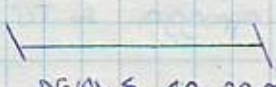
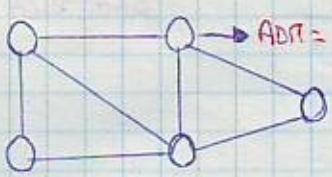
Succede che abbiamo un accodamento intenso e quindi i pacchetti vocali subiscono un ritardo notevole. Siccome:



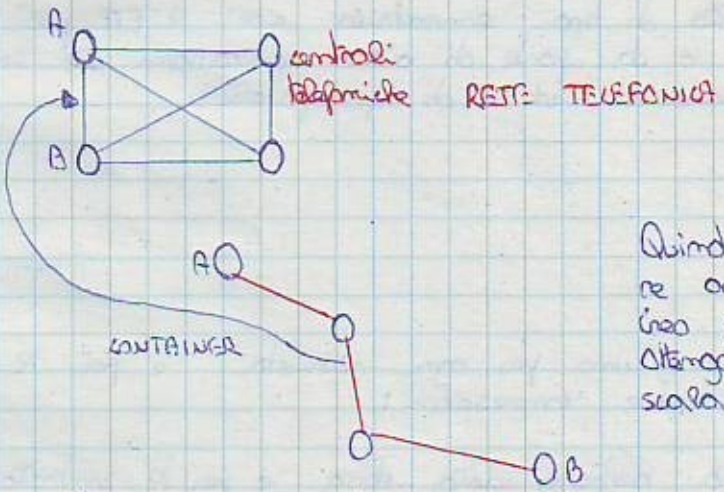
diventa inutile se poi tutti i pacchetti arrivano lo stesso a destinazione.

Questa cosa non accade sulle reti telefoniche classiche cioè quelle che funzionano a circuito cioè quelle che hanno un canale dedicato. Questo perché il ritardo è deterministico e piccolino. Un problema che rappresenta una sfida per il futuro è il problema legato alla capacità di garantire la qualità del servizio. Tutti sanno che gli ISP coincidono con i vecchi operatori di telefonia. Questi ultimi però hanno una rete. Questi hanno una rete fisica che è costituita da un insieme di nodi e di link:

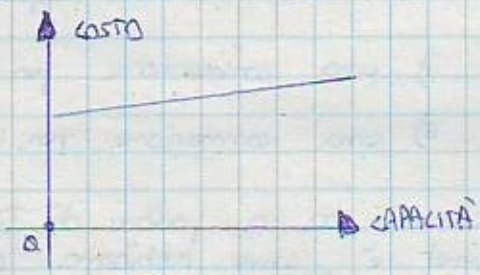


che sono degli smistatori di flusso.

ed hanno una rete telefonica che è sovrapposta a quella fisica. Quindi:

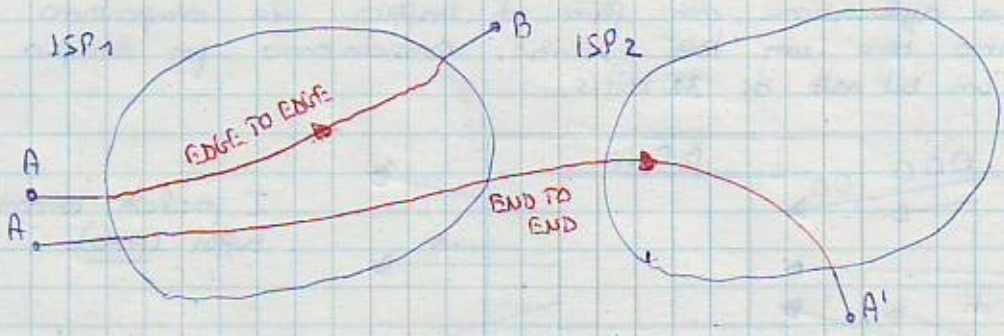


Quindi con una rete fisica posso servire anche più reti logiche sovrapposte. Uso un container che unisce tutti i nodi. Ottengo così una grande economia di scala.



