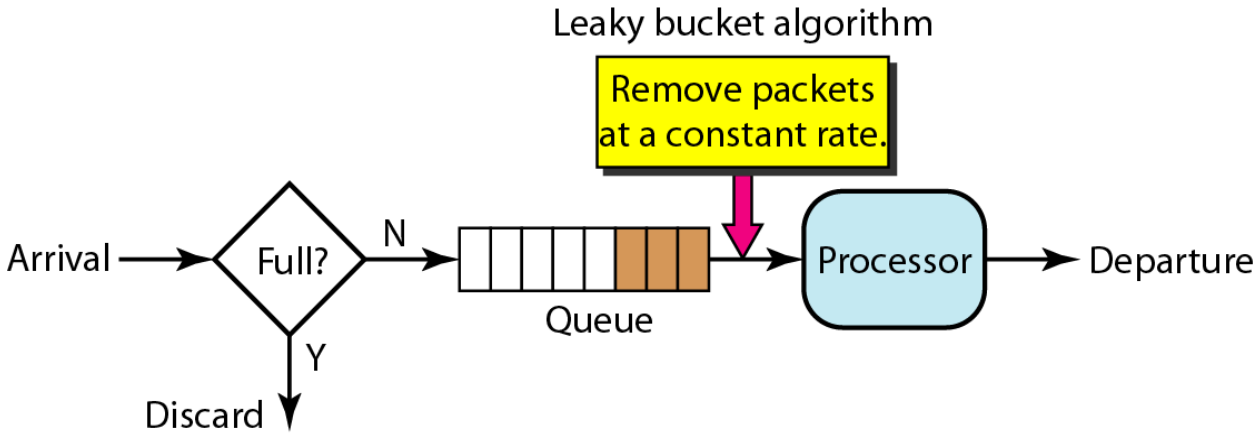
Secchio bucato

- I frame arrivano a burst ed entrano in una coda
- Un processo li fa uscire ad un ritmo costante
- Come un'imbuto









Frame variabili

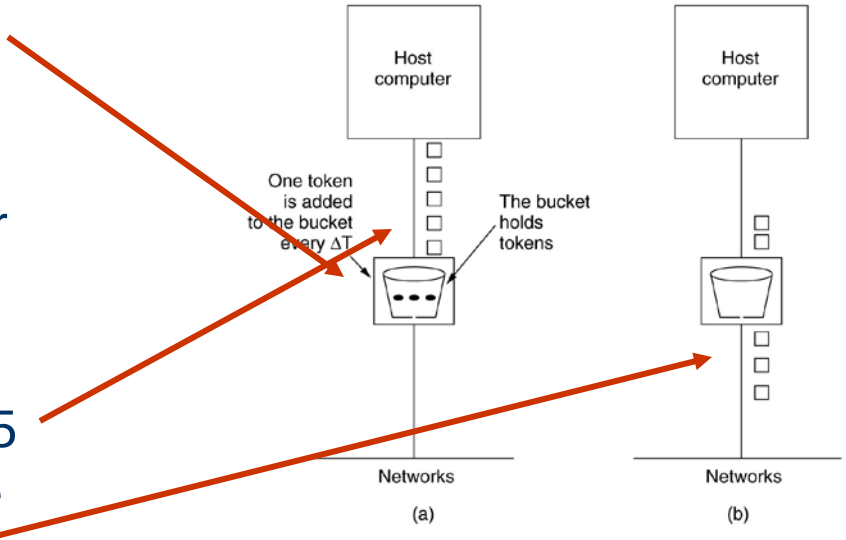
- **Leaky Bucket** funziona bene con pacchetti di lunghezza fissa, es celle ATM
- Se ho frame di lunghezza variabile è meglio estrarre un numero fisso di byte per ogni scatto piuttosto che un solo pacchetto
- Es se manda 1024 bytes per scatto, posso far passare un pacchetto da 1024, o due da 512 o 4 da 256. Se quello che rimane è troppo poco lo metto in attesa del prossimo pacchetto

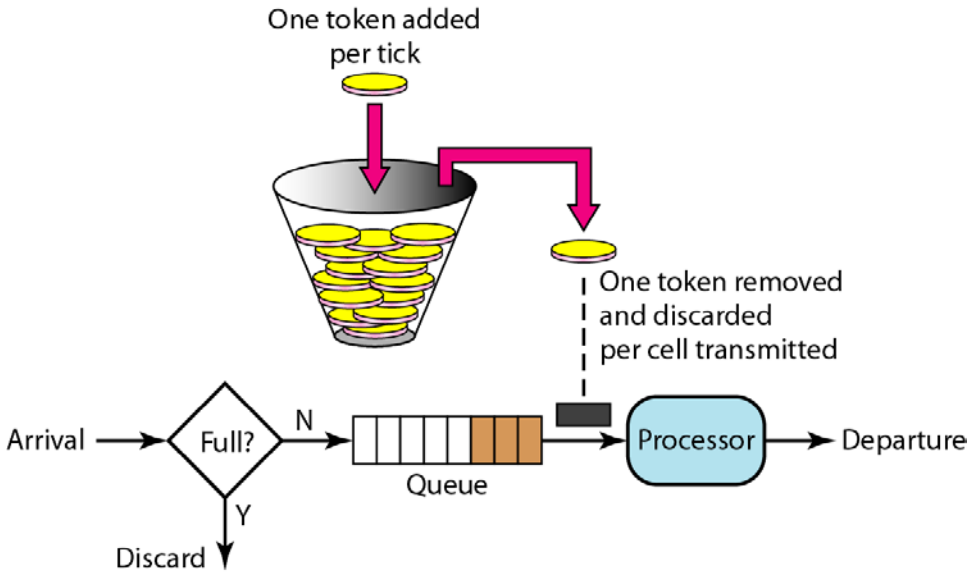
Token bucket

- Leaky bucket crea un pattern di uscita rigidamente piatto sul livello medio, per quanto il traffico sia a burst
- Alcune applicazioni preferiscono velocizzare l'uscita quando arriva un grosso burst
- Un algoritmo più adatto è allora il **Token Bucket**, in cui nel secchio si mettono dei token generati da un clock ogni ΔT secondi

Token Bucket

- Es Ho un bucket con 3 token. Per far passare un pacchetto devo catturare e un token per poi poterlo distruggere (spendere)
- Quindi se mi trovo con 5 pacchetti in ingresso ne escono solo 3.
- Glialtri 2 devono aspettare che venga generato due token quindi due volte ΔT



