CAPITOLO 1
ARCHITETTURA DEL PROTOCOLLO XCP

1.1 Generalità

Nel protocollo XCP viene utilizzato un controllo di congestione attraverso un meccanismo retroazionato, in cui la rete informa esplicitamente lo stato di congestione al sender in modo che quest'ultimo reagisca proporzionalmente. Questa procedura, infatti, permette al sender di decrementare la velocità di byte trasferiti in modo rapido, nel caso in cui la rete è altamente congestionata o lentamente nel caso in cui la velocità di trasferimento supera di poco la capacità del canale di comunicazione. Di conseguenza il protocollo risulta essere più reattivo e meno instabile per piccoli ritardi.

In secondo luogo, l'aggressività della sorgente dovrebbe essere regolata in accordo al ritardo nel ciclo di retroazione. La dinamica del controllo di congestione può essere riassunta come un controllo ad anello retroazionato con ritardo. La caratteristica principale di questo sistema è che diventa instabile per elevati ritardi di retroazione e quindi, per tener conto di questo effetto destabilizzante, il sistema rallenta l'effetto all'aumentare del ritardo sul ramo di retroazione. Questo significa che all'aumentare del ritardo, la sorgente dovrebbe cambiare il suo rate di invio più lentamente. Questo problema è stato risolto in parte da altre ricerche [Paganini01], ma un altro problema importante da risolvere è come la retroazione dovrebbe dipendere esattamente dal ritardo per garantire la stabilità. Quindi, sfruttando le nozioni della teoria del controllo, la retroazione basata sul rate - mismatch dovrebbe essere inversamente proporzionale al ritardo e la retroazione basata sul queue – mismatch dovrebbe essere inversamente proporzionale al quadrato del ritardo.

Inoltre, un principio fondamentale derivante dalla teoria dei controlli, specifica che un controllore deve reagire tanto velocemente in relazione alla dinamica del segnale controllato. Nei controllori a gestione di coda attiva (AQM – Active Queue Management), il segnale controllato è il traffico complessivo che attraversa il link ed il controllore cerca di regolare l'input rate cercando di raggiungere il valore della capacità del link. Questo obbiettivo può essere irraggiungibile quando il traffico inviato riguarda flussi di tipo TCP, perché la dinamica di questo tipo di traffico dipende dal numero di flussi (N). Infatti il rate di ingresso incrementa di N pacchetti per RTT, o decrementa proporzionalmente a 1/N e poiché il numero di pacchetti complessivo non è costante e cambia temporalmente, un controllore AQM con parametri costanti risulterebbe inefficace. XCP cerca di risolvere questo tipo di problema fornendo la dinamica del traffico complessivo indipendentemente

dal numero di flussi e per questo separa il controllo di efficienza (controllo di congestione) dal controllo di fairness (fairness control).
In genere, efficienza e imparzialità di banda sono governati dalla stessa legge di controllo, come ad esempio l'AIMD (Additive – Increase Multiplicative – Decrise) per TCP, utilizzata contemporaneamente per efficienza e distribuzione di banda. XCP utilizza un controllo di tipo MIMD (Moltiplicative – Increase Moltiplicative Decrease) per il controllo di efficienza e AIMD per il controllo sull'imparzialità di banda. Concettualmente, infatti, efficienza e imparzialità di banda sono indipendenti; l'efficienza riguarda solo il comportamento del traffico complessivo: quando il traffico generato eguaglia o si avvicina alla capacità del link, non si genera coda e quindi l'utilizzo del link è ottimo. L'imparzialità della distribuzione del traffico (fairness) riguarda, invece, il throughput dei flussi che condividono il link. L'ottimalità si ottiene quando i flussi che condividono un link hanno lo stesso throughput senza creare congestione.
Il router utilizzato in XCP effettua sia il controllo di efficienza (EC) che il controllo sulla imparzialità di banda imparziale (FC). La separazione dei due controlli permette di, in caso necessario, apportare modifiche su uno dei due controllori senza rianalizzare l'altro. Inoltre, esso fornisce una struttura flessibile per quanto riguarda l'allocazione di banda .