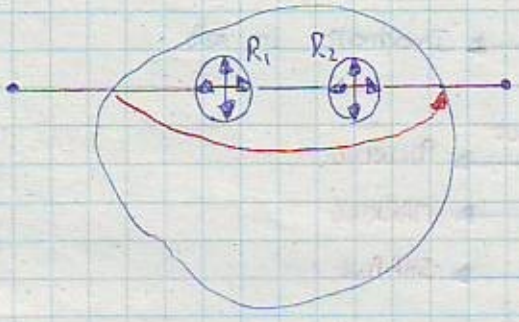
Quindi se raddoppio la capacità, il costo non raddoppia.

La gestione del traffico è la chiave per garantire la qualità del servizio ma è piuttosto complicata. Consideriamo:



Un conto è garantire la qualità del servizio END TO END ed un conto è garantire la

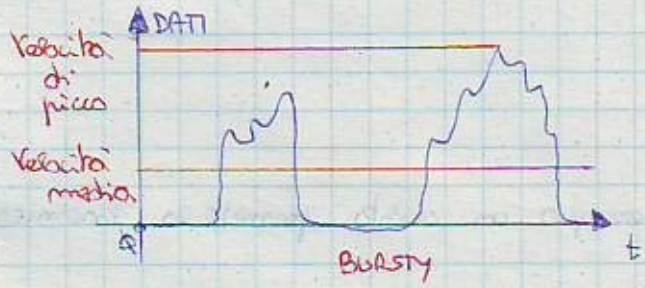
qualità del servizio END-TO-END. È più facile garantire la qualità di servizio per END-TO-END. Consideriamo:



Ogni router avrà l'occasione di rovinare la qualità del servizio a causa delle proprie politiche di buffering. Deve quindi garantire la qualità del servizio a livello locale.
Per prima cosa bisogna specificare quale qualità di servizio si vuole garantire. Quindi:

SLA = Service Level Agreement mi definisce la qualità del servizio che l'utente si aspetta.

Questo SLA dipende dalla banda minima richiesta se si parla di dati, o dipende dalla soglia di ritardo se si parla di telefonia.
Per quanto riguarda per esempio le VIDEO/AUDIO STREAMING bisogna garantire sia la banda sia la soglia. Lo streaming purtroppo non può essere fatto in real-time. Quindi il contratto tra cliente e ISP viene specificato attraverso SLA. Bisogna poi anche specificare il TRAFFIC PROFILE che ci dice come è fatto il traffico inserito dall'utente. Si ricordi che la rete deve possedere delle strutture in grado di reggere un determinato TRAFFIC PROFILE.
Se abbiamo una sorgente di dati si ha:

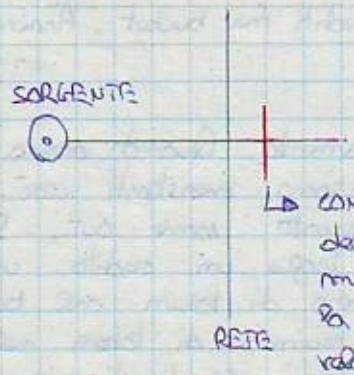


Per esempio: traffico vocale.

La velocità che incide sulla banda è la velocità media e non quella di picco.
Definiamo:

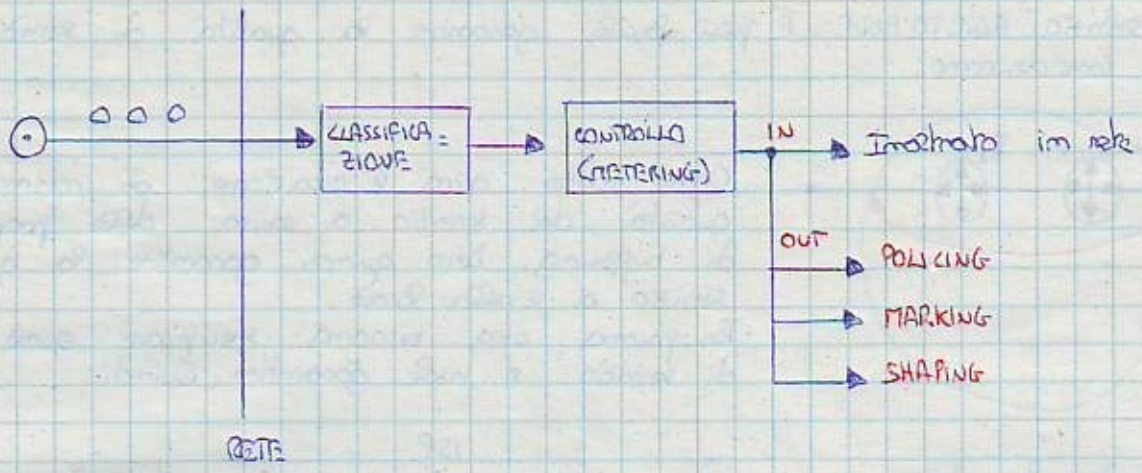
$$\text{BURSTINESS} = \frac{V_{picco}}{V_{media}}$$