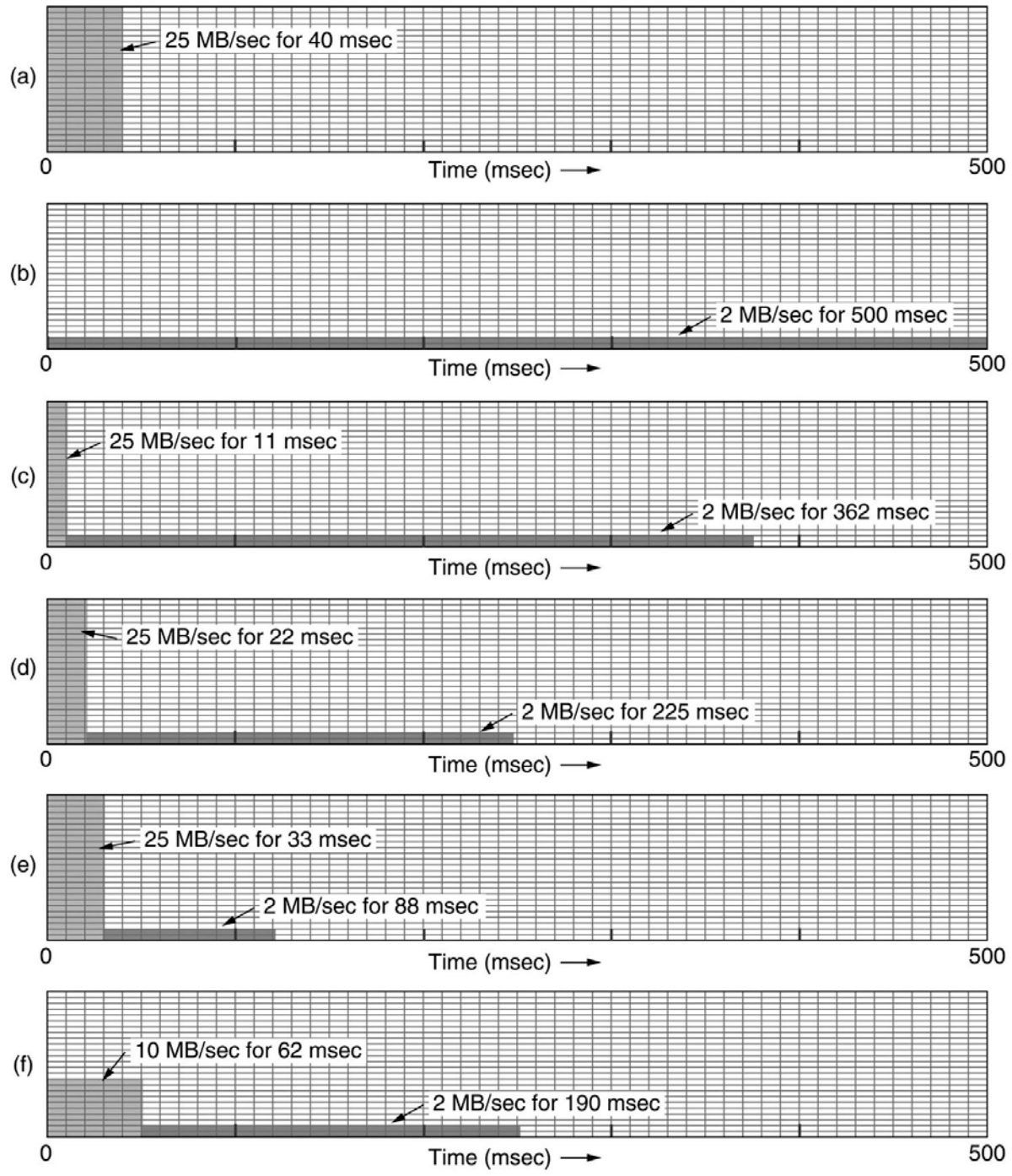


Recupero il tempo idle

- Il leaky bucket non mi permetteva di recuperare il tempo in cui nessun traffico arriva
- Invece con il token bucket posso accumulare token fino a un massimo N, quindi permetto burst fino a N pacchetti dopo i quali vado a rate costante pari al rate di creazione dei token
- Ho quindi una risposta più veloce ai burst, ma al contempo i burst troppo lunghi vengono smorzati
- Anche qui posso avere una variante in cui un token non rappresenta il diritto di trasmettere un pacchetto ma un certo numero M di bytes

I due insieme

- Un potenziale problema del token bucket è che permette comunque dei grandi burst, anche se li accorcia
- A volte vorrei solo tagliare il rate di picco ma senza scendere ad un valore medio bassissimo
- Allora inserisco dietro il token bucket un leaky bucket con una banda più alta del rate di creazione dei token ma più bassa del rate massimo della rete



- (a) Traffico in ingresso
- (b) output di un **leaky bucket** da 2MB/s
- (c) output di **token bucket** con creation rate di 2MB/s e con capacità iniziale di 250 KB
- (d) … di 500 KB
- (e) … di 750 KB
- (f) come(d) ma aggiungo in coda un **leaky bucket** da 10MB/s per spianare il picco da 25MB/s

Resource Reservation

- Saper plasmare il profilo del traffico è un buon inizio per garantire QoS, tuttavia per poterne approfittare bisognerebbe essere capaci di mandare tutti i pacchetti nella stessa route
- Se ogni pacchetto sceglie la sua strada è difficile garantire qualcosa
- Sarebbe utile poter stabilire un VC da sorgente a destinazione in modo che tutti i pacchetti del flusso seguano lo stessa route
- A questo punto posso prenotare l'uso di alcune risorse lunga il percorso per essere sicuro di avere ovunque la capacità minima necessaria

Risorse

- Bandwidth. Se ogni flow ha bisogno di 1 Mbps e ho un link a 2 Mbps non posso mandarci 3 flussi. Posso accettare due flow ma non posso fare overbooking su di una linea di uscita
- Buffer space. Quando il pacchetto arriva viene depositato in una scheda di interfaccia, poi il software del router lo copia in RAM e lo accoda in un buffer per essere trasmesso in una linea di uscita. Per avere buona QoS deve prenotare alcuni buffer per ogni specifico flow in modo che questo non debba competere con gli altri flussi per accedere ai buffer

Risorse

- Cicli di CPU: Il router ha bisogno di tempo di CPU per processare un pacchetto, quindi ne può processare solo un certo numero al secondo
- Se ci metto 1μs per un pacchetto sembrebbe che il router sia in grado di processare 1 milione di pacchetti per secondo
- Invece anche con rate inferiori possono crearsi delle code e accumularsi ritardi

Calcolo delay

- I pacchetti arrivano a caso con rate medio di λ packet/sec
- Anche il tempo di cpu ha distribuzione casuale con media μ packet/sec
- Se queste distribuzione casuali sono Poissoniane si può dimostrare che il ritardo medio di un pacchetto è

$$T = \frac{1}{\mu} \times \frac{1}{1 - \lambda / \mu} = \frac{1}{\mu} \times \frac{1}{1 - \rho}$$

- Dove $\rho = \lambda/\mu$ è il carico della cpu
- Il primo termine $1/\mu$ è il tempo di servizio in assenza di competizione. Il secondo è il rallentamento causato dalla competizione con gli altri pacchetti

Esempio numerico

- λ = 950000 pacchetti/sec
- μ = 1000000 pacchetti/sec
- Quindi ρ = 0.95 per cui T =20 μsec invece che 1 μs che comprende sia il tempo di accodamento che il tempo di gestione da parte del router
- Se ci sono diciamo 30 router lungo il percorso del flusso, solo per l'accodamento ho 600 μs di delay (poi ci sarebbe da aggiungere il delay dovuto al tempo di propagazione nel link)

Admission Control

- Ora sappiamo modellare la forma del flusso e come costringerlo ad andare in un unico percorso
- Quando questo flusso viene proposto ad un router, questo deve decidere, conoscendo le proprie capacità e quanti flussi ha già promesso di gestire
- Non è una decisione facile perché sebbene molte applicazioni conoscono le loro necessità di bandwidth ma poche sanno di quanti buffer o cicli di cpu necessitano, quindi devo trovare un altro modo per descrivere i flussi
- Inoltre alcune applicazioni sopportano meglio di altre il fallimento occasionale di una deadline

Adattamento dei parametri

- Infine alcune applicazioni possono adattarsi a cambiare parametri mentre altre no. Per esempio un film a 30 frame/sec si potrebbe vedere anche a 20 frame/sec o a dimensioni ridotte se non c'è abbastanza bandwidth
- Visto che alla contrattazione partecipano diverse entità (mittente, destinatario e tutti i router intermedi) i flussi devono essere descritti accuratamente dai parametri
- Di solito il mittente produce un insieme di parametri che vorrebbe usare. Ogni router le esamina ed eventualmente le modifica ma solo **in ribasso**. Quando le specifiche del flusso arrivano al destinatario i parametri possono considerarsi fissati

Esempio di specifiche

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

- Esempio basato su RFC 2210 e 2211 basato su cinque parametri

1. Byte/sec messi nel bucket, quindi il massimo rate a cui il mittente può trasmettere in media
2. Dimensioni del bucket
3. Massimo rate di trasmissione tollerato, anche per brevi intervalli
4. Dimensione minima del pacchetto
5. Dimensione massima

Packet size

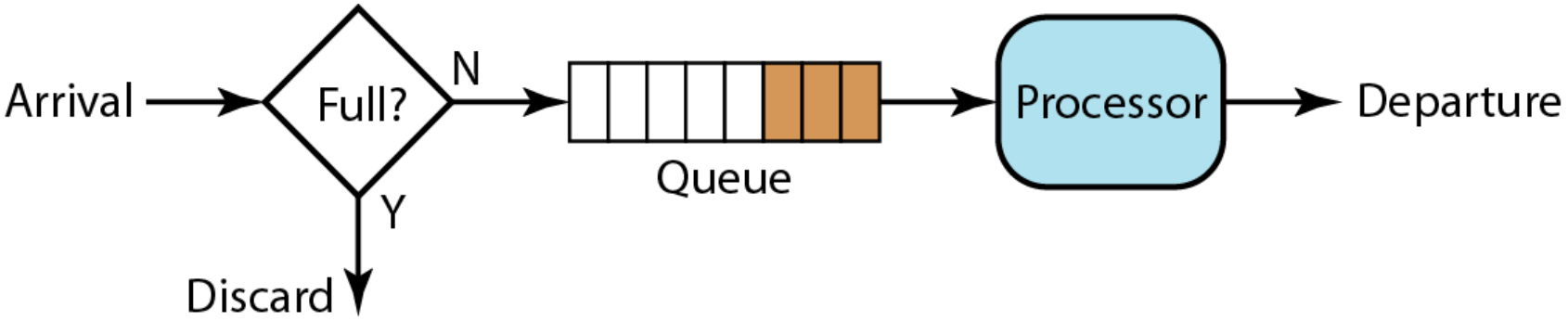
- Le dimensioni del pacchetto comprendono anche gli header TCP e IP
- Sono importanti perché l'elaborazione ha dei tempi fissi non importa quanto sia corto il pacchetto
- Un router potrebbe gestire 10000 pacchetti da 1 KB ciascuno al secondo ma magari non gestire 100000 pacchetti da 50 byte ciascuno al secondo sebbene il data rate sia inferiore
- Invece le dimensioni massime sono importanti per limitazioni interne alle reti, per esempio se il cammino attraversa Ethernet il pacchetto non deve eccedere i 1500 byte

