**QoS**

Internet nasce per trasportare protocolli da una parte all’altra. Usiamo tantissime applicazioni che si differenziano da quello per cui era stata progettata. Servirebbe che tutti abbiano abbastanza banda ma questo è molto difficile.

Teleconferenza, giochi, realtà virtuali, realtà aumentata, internet per applicazioni di robotica, internet per fornire applicazione di tatto.

**Flusso continuo dei dati:**

Interattività: con altri umani, con il computer 🡪 obiettivo di minimizzare la latenza, basso RTT

Inoltre, vorrei riprodurre il traffico così come è stato inviato ~60fps 🡪 60 frame al secondo vuol dire che ogni sessantesimo di secondo devo mandare un frame e la destinazione deve riceverlo dopo la minor latenza possibile. Inoltre, vorrei che questo **ritardo** sia **costante** in modo tale da mantenere il profilo di dato. Problema nasce se i frame li ricevo ad una diversa frequenza e il video player non riesce più a riprodurre bene. *NOTA: solitamente si raggruppano i frame (es 5) e mando blocchetto di frame e così via in modo tale da ridurre leggermente la frequenza di invio dei pacchetti. I servizi vorrebbero ritardi fissi, costanti, throughput elevato…*

Nota: Sull’host si possono applicare migliorie dal livello 4 in su (L4: modifiche da apportare a TCP/UDP come machine learning; L7: livello applicazione come buffering)  
Nota: Per fare una soluzione completa, bisogna concentrarsi su Host e Rete, noi ci focalizziamo su RETE.

Livello host 🡪 creare programmi che riescono a tollerare le perdite dei pacchetti

**COSA FARE COME RETE per garantire QoS?**

* **OBBIETTIVI**:
  + **Limitare perdita** di pacchetti
  + Avere i **ritardi costanti** ( se i ritardi non sono costanti, problemi)
    - Es: youTube
    - *Mandare fuori i pacchetti con la stessa frequenza con cui arrivano*

**Nota**: TCP in caso di perdita di pacchetti, come se ne accorge, rimanda il pacchetto dunque aumenta il delay 🡪 per applicazioni real time conviene perdere il pacchetto e andare avanti 🡪 **infatti per applicazioni real time si usa udp.**

* + *Per* ***interattività****: oltre a* ***ritardi costanti****, devono essere anche molto* ***bassi***
    - *Es: chiamate (max 100-150ms), tactile Internet (~1ms di round trip)*
  + Avere un **buon throughput**: (collo di bottiglia può diventare lo switch)
    - Migliorare switching, processing, buffering

***Nota****: Throughput e delay sono collegati tra di loro ma avere tanto throughput non mi garantisce un ritardo basso.*

***Nota2:*** *Se ho un throughput elevato, devo avere anche un buffer elevato dove i pacchetti vengono messi in attesa, se lo switch riesce ad avere quella velocità di inoltro, stesso throughput in uscita, ma potrebbe essersi creata una grande latenza o ritardi/frequenze non costanti,* ***Throughput alto 🡪 non vuol dire latenza bassa****. Molto* ***spesso bisogna scegliere*** *tra throughput e latenza (se voglio un throughput alto, avrò un aumento della latenza; se inoltro meno pacchetti (Throughput più basso) avrò probabilmente una latenza minore.*

RITARDO COSTANTE E BASSO: (pacchetti nel buffer, processati, inoltrati)

* Se ho congestione 🡪 aumenta il ritardo (dunque dipende dal carico istantaneo del nodo)

**Nota**: Processing time dipende da tecnologia; WaitingTime non cambia via HW, posso lavorarci andando a limitare il nr di pacchetti in coda. Inoltre, se il buffer si riempie🡪 perde i nuovi pacchetti che gli arrivano.

* **Supportare QoS**
  + Classificazione del traffico
  + Algoritmi di scheduling
  + Controllo sul traffico che entra sulla rete
  + QoS routing (*mpls)*
  + OBBIETTIVO:
    - RIDURRE IL NR DI PACCHETTI SUL NODO
    - GESTIRE IN MANIERA APPROPRIATA i pacchetti che necessitano uno specifico QoS

TOOL PER SUPPORTARE LA QUALITA’ DI SERVIZIO: difficile fornire garanzie

Identificare quale qualità si deve fornire 🡪 dunque capire su quale coda devo inserire quel pacchetto.