Poi abbiamo sorgenti domestiche più "addomesticabili". Per ogni una singola sorgente di traffico bisogna controllare il traffico.

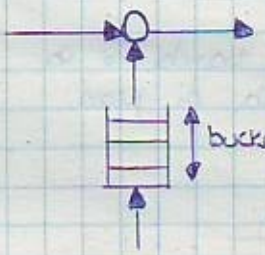


↳ CONTROLLO DEL TRAFFICO: per fare in modo che il traffico stia dentro ai dati specifici non definiti. Tale controllo mi controlla la velocità media, la velocità di picco, la lunghezza massima dei burst dei pacchetti alla velocità di picco. Esso si basa su ciò che è stato dichiarato nel traffic profile.

Inoltre il contratto contiene l'identità dell'utente che viene a sua volta controllata. Vediamo brevemente il controllo del traffico immesso in rete:



La classificazione serve per vedere a quale contratto appartiene il pacchetto. Il controllo serve per capire se il singolo pacchetto è aderente al profilo di traffico dichiarato. Se un pacchetto è fuori dal profilo, posso ammazzarlo attraverso il POLICING, oppure posso marcarlo come pacchetto fuori profilo settando il campo TOS nell'header IP; I pacchetti marcati passano solo se la rete è libera altrimenti vengono uccisi. Alternativamente un pacchetto può essere ritardato in un buffer attraverso lo shaping. Quest'ultima operazione manda avanti il pacchetto solo quando è possibile farlo rimanendo aderenti al profilo del traffico. Vediamo ora un meccanismo che realizza il metering e che viene chiamato token bucket. Graficamente si ha:



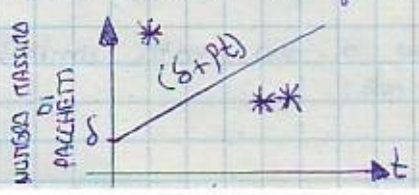
TB(δ, ρ)

bucket = contiene dei crediti dove per esempio un credito permette la trasmissione di un pacchetto

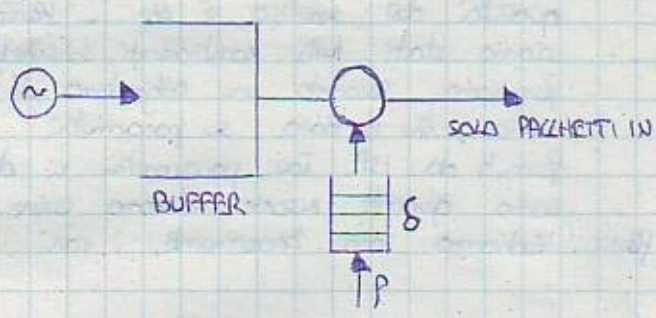
Il bucket ha due parametri:

1) δ = che mi da il numero massimo di crediti accumulabili
2) ρ = mi permette la velocità di arrivo dei crediti nel bucket. Arriva un credito ogni $1/\rho$ secondi.