Specifiche vs risorse

- Come fa il router a convertire delle specifiche di flusso in risorse da riservare?
- La conversione dipende da ogni specifica implementazione
- Es abbiamo un router in grado di gestire 100000 packet/sec.
  - Se gli propongono un flusso da 1 MB/sec con min 512 Byte, allora sa che deve gestire fino a 2048 packet/sec da quel flusso
  - Quindi deve riservare almeno 2% di CPU, meglio qualcosa in più per evitare ritardi di accodamento
  - Se la policy del router è di non allocare più del 50% di CPU (che implica un fattore due per il delay e se è gia al 49% deve respingere quel flusso)

Packet Scheduling

- Quando un router gestisce diversi flussi, c'è il pericolo che flusso molto esigente danneggi gli altri
- Se i pacchetti vengono elaborati in ordine di arrivo (FIFO) una sorgente aggressiva potrebbe impadronirsi delle risorse dei router nel transito riducendo la Qos per gli altri
- Per ovviare a questo problema sono stati pensati diversi algoritmi di packet scheduling

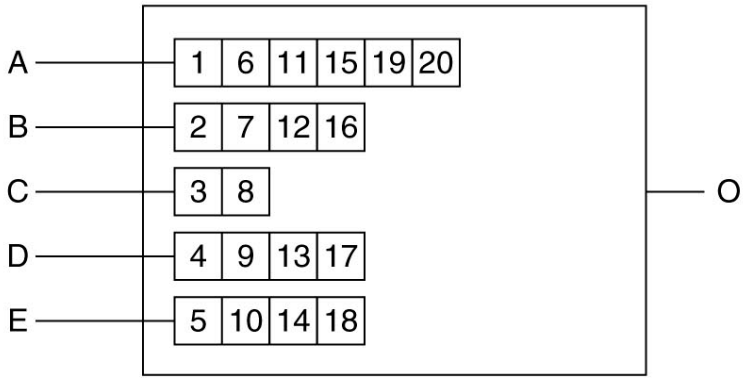


Fair Queueing

- I router hanno code separate per ogni flow per ogni linea di uscita
- Quando una linea diventa idle, il router passa alla successiva in round robin. In questo modo se N host sono in competizione per una certa linea di uscita, ognuno ha un pacchetto ogni N e mandando più pacchetti non ci si guadagna nulla
- In questo modo però viene data più banda a chi usa pacchetti grossi rispetto a chi usa pacchetti piccoli

Queueing per byte

- Un miglioramento consente di simulare un round robin **byte by byte** invece che **packet by packet**
- Il router fa uno scan per byte di ogni coda finché non trova dove finisce ogni pacchetto con una specie di clock virtuale, poi i pacchetti vengono ordinati per lunghezza e mandati in quell'ordine



(a)

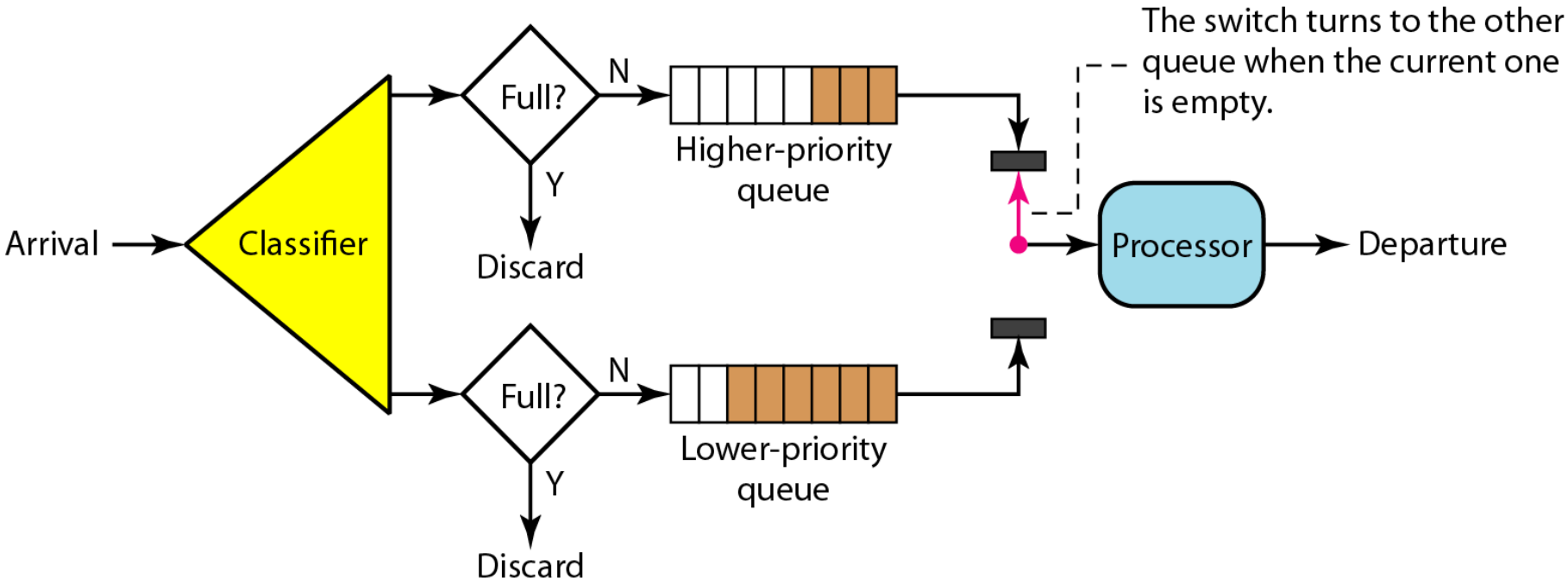
Packet	Finishing time
C	8
B	16
D	17
E	18
A	20

(b)

Priority Queueing

- L’algoritmo appena descritto assegna a tutti gli host la stessa priorità
- Ma noi potremmo volere dare più priorità diverse. Mandiamo traffico nella coda a bassa priorità solo se è vuota la coda ad alta priorità
- Ottimo per fornire QoS per la applicazioni ad alta priorità ma a prezzo di deteriorare la qualità delle altre applicazioni fino ad arrivare a ricevere nessun servizio se c’è un flusso continuo ad alta priorità