1.2 La struttura

XCP fornisce una giuntura tra i terminali e i routers. Come TCP, XCP è un protocollo di controllo di congestione window – based, cioè trasmette più pacchetti consecutivi in attesa dei riscontri dei primi trasmessi ed è best effort. Inoltre XCP fornisce un'architettura abbastanza flessibile poiché può supportare differenti servizi e può coesistere con TCP nella stessa rete e quindi è detto TCP – friendly.
Nel protocollo XCP i componenti che gestiscono la comunicazione sono : il sender host (terminale sorgente), il receiver host (terminale ricevente) e i nodi intermedi rappresentati dai router in cui si vengono a creare le code di attesa. Prima di tutto il sender stima la congestion window (cwnd) e il round trip time (RTT) e li comunica al router tramite il *congestion header* (CH), che è un nuovo livello protocollare di 20 bytes introdotto da XCP che si colloca tra l'IP header ed il TCP header, presente in ogni pacchetto. Il router ha il compito di calcolare lo stato di congestione attraverso la differenza tra banda disponibile del link e rate del traffico di ingresso e informare ai vari flussi, che condividono il link, di quanto devono incrementare o decrementare la loro congestion window attraverso il valore del feedback, in modo che il sistema converga alla stabilità. L'ultimo router più

congestionato del percorso, può ridurre il valore del feedback nella *congestion header* sovrascrivendolo al valore precedente. Quando il pacchetto raggiunge il receiver, il valore del feedback è rinviato al sender attraverso un pacchetto di ACK e il sender aggiornerà la sua cwnd in accordo al valore ricevuto. Il funzionamento descritto può essere riassunto nello schema seguente:

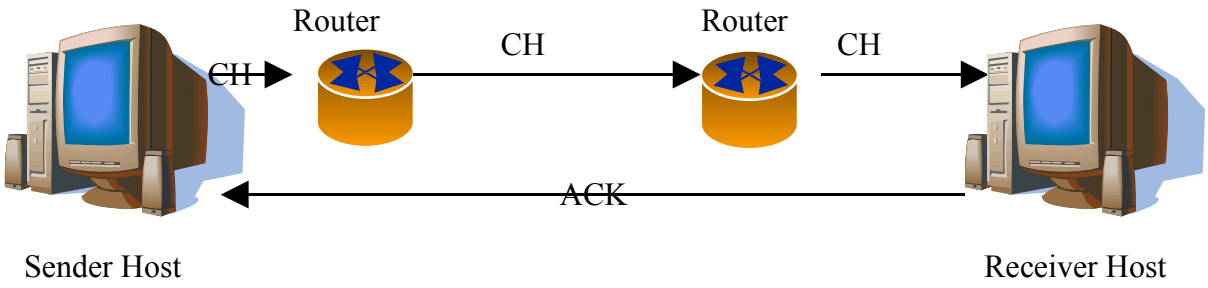

Fig. 1.1: schema elementare dell'andamento dei pacchetti

Vediamo ora la struttura della *congestion header*:

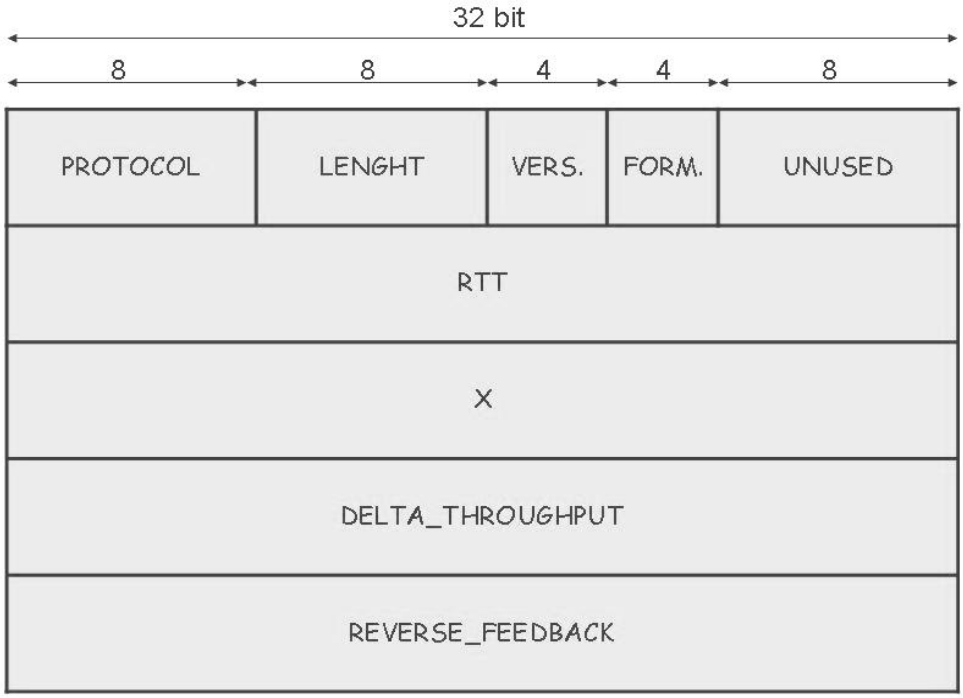

Fig.1.2 : Congestion Header

- ***Protocol (8bits):*** indica il livello protocollare successivo utilizzato nel campo dati del pacchetto. I valori dei vari protocolli sono specificati dalla IANA (Internet Assigned Numbers Autority).
- ***Lenght (8 bits):*** indica la grandezza della *congestion header* , misurata in bytes. Nel nostro caso assumerà sempre il valore di 20 bytes.

- ***Version (4 bits):*** indica la versione di XCP che si sta utilizzando. La versione di XCP descritta in questa tesi corrisponde al valore di 0x02. Valori futuri saranno assegnati dalla IANA.
- ***Format (4 bits):*** contiene un codice indicante il formato della *congestion header*. Esistono due tipi di formato: *Standard Format* e *Minimal Format.*

—

FORMAT	CODE
Standard Format	0x1
Minimal Format	0x2

Lo *Standard Format* indica che vengono utilizzati tutti i campi della *congestion header*. Generalmente questo formato è utilizzato quando il pacchetto è inviato dal sender al receiver. Il *Minimal Format* indica che i campi relativi a X, Delta_Throughput ed RTT non sono utilizzati per cui sono inizializzati a zero. Questo formato viene utilizzato quando il pacchetto è inviato all'indietro dal receiver al sender per comunicare la situazione della rete.

- ***Unused (8 bits):*** questa sezione non è utilizzata e per questo viene settata a zero.
- ***RTT (32 bits):*** indica il round – trip time calcolato dal sender misurato in secondi (il valore 1 equivale a 2^{-28} secondi). Il valore minimo possibile è zero che vuol dire che il sender non conosce il valore del round - trip time. Il valore massimo che può essere assunto è 15.9999999963 secondi, al passo di 3.7ns.
- ***X (32 bits):*** indica il tempo che intercorre tra un pacchetto e l'altro ($\frac{RTT}{cwnd}$) e viene calcolato dal sender. E' utilizzato lo stesso formato del campo RTT.

- ***Delta_Throughput (32 bits):*** indica il throughput desiderato o quello corrente. Inizialmente è settato dal sender il cui valore indica di quanto esso vuole aumentare il suo throughput; può essere successivamente ridotto dai routers lungo il percorso. E' misurato in bytes al secondo e può essere anche negativo. Il valore minimo che può assumere è -17 Gbps, mentre quello massimo è 17 Gbps, in intervalli di 1 byte al secondo.

- ***Reverse_Feedback (32 bits):*** indica il valore del *Delta_Throughput* ricevuto dal receiver. Il receiver copia questo valore nel campo Reverse_Feedback del pacchetto di ACK da inviare al sender.