Si noti che δ si misura in crediti. Vediamone il funzionamento. Quando arriva un pacchetto si va a controllare i crediti nel bucket. Se tali crediti sono inesistenti cioè se il numero di crediti nel bucket è zero, allora il pacchetto viene marcato come OUT. Se nel bucket c'è almeno un credito il pacchetto passa come IN e si toglie un credito dal bucket. Se mi arriva un token (credito) e se il numero di token nel bucket è minore di δ, aumento di un credito il bucket altrimenti se il numero di token nel bucket è uguale a δ, cancello il nuovo credito e il bucket rimane così. Ci chiediamo ora quale è il numero massimo di pacchetti IN permesso in funzione del tempo.

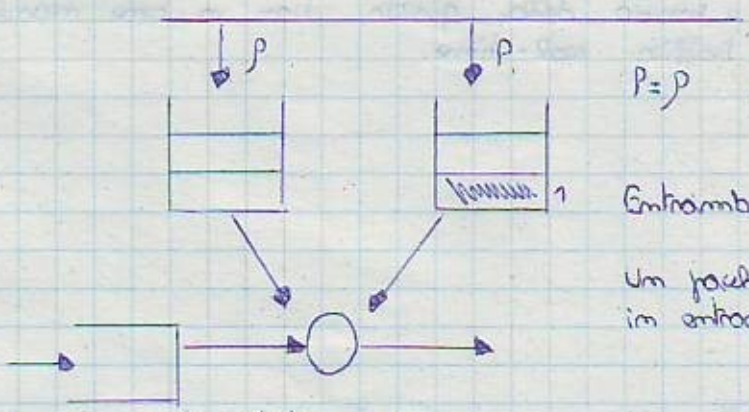


Il processo di metering quindi ci consente di verificare le burst dei δ pacchetti in profile e la velocità media dei pacchetti IN pari a ρ pacchetti al secondo. Il profile di traffico è quindi un token bucket con un certo ρ e un certo δ. Analizziamo ora cosa succede nel caso dell'operazione di shaping. All'arrivo di un nuovo pacchetto se il buffer ha già dei pacchetti quest'ultimo viene accodato. Se il buffer è vuoto a vedere il contenuto del bucket. Se il bucket ha almeno un credito allora passa il pacchetto come IN e si toglie un credito, altrimenti il pacchetto resta in coda.



Quando arriva un token se il bucket è vuoto a vedere il buffer di ingresso. Se il buffer ha dei pacchetti si serve un pacchetto e non si incrementa il bucket. Se il buffer è vuoto allora incrementa il bucket di un credito. Se il bucket ha meno di δ crediti incrementa il bucket e se il bucket ha δ crediti si cancella il nuovo credito.

Tutte queste operazioni rendono prevedibile il traffico che entra in rete. Questo permette anche la allocazione delle risorse. Ma come si limita la velocità di picco, visto che il token bucket non la fa? Si applicano due token bucket.

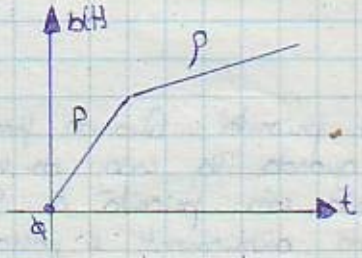


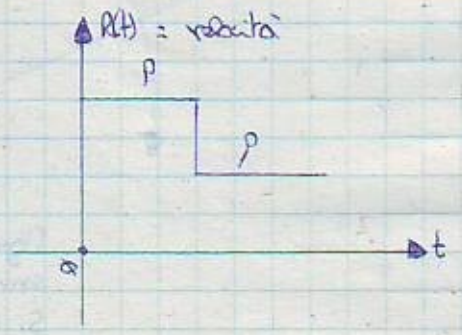
$P = \rho$ con P = velocità di picco
ρ = velocità media

Entrambe queste velocità vengono controllate.

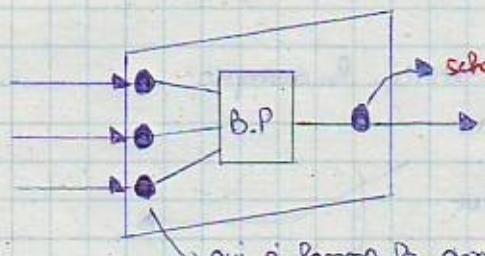
Un pacchetto passa se è presente almeno un token in entrambi i bucket.