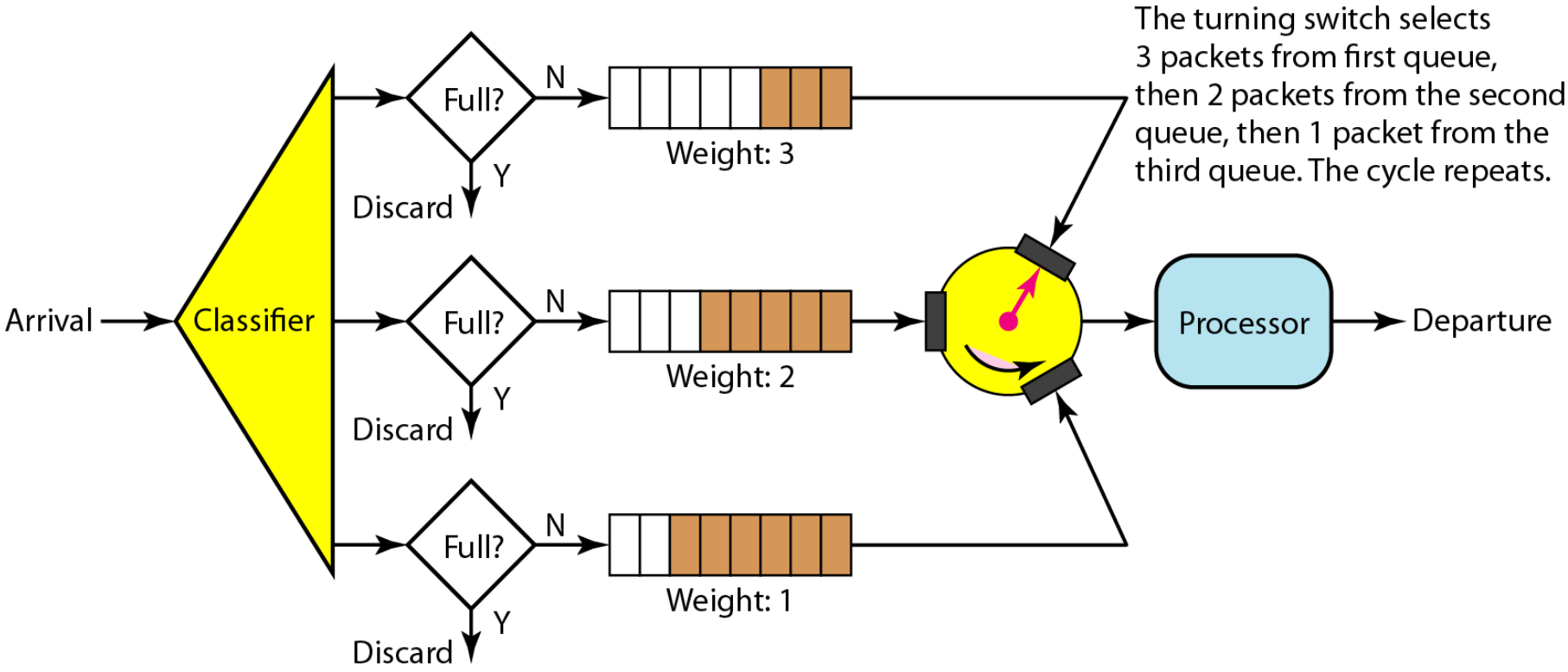
Alta e bassa priorità



Weigted Fair Queueing

- Un meccanismo migliore è quello delle code pesate per cui le code a bassa priorità ricevono comunque un servizio anche se ci sono dei pacchetti ad alta priorità
- Noi potremmo volere dare più priorità ad un video server che ad un file server
- In questo caso si possono mandare due bytes per ogni tick.
- Questo algoritmo molto usato si chiama **Weighted Fair Queuing**.
- Se metto pesi proporzionali al numero di flussi in uscita da una macchina, ogni processo prende la stessa bandwidth

Weighted fair queueing



IS o DS

- Due modelli che enfatizzano la QoS nel network layer anche se possono essere utilizzati a livello di trasporto
- **Integrated Services** e **Differentiated Services**

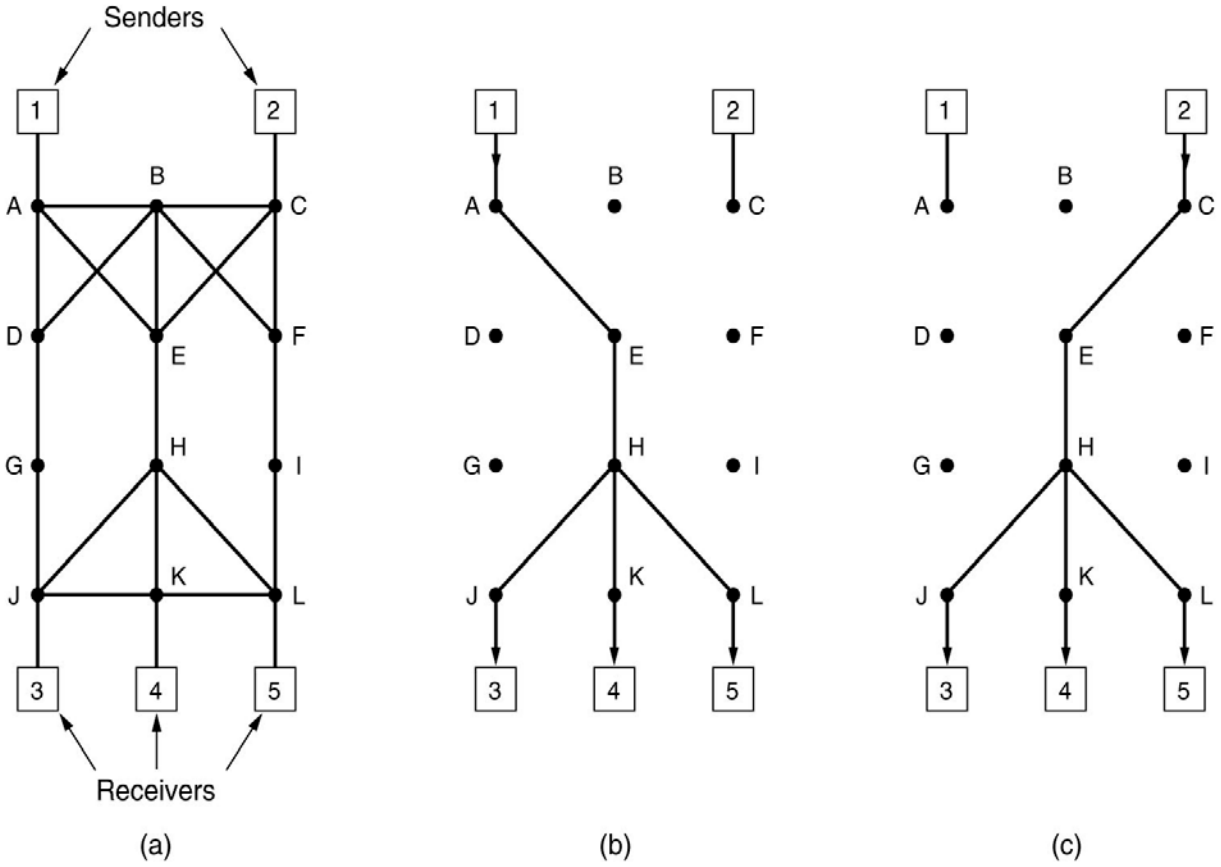
Integrated Services

- Nel periodo 95-97 ci fu uno sforzo di IETF per un'architettura per lo streaming multimediale (RFC da 2205 a 2210)
- A questo lavoro venne dato il nome generico di **algoritmi flow-based** o **integrated services**, sia per applicazioni unicast (un videoclip da un sito di news) che multicast (una stazione televisiva che trasmette via IP a diversi ricevitori in diversi posti)
- Vediamo il multicast di cui unicast è un caso particolare
- L'appartenenza ad un gruppo multicast può cambiare dinamicamente, per esempio quando uno spettatore cambia canale, per cui riservare la banda in anticipo non serve a nulla,

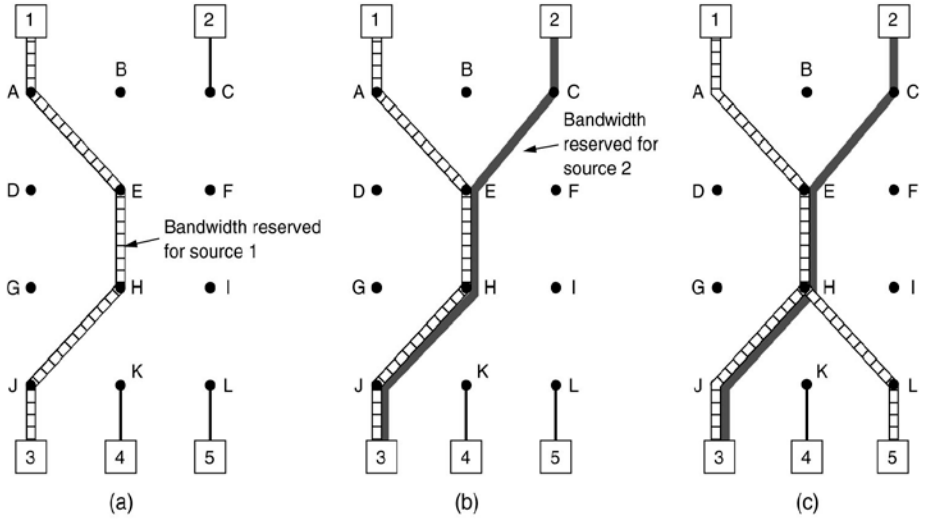
Resource reSerVation Protocol

- **RSVP** è il protocollo per fare le riservazioni, i dati vengono trasmessi con altri protocolli
  - Una specifica di flusso è costituita da un Rspec (specifica di risorse, memoria, bandwidth ecc) ed una Tspec (specifica di traffico, che forma avrà il traffico)
- Nella forma più semplice usa multicast routing con spanning tree. Ogni gruppo ha il suo indirizzo di gruppo che il mittente mette nei pacchetti che deve mandare al gruppo
- L'unica differenza da un normale multicasting sono alcune informazioni che vengono mandate, sempre in multicast, ai router lungo l'albero per mantenere in memoria certe strutture