Es:

* Suddivido il traffico in due classi (RealTime, BestEffort) 🡪 **CLASSIFICAZIONE**
* Creo due buffer:
  + Buffer1: tutto traffico realTime
  + Buffer2: tutto traffico bestEffort
* Decido quale strategia usare

**CLASSIFICAZIONE**

* **QUINTUPLA**: per identificare il flusso dati/classe direttamente dall’header del pacchetto  
  (non Payload che potrebbe essere cifrato ---ipSec ha alcune soluzione in cui si cifra anche header ma raro---)
  + - IPdestinazione
    - IPsorgente
    - Protocollo di trasporto
    - Porta destinazione
    - Porta sorgente

Es: se arriva su porta 80 so che è http, se arriva su quella di traffico video, la tratto come traffico video

**HW**: ASIC o CAM possono essere soluzioni HW per svolgere algoritmi semplici

**Per salvare classe**: DifferentalService in IPV6, EXPERIMENTAL BIT IN MPLS

**Nota**: classificazione svolta da operatore e non da client

**GESTIONE/SCHEDULING:**

* Metto pacchetti in coda FIFO e poi li processo
* Se ho più buffer 🡪 metto vari pacchetti nelle varie code e devo decidere da quale devo fare POP 🡪 ATTIVITA’ DI SCHEDULING
  + Ordine di arrivo a prescindere dal buffer –slide17 🡪 qualcuno non soddisfatto sicuro
* **ALGORITMI DI SCHEDULING**
  + **RoundRobin**: 1-2-3-1-2-3-1-2-3 se c’è pacchetto, altrimenti salto e passo a successivo
    - Funziona abbastanza bene
  + **RoundRobin** **pesato**: 1-2-3-1-2-3-1-2-3 ma con pesi maggiori ai buffer che voglio favorire (se hai 1/2, una volta prende, una volta ti salta)
    - Ogni giro prendo il nr di pacchetti selezionato dal peso
    - Nota: Peso posso farlo variare
  + **Priority queuing**: associo una priorità ai buffer
    - Prendo sempre i pacchetti al buffer con priorità più alta
    - Priorità a priori, non può variare
  + **Class based Queuing CBQ**:
  + **Weighted Fair Queuing WFQ**: esegue operazioni in FloatingPoint, ogni flusso ha una coda associata
    - Ottimo se:
      * So come è stato generato il traffico
      * Il nr di flussi da gestire è molto basso
      * Ho un HW che sappi fare FloatingPoint
  + **Deadline Queueing**: prima chi scade prima

Discorso delle code può essere usato sia per input che per output 🡪 interessa molto quindi la capacità dello switch di processare e inviare pacchetti

**Capacità di switching** 🡪 riguarda HW 🡪 trasportare un pacchetto da input ad output nel modo più veloce possibile 🡪 SWITCHING immediato

Oltre agli algoritmi di classificazione scheduling abbiamo bisogno di fare controllo su traffico

**CONTROLLO SU TRAFFICO**Può essere fatto:

* **A LIVELLO DI PACCHETTO**
  + **Policing**: Consumatore e operatore possono mettersi d’accordo sui requisiti
  + **Shaping**: Operatore chiede al consumatore di generare del traffico con un certo pattern do traffico (Frequenza, throughput) 🡪 *es: se sta inviando troppo traffico, deve rallentare un po’ (magari se non rallenti, ti perdo il pacchetto)*
* **A LIVELLO DI FLUSSO:** si basa sul sapere che tipo di traffico genererà l’utente
  + **Riservare risorse:** es con RSVP per IP
  + **Network** **Engineering:** quanti link e router mettere e dove 🡪 dimensionare la rete nella maniera opportuna
  + **Traffic Engineering:** decidere come il traffico deve andare sulla rete (es: con MPLS puntavo a bilanciare il traffico sulla rete)

Controllo sul traffico: devo controllare che il **traffico sia quello che mia spettavo**:

* **TOKEN/LEAKY BUCKET**: Invio un pacchetto ogni volta che scende un bucket 🡪 riesco a gestire un medio e massimo 🡪 se ho tanti pacchetti in coda 🡪 o aspettano o scarto
  + Se abbiamo fatto policing: io li scarto perché mi hai mandato più di quanto concordato
  + Se faccio shaping: invece di scartare questi pacchetti, si rallentano pacchetti (solitamente svolto lato consumatore)

Ho modo quindi **di controllare bit rate medio e momento di picco** in cui genero traffico (BURST): scartando/rallentando/mettendo in una coda a priorità più bassa dei pacchetti

* **Riservare risorse:** es con RSVP
  + **Solo se proprio serve:** bisogna fare signaling per segnalare questa richiesta
* Posso fare **QoS routing**: cerco una strada con quelle caratteristiche
  + Può causare instabilità
* Soluzioni **NE-TE**
  + Dimensiono la rete
  + Determino matrici di traffico
  + Monitoro continuamente se il dimensionamento e l’instradamento sono corretti
* Sono stati proposti per supportare QoS
  + **IntServ**: IntegretedServices: riservo le risorse **per flusso** (RSVP)
    - * Quintupla
      * Avg bit rate
      * Max bit rate
    - **Ogni router** riceva questa richiesta con le caratteristiche e decide se accettare o meno
    - **Decisione per flusso**
    - **Non scalabile**

Non tutti i dispositivi lo implementano, molto complesso su larga scala

* **DiffServ**: Offrire del **supporto** (non garanzia) per classi
  + No garanzie
  + Non riservo risorse
  + Supporto per classi attraverso
    - TrafficEngegnering
    - Network Egnegnering
    - Policing

**Semplice e scalabile** ma con efficienza leggermente minore