Sia $b(t)$ il numero di bit trasmessi.



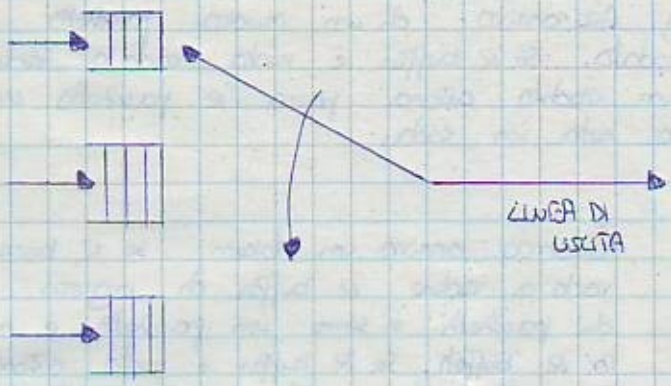


Definiamo edge router un router che controlla se effettivamente si comporta bene rispetto al contratto. Schematicamente si ha:



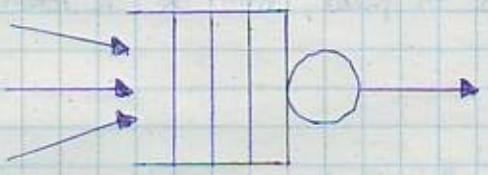
Nella scheda di interfaccia c'è lo scheduler.

Lo scheduler sostanzialmente è quel dispositivo che decide quale coda servire e lo fa seguendo delle specifiche politiche.



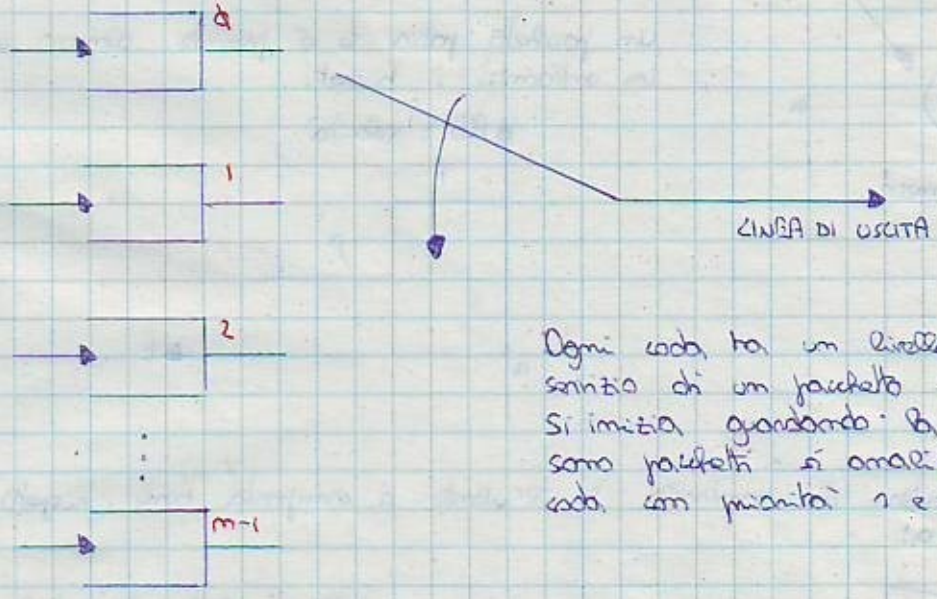
È qui che si determina la qualità dei servizi ricevuti dai flussi. Una condizione necessaria e sufficiente per la garanzia della qualità del servizio è che i flussi siano stati tutti controllati. Lo scheduler funziona secondo un algoritmo di scelta che si basa su parametri forniti da ISP. Tali parametri ci dicono quante risorse devono essere assegnate alla singola coda e quindi al singolo flusso. Vediamo ora brevemente i vari tipi di scheduler:

1) FIFO



Questo scheduler non serve a niente. Va bene per il servizio best effort ma non va bene per il controllo e servizio della qualità. Non va bene neanche per il traffico real-time.