Spanning tree

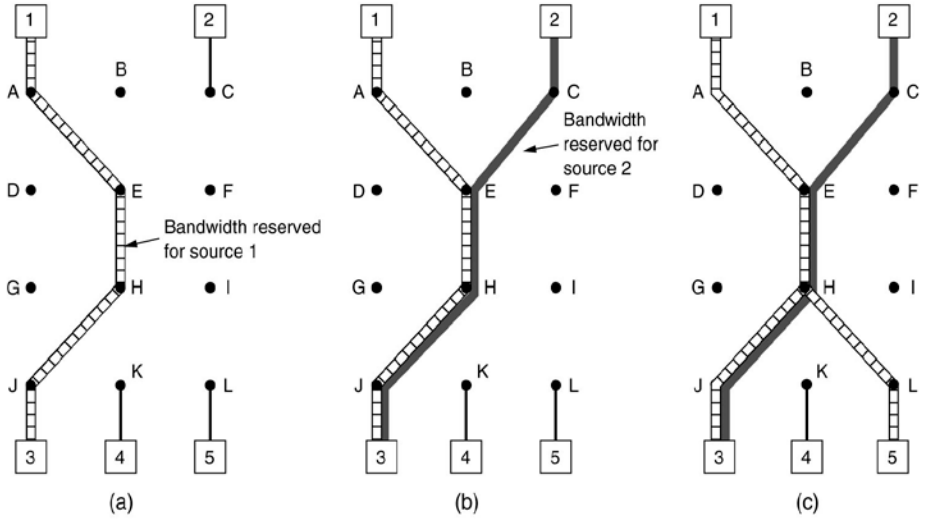


Prenotazione



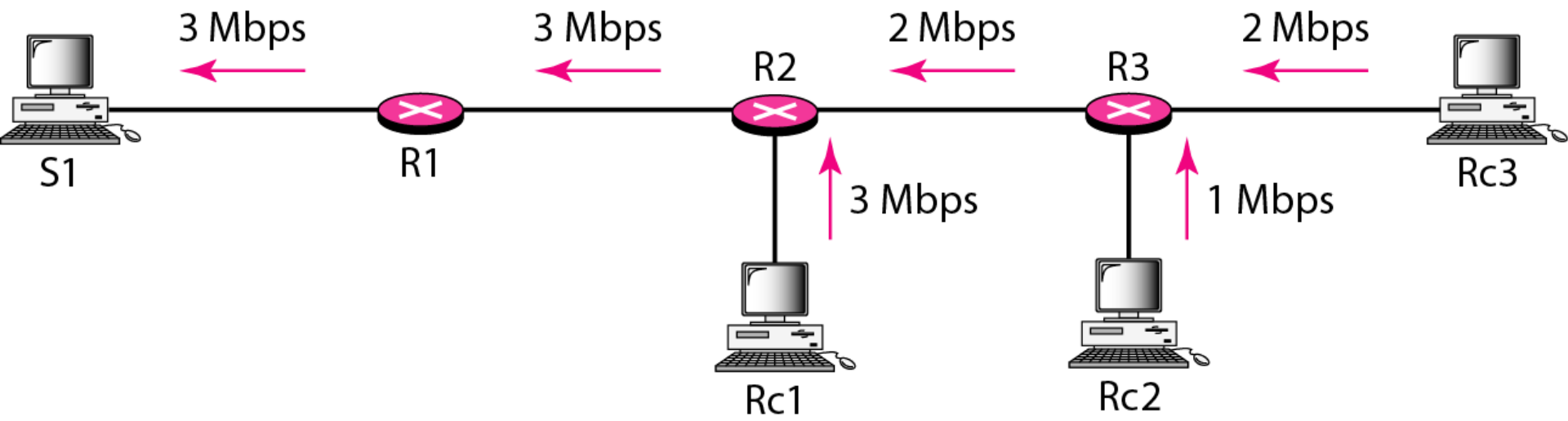
- Host 3 chiede un canale verso 1, quando lo ottiene i pacchetti vanno da 1 a 3 senza problemi
- Se poi host 3 chiede anche un canale verso 2 per vedere due canali tv contemporaneamente, viene riservato un secondo cammino (tra host 3 e router E non basterebbe il primo perché sono due stream indipendenti)

Prenotazione



- Infine anche host 5 vuole vedere un programma di host1, prima viene dedicata banda fino a H ma poi il router vede che c'è gia un feed da host 1 quindi non deve riservare altra banda.
- Se host 3 e host 5 chiedono banda diversa (per esempio uno vuol vedere ad alta risoluzione) allora la capacità riservata deve essere sufficiente per soddisfare il richiedente più avido

RSVP



Problemi dei flow based

- Gli algoritmi flow based hanno dei lati negativi
  - Richiedono di riservare in anticipo risorse per ogni flow, che non scala bene quando ci sono migliaia o milioni di flow
  - Inoltre mantengono tabelle di stati per ogni flow al loro interno, rendendosi vulnerabili a crash nei router
  - Infine richiedono pesanti cambiamenti ai software nei router e complessi scambi router-router nella fase di setup dei flow
- Di conseguenza ci sono pochissime implementazioni pratiche di RSVP

Differentiated Services

- Un approccio più semplice per la QoS chiamato class-based (opposto a flow based), standardizzato da IETF in una architettura chiamata **Differentiated Services** (RFC 2474 e 2475)
- I DS possono essere offerti da un insieme di router all'interno di un dominio di amministrazione. L'amministratore definisce un insieme di classi di servizio con regole di forwarding corrispondenti
- Se un cliente si iscrive a DS, i suoi pacchetti che entrano nel domino possono avere un campo Type of Service, secondo il quale alcune classi avranno un servizio migliore di altre

Differentiated Services

- Al traffico in una classe si potrebbe chiedere di avere una certa forma con un certo rate di uscita
- Un ISP potrebbe far pagare un extra per i pacchetti “premium” o permettere solo N pacchetti “premium” al mese ad un costo fisso aggiuntivo
- Tutto questo schema non richiede nessun complicato setup iniziale, nessuna riservazione di risorse, nessuna estenuante contrattazione end-to-end per ogni flow

Differentiated Services

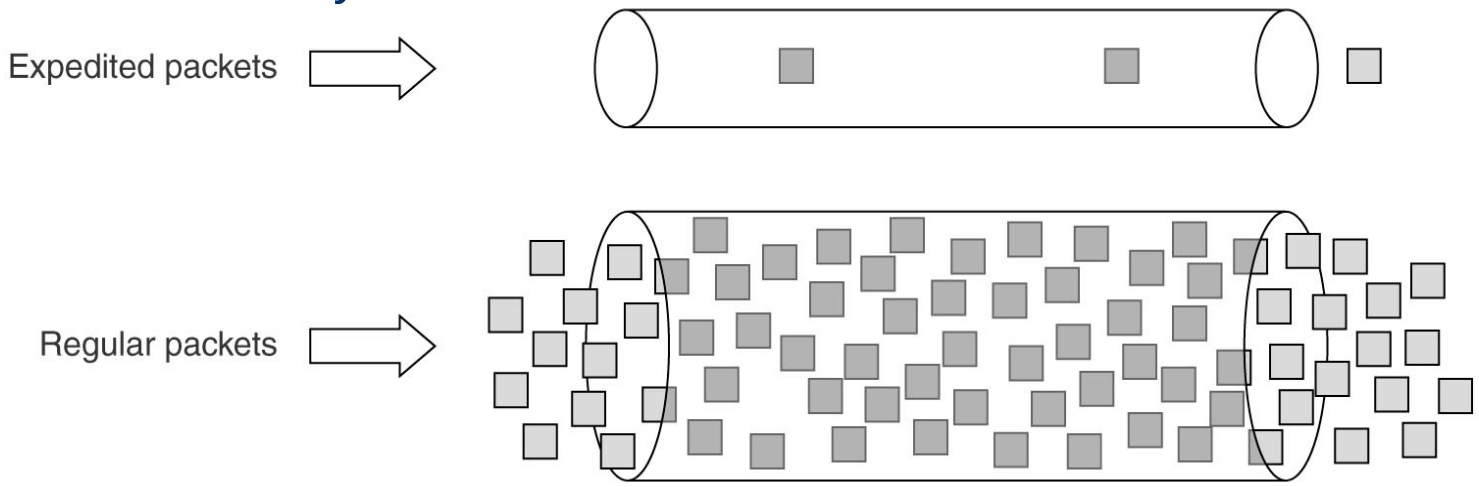
- Un po' come si paga diversamente per posta normale e prioritaria o consegna di pacchi overnigth, un giorno, due giorni, tre giorni. Un aereo ha first class, business, economic.
- Per i pacchetti le classi differiscono in termini di delay, jitter, probabilità di essere scartate in caso di congestione
- Se prendiamo telefonia su IP, con RSVP ogni telefonata ha il suo flusso e le sue risorse riservate, mentre con DS tutte le telefonate nel loro insieme hanno un trattamento privilegiato rispetto al traffico bulk.
- Sono quindi risorse che nessun pacchetto di un file transfer può usare ma anche nessuna telefonata Voip ha risorse private

Explicited Forwarding

- La scelta delle classi spetta agli operatori ma IETF cerca di standardizzare anche questi aspetti perché il traffico possa andare tra subnet di diversi operatori
- La classe più semplice è Expedited Forwarding (RFC 3246) con due classi
- Normale per la maggior parte del traffico e Expedited (Prioritaria) che deve viaggiare nella rete come se non ci fosse altro traffico
- Come se ci fossero due tubi logici (ma la linea fisica è una sola)

Explicited Forwarding

- Ogni router ha due code di uscita per ogni linea, una normale ed una prioritaria.
- Lo scheduling usa un algoritmo tipo WFQ, per cui se per esempio il 10% del traffico è prioritario, gli dedichiamo 20% della bandwidth, per esempio mandando un pacchetto expedited ogni 4 normali (assumento distribuzione di dimensioni uguali per le due classi)
- La linea expedited ha quindi il doppio della banda necessaria e quindi il suo traffico avrà meno delay



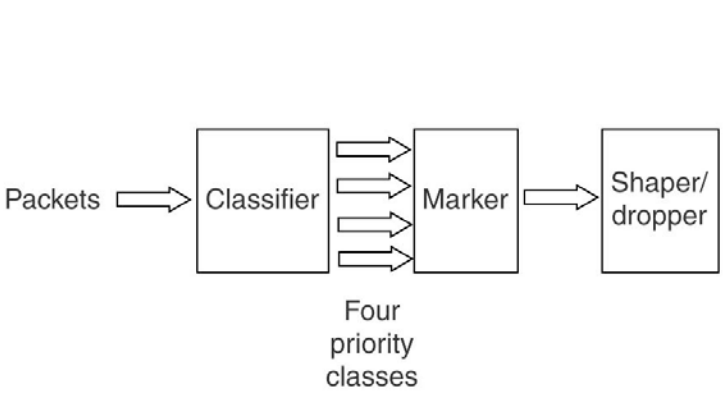
Assured forwarding

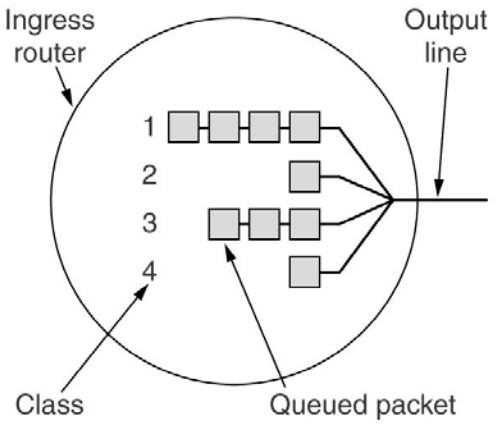
- Uno schema leggermente più elaborato è Assured Forwarding RFC 2597
- Ci sono 4 classi di priorità ognuna con le sue risorse.
- Ogni classe ha 3 probabilità di scarto dei pacchetti in caso di congestione (bassa, media, alta)
- In pratica abbiamo 12 classi di servizio
- All'inizio bisogna classificare i pacchetti per esempio sull'host mittente che conosce meglio quali pacchetti appartengono a quale flow

Assured forwarding

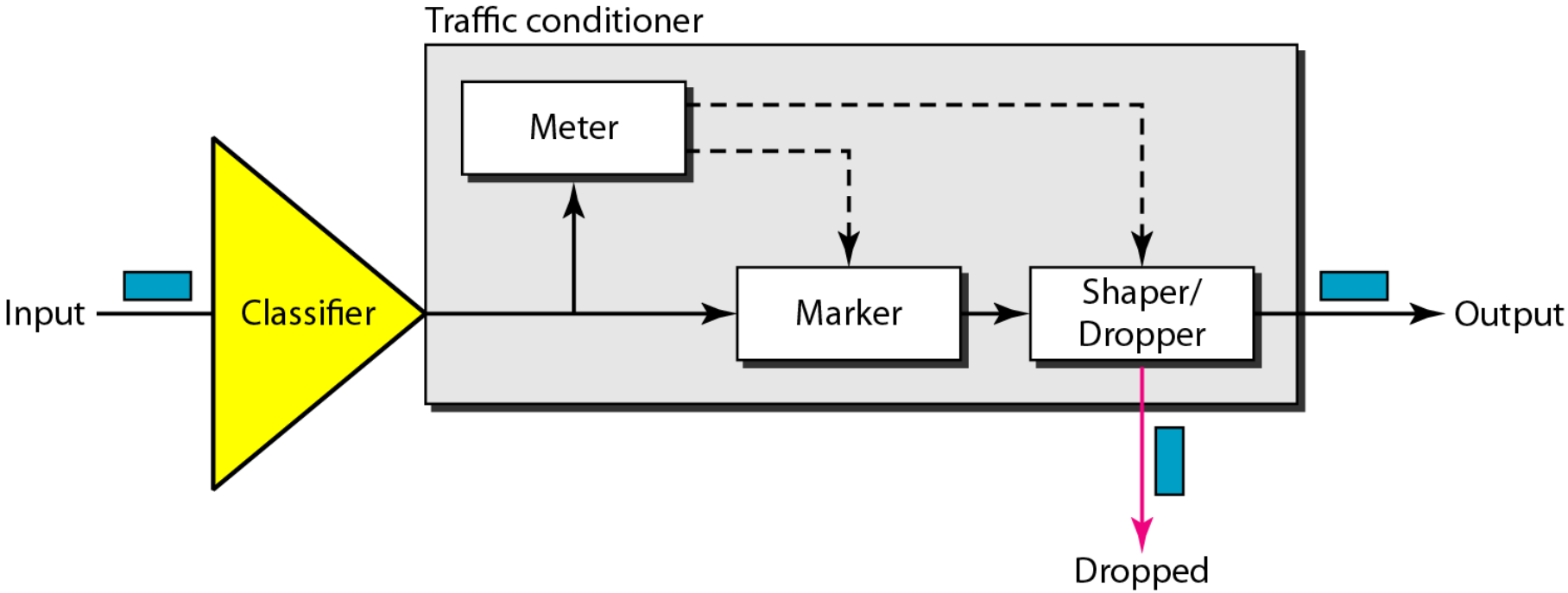
- Poi i pacchetti vengono marcati secondo la loro classe, per fortuna abbiamo gli 8 bit del campo Type of Service di IP. La RFC 2597 specifica che ne possiamo usare 6 per le service class

- Al terzo passo i pacchetti passano uno filtro di shape e di dropping che può ritardare o droppare qualche pacchetto, per esempio usando leaky o toker





classificatore



Label Switching

- Mentre IETF lavorava a Integrated Services e Differentiated Services alcuni venditori di router hanno sviluppato una tecnologia per velocizzare il routing
- Viene messa una label in testa ad ogni pacchetto e questa label viene usata per il routing al posto dell'indirizzo nel pacchetto
- In questo modo il routing è velocissimo, basta guardare in una lookup table dove mandare il pacchetto
- Tecnica molto simile ai Virtual Circuit di X25, Frame Relay e ATM