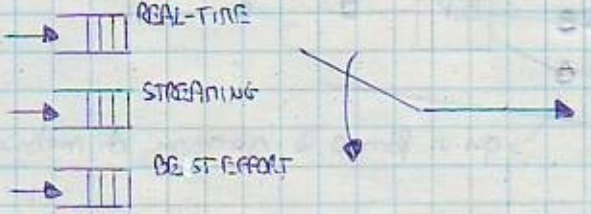
2) STATIC PRIORITY



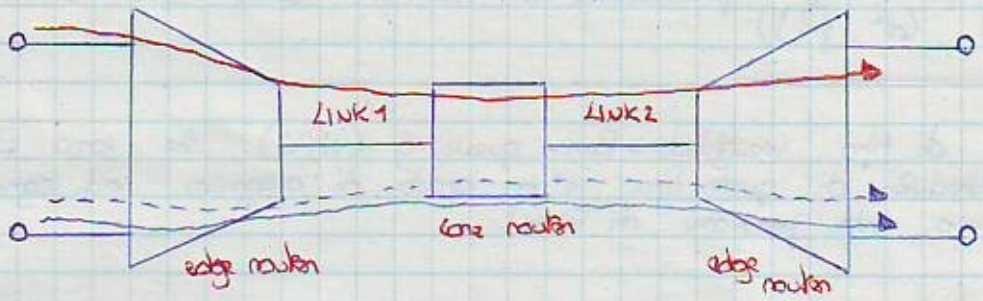
Ogni coda ha un livello di priorità. Quando finisce il servizio di un pacchetto si guarda la coda da servire. Si inizia guardando la coda con priorità 0. Se ci sono pacchetti si analizzano, altrimenti si passa alla coda con priorità 1 e così via.

Così facendo posso anche definire classi o categorie di servizi. Per esempio:

1) SERVIZI REAL TIME
2) SERVIZI DI DATI O STREAMING
3) BEST-EFFORT



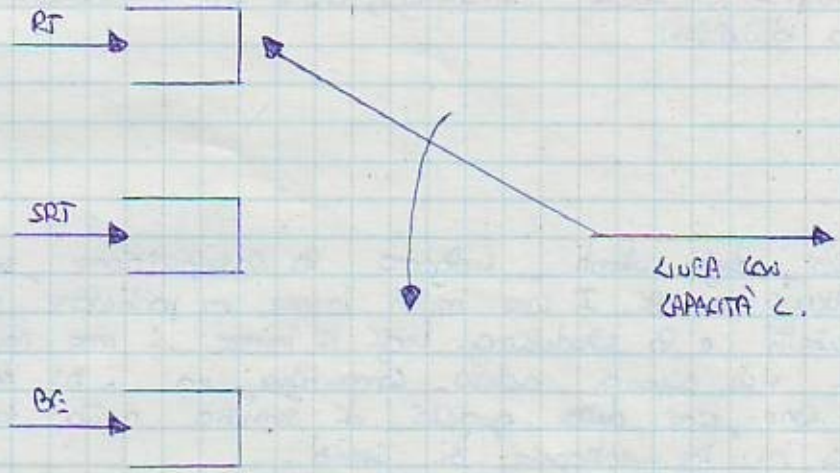
All'ingresso in rete i pacchetti vengono classificati per cui oltre ad individuare il traffic profile viene identificato anche il service profile all'interno del contratto. In merito il campo TOS dell'header IP viene usato per definire il livello di priorità definendo così le categorie di servizi. Consideriamo ora:



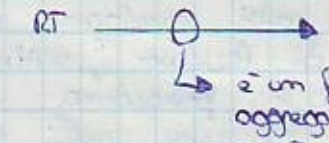
Come dimensiono i link e i router in modo che la qualità di servizio sia rispettata? Nel momento di picco cioè nel momento di traffico più intenso si prevede di avere N_{RT} richieste di servizio real-time, N_{STR} richieste di servizio di tipo streaming ed N_{BE} richieste best-effort. Progetto le capacità dei link 1 ed del link 2 che sono rispettivamente C_1 e C_2 e sfrutto la parametrizzazione degli schedulers in modo da garantire la qualità di servizio nella situazione di massimo traffico. Inoltre bisogna attivare la procedura CAC = Connection Admission Control che blocca all'ingresso le richieste che eccedono i limiti massimi per i quali la rete è stata progettata. Abbiamo quindi due funzioni di controllo in real-time:

1) CAC
2) POLICING / SHAPING / MARKING

Così in ogni punto di accesso alla rete ogni flusso è dimensionato. Si noti che il ritardo si ha quando convergono i flussi e non quando si dirama. Ci chiediamo ora però come posso parametrizzare la linea di uscita e lo scheduler. Supponiamo di avere uno scheduler STATIC PRIORITY:



Dobbiamo garantire i ritardi probabilistici data questa ordine di servizio. Si noti che:



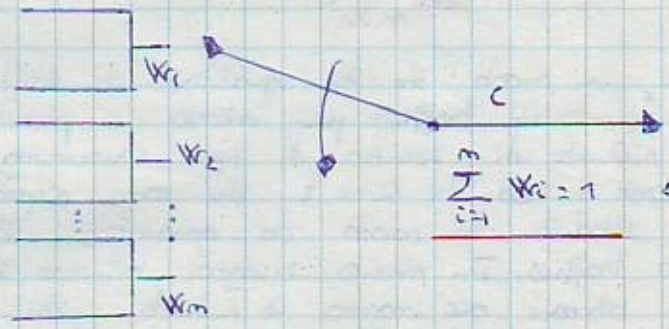
Per ogni flusso definisco l'aggregato massimo. Esiste una formula che, dati il numero di classi di servizio e dato un $TB(\rho_i, \delta_i)$ che descrive le proprietà del traffico della classe di servizio i nel caso di traffico di picco, e dati le qualità di servizio: $P(D^i > d^i) \leq p^i$, ci da la capacità della linea di uscita dato lo scheduler di tipo STATIC PRIORITY. Purtroppo lo STATIC PRIORITY funziona, con poche classi di servizio, ma i possono trasportare servizi di real-time. Si noti che:

D^i = classe del traffico
q^i = soglia
p^i = probabilità

Quindi: soglia di ritardo; probabilità di violazione della soglia di ritardo.

$$C = p(TB(\rho, \delta), (d^i, p^i))$$

Esistono anche schedulers di tipo Weighted-Fair-queueing (WFQ) che sono schedulers di tipo RATE BASED. Uno scheduler di questo tipo è in grado di assegnare ad ogni coda una certa banda (cioè una certa frazione di link).



$$\sum_{i=1}^{n} w_i = 1 \quad \text{con} \quad i \geq C \cdot w_i$$

Come si può notare è uno scheduler più evoluto. Tutti i router hanno i due tipi di scheduler menzionati. Inoltre questo scheduler consente di gestire il flusso singolo e classificare i flussi ai bordi della rete e mantenere l'identità di ogni singolo flusso. Purtroppo ciò costa e soffre di scalabilità. Vediamo ora l'architettura per le reti IP con qualità di servizio. Esistono:

- Servizi IP integrati (INTSERV) (IS)
- Servizi IP differenziati (DIFFSERV) (DS)

entrambi possono usare MPLS (Multiprotocol Label Switching), che è un meccanismo di routing aggiuntivo. I router vengono divisi in:

1) edge routers
2) core routers (router interni)

Gli edge routers raccolgono il traffico degli utenti, svolgono la classificazione, svolgono funzioni di POLICING, SHAPING, MARKING e CAC. I core routers invece, in particolare nei DS fanno solo il forwarding dei pacchetti e lo scheduling. Negli IS invece i core routers possono intervenire nelle funzioni viste all'esame. Comunque sia i DS che i IS si occupano della EDGE TO EDGE QOS, cioè della qualità di servizio nella comunicazione EDGE-TO-EDGE. Negli IS si distinguono in tre categorie di servizi:

1) Guaranteed Service (GS)
2) Controlled Load Service (CLS)
3) Best effort (BE)

Ogni singolo flusso può appartenere a uno di questi tre servizi. C'è una gestione