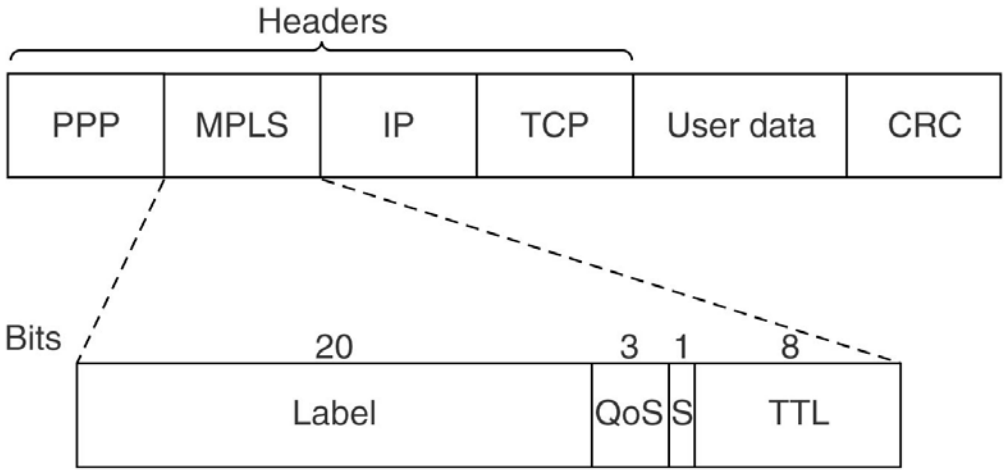
Label Switching

- La nuova tecnica viene chiamata label switching o flow switching e fu infine standardizzata da IETF come MPLS (Multi Protocol Label Switching) RFC 3031 et al.
- Dove mettere la label?
  - Non ci sono campi nell'header IP che non è stato progettato per portare VC
  - Nuovo header MPLS

MPLS ed header MPLS

- MPLS
  - Vediamo una linea router router che usa PPP come framing protocol
  - In questo caso possiamo dire che MPLS è un protocollo di layer 2.5
  - Il campo più importante è Label che contiene l'indice nella lookup table
  - QoS indica la classe di servizio
  - S per reti gerarchiche
  - TTL Time to Live

L2 L2.5 L3 L4 L7



Multi protocol

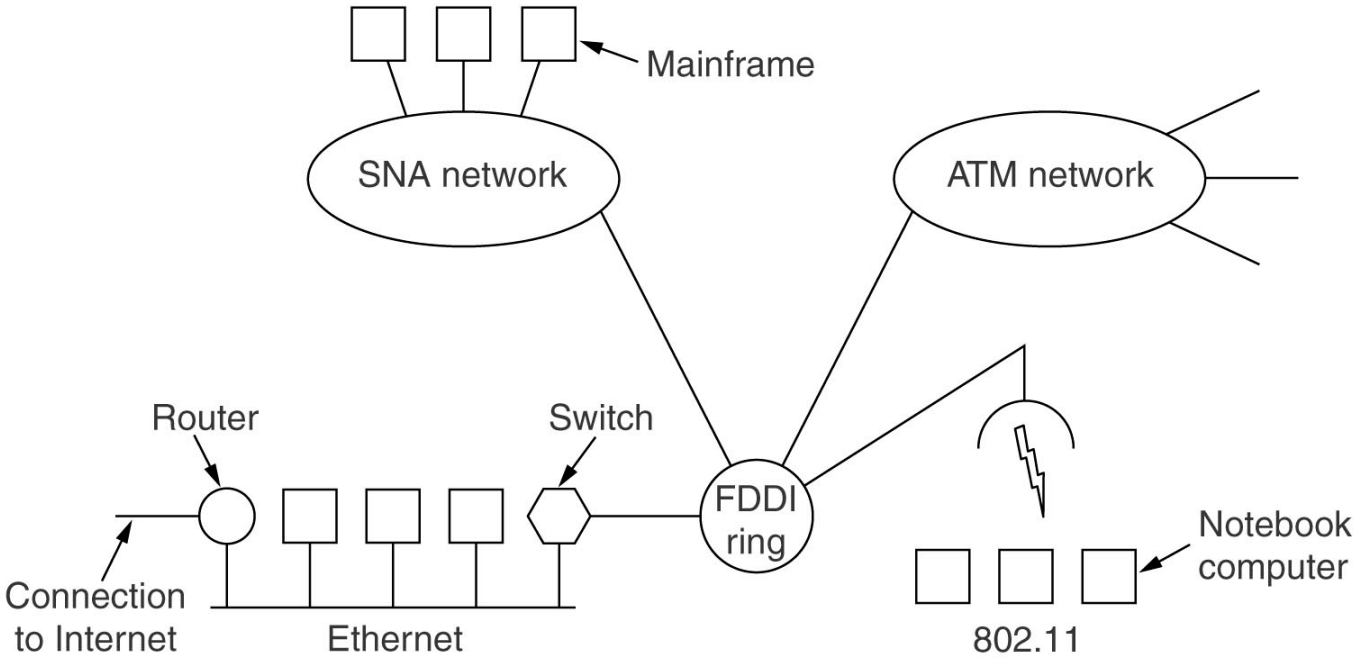
- **MPLS** è indipendente da data link e dal network
- Quindi si possono costruire router **MPLS** che gestiscono sia pacchetti IP che celle ATM (da cui il nome multi protocol)
- Quando un pacchetto arriva il router controlla su quale linea deve forwardarlo usando la label come indice in una tabella e crea un nuova label
- La label infatti ha significato solo locale

Aggregazione

- A differenza dei tradizionali VC in cui ogni flow ha il suo insieme di label, qui i router tendono a raggruppare diversi flow che arrivano ad un particolare router o LAN con un'unica label
- I flow con la stessa label si dicono avere la stessa FEC (**Forward Equivalence Class**)
- Con un VC normale non si può fare questo raggruppamento perché alla fine non sapremmo come distinguere un flusso dall'altro, invece con MPLS il pacchetto contiene sempre l'indirizzo di destinazione finale

Internetworking

- Problema molto complicato. Come unire reti con diversi protocolli?
  - IP, IBM SNA, Appletalk, DECNET, NCP/IPX, ATM
- Serve un router multiprotocollo
- Forme di indirizzamento diverso, dimensioni dei pacchetti…



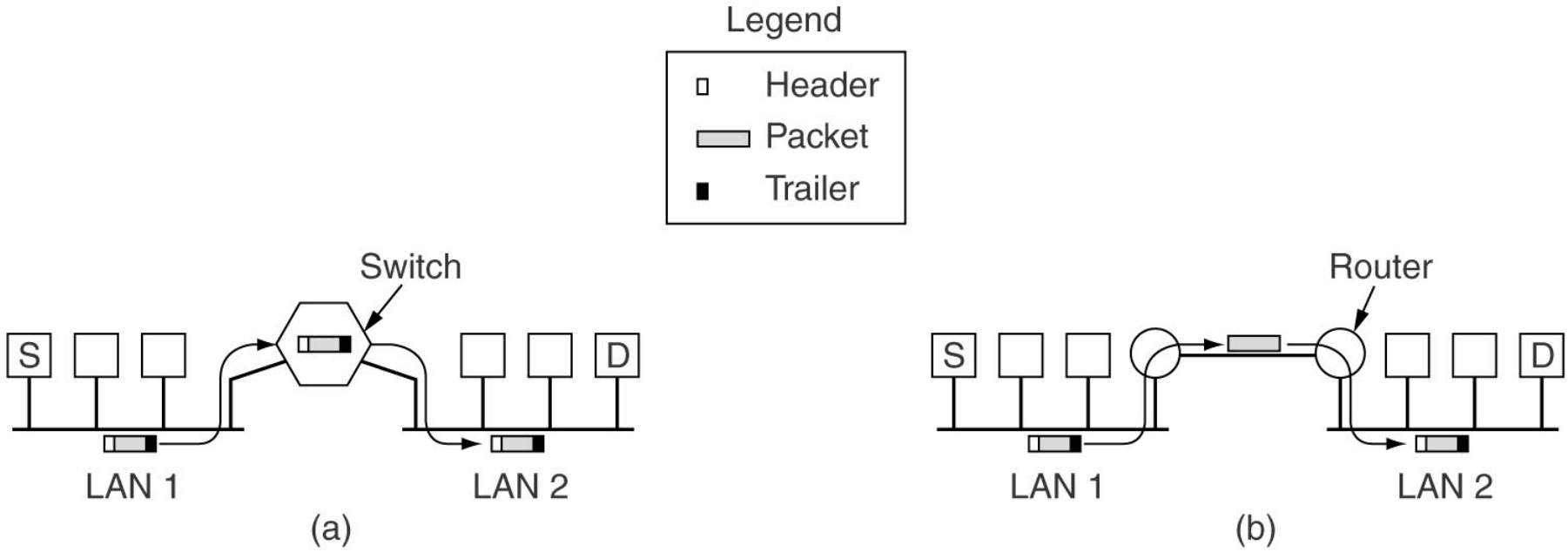
Come differiscono due reti?

Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

Internetworking

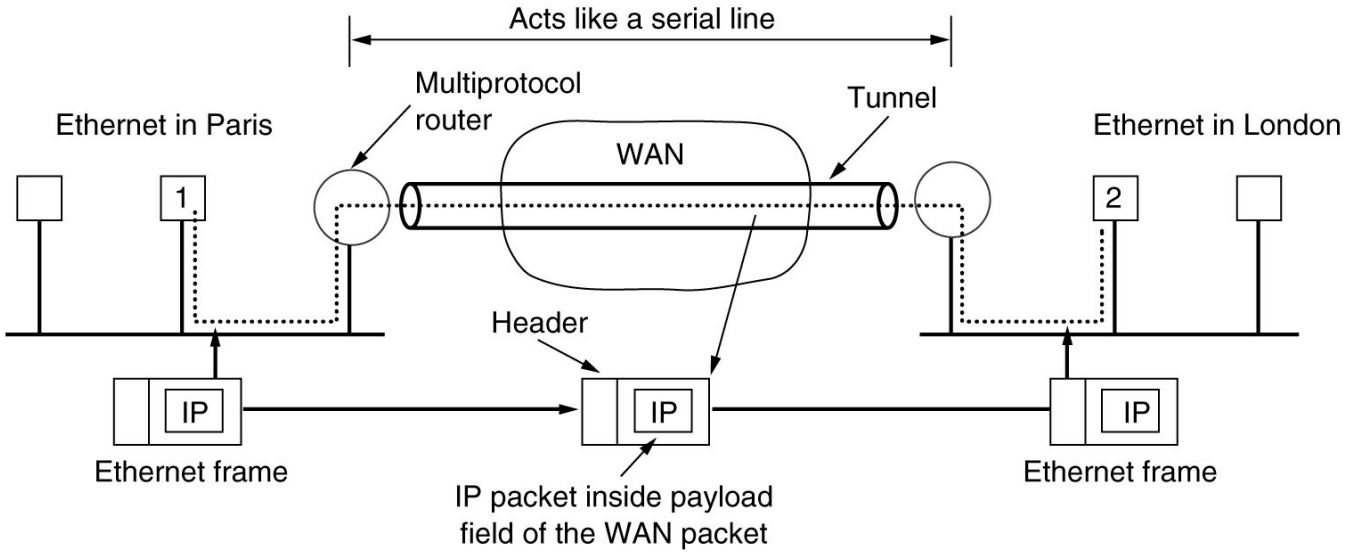
- A livello di trasporto
  - Un gateway di trasporto potrebbe prendere un flusso TCP e mandarlo in una connessione SNA
- A livello applicativo
  - Un application gateway traduce messaggi dal formato internet RFC822 a X400, cambiando diversi header
- A livello network
  - A differenza del livello datalink in cui per esempio due LAN si connettono via uno switch usando mac address a livello network ci devono essere uno o due router che estraggono i pacchetti dai frame

Connetto 2 ethernet



Tunneling

- Tra tutti i possibili casi di internetworking uno è particolarmente semplice
  - Host sorgente e host destinazione sono nello stesso tipo di rete ma c'è un diverso tipo di rete tra di loro
  - Es due reti TCP/IP basate Ethernet connesse da un link ATM

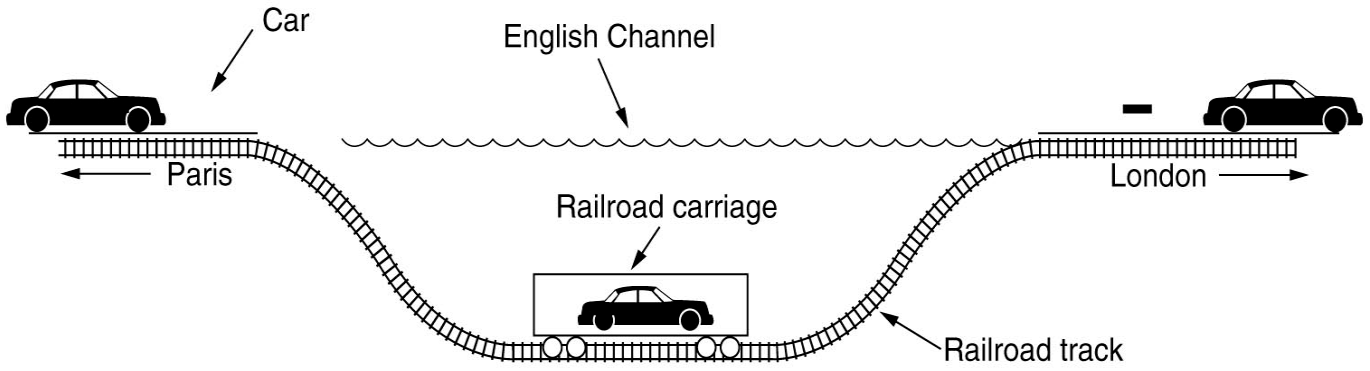


Tunneling

- La soluzione si chiama Tunneling
- L'host 1 manda un pacchetto IP all'host 2, allora lo mette in un frame Ethernet
- Quando questo arriva al router multi-protocollo questo estrae il pacchetto IP, lo mette nel protocollo di network di WAN e lo spedisce al router multiprotocollo all'altro lato
- Qui il pacchetto IP viene tolto e messo di nuovo in un frame Ethernet
- La WAN può essere vista come un tunnel da un router multiprotocollo all'altro

Tunneling

- Il pacchetto IP non si deve preoccupare di interagire con la WAN, e neppure gli host nella Ethernet
- In realtà è come se ci fosse una lunga linea seriale
- Come un auto che si può muovere liberamente nelle strade di Francia o UK ma viene ingabbiata in un treno nel tunnel



Max packet size

- Ogni rete pone limiti alle dimensioni massime del pacchetto
  - Hardware (dimensioni del frame Ethernet)
  - Sistema Operativo (es. tutti i buffer sono da 512 Bytes)
  - Protocolli. Es: numero di bit nel campo di packet length
  - Accordo con standard (inter)nazionali
  - Desiderio di ridurre le ritrasmissioni indotte da errori
  - Desiderio di impedire che un pacchetto occupi il canale troppo a lungo
- Il range dei possibili **max packet size** varia tra i 48 byte di ATM e 65536 di IP

Fragmentation

- Questo implica che quando un pacchetto deve attraversare una rete con max packet inferiore deve essere diviso dai router in frammenti, ognuno dei quali viaggia nel suo pacchetto internet
- Tuttavia, come tutti coloro che abitano con bambini sanno, è molto più facile rompere un oggetto in pezzi che fare il contrario (ce lo conferma la seconda legge della termodinamica)

Trasparente

- Prima strategia: Trasparente
  - Ogni rete rende trasparente la frammentazione ad ogni rete successiva. Tutti i frammenti che entrano in una rete sono indirizzati allo stesso gateway di uscita dove i pezzi sono ricombinati
  - Quindi il gateway di uscita deve sapere quando ha ricevuto tutti i pezzi prima di spedire oltre il pacchetto
  - Tutti i pacchetti devono seguire la stessa strada a costo di perdere in performance.
  - Infine il fatto di dover frammentare e riassemblare i pacchetti ogni volta causa un overhead di cpu e di tempo

Non trasparente

- Seconda Strategia: Non trasparente:
  - I frammenti vengono riassemblati solo alla fine, all'host di destinazione
  - IP per esempio funziona in questo modo
  - Richiede che tutti gli host siano in grado di fare la frammentazione e deframmentazione
  - Ogni frammento ha il suo header che aggiunge overhead, mentre nel caso "trasparente", alla fine della rete small packet, il pacchetto torna grande e l'overhead diminuisce
  - Invece c'è il vantaggio che ogni pacchetto può sfruttare diversi gateways e quindi avere migliori prestazioni

Ricombinazione