1.3 Il Sender

Il sender è un computer che trasmette informazioni sulla rete. Nel nostro caso esso deve adeguarsi al protocollo in questione cercando di non inviare una quantità di informazioni maggiore a quella permessa dal router XCP. Inoltre il sender deve mantenere cinque parametri: il valore del throughput desiderato, la stima del throughput corrente, il valore massimo del throughput permesso dal protocollo XCP, la stima del tempo di interarrivo dei pacchetti (**X**) ed infine la stima del round - trip time corrente. Per quanto riguarda il throughput desiderato, il sender deve scegliere un valore sensato, come ad esempio la velocità dell'interfaccia locale o il valore del massimo throughput assegnato da API. Analizziamo ora come il sender riempie i vari campi della *congestion header*:

1) Prima di tutto setta il campo relativo all'RTT tramite una stima approssimata del round – trip time o inserendo il valore zero se questa stima non può essere effettuata.
2) Successivamente riempie il campo X relativo alla stima del valore corrente del tempo di interarrivo dei pacchetti oppure inserisce zero se la stima non è disponibile. I pacchetti in cui hanno X=0 possono ricevere solo un feedback negativo. Come visto prima, il valore di X può essere stimato come il rapporto tra l'RTT e la grandezza della congestion window. In alternativa, può essere calcolato come il rapporto tra la grandezza dei pacchetti ed il throughput stimato.
3) Come terza operazione, il sender deve calcolare il cambiamento desiderato del throughput (in genere è un incremento). Questo viene calcolato come la differenza tra il valore del throughput desiderato e quello del throughout corrente. Tuttavia, se il sender non ha abbastanza informazioni per poter inviare la stima del throughput corrente, il valore del throughput desiderato dovrebbe essere zero.
4) Ora viene diviso il valore del cambiamento del throughput con il numero di pacchetti per ogni round – trip time ed il risultato viene collocato nel campo *Delta_Throughput* della *congestion header*. Questa distribuzione per pacchetto del cambiamento del throughput è necessaria perché il router XCP non

mantiene lo stato di congestione per flusso; esso deve trattare ogni pacchetto indipendente dagli altri dello stesso flusso. Il numero di pacchetti per un RTT può essere stimato come il prodotto tra il throughput corrente e l'RTT, diviso la grandezza massima di un frame (Maximun Segment Size – MSS).

Il *Delta_Throughput* (in bytes/secondi) è calcolato nel seguente modo:

$$Delta_{Throughput} = \frac{throughput_{desiderato} - throughput}{throughput \cdot RTT - MSS}$$

dove:

- ***throughput_desiderato*** è misurato in bytes/secondo;
- ***throughput*** è misurato in bytes/secondo;
- ***RTT*** è misurato in secondi;
- ***MSS*** è misurato in bytes.

Comunque, come detto prima, *Delta_Throughput* dovrebbe essere settato a zero se non c'è necessità di aumentare il throughput poiché c'è insufficienza di informazioni per sostenere il throughput per il successivo RTT. Un problema da esaminare e studiare in futuro è come trattare il caso in cui *Delta_Throughput* è minore di 1Bps. Dato che nella *congestion header* si ha una rappresentazione di interi, in questo caso verrà impostato zero come valore di *Delta_Throughput*. In realtà si vorrebbe inviare una frazione dei pacchetti in un round trip time con il valore di *Delta_Throughput* diverso da zero.

5) Per quanto riguarda il TCP, la stima del throughput può essere ottenuta, per esempio, dividendo la congestion window (cwnd, in bytes) con l'RTT. In realtà il rapporto cwnd/RTT differisce dal vero throughput in due aspetti. Per prima cosa, la cwnd non tiene conto della grandezza dell'header. Questo diventa significante per applicazioni di XCP in flussi real – time che inviano una grande quantità di piccoli pacchetti, ma probabilmente non è molto preoccupante per flussi TCP che tendono a inviare pacchetti di grandi dimensioni. In secondo luogo, la cwnd rappresenta il permesso del sender per inviare dati. Se l'applicazione non considera tutte le cwnd disponibili, il throughput annunciato sarà più grande del throughput reale. In un router XCP questo può provocare la nascita di allocazione di feedback negativo non esatto.

1.3.1 Reazione del sender alla ricezione del feedback

L'informazione di Delta_Throughput ritorna al sender nel campo reverse_feedback contenuto nel messaggio di ACK in maniera tale che il sender regoli la sua congestion window (cwnd) adattandola alla nuova banda assegnata. Visto che il receiver può inviare pochi messaggi di ACK, esso somma tutti i Delta_Throughput dei pacchetti ricevuti in un intervallo di tempo e invia il valore ottenuto attraverso il successivo messaggio di ACK nel campo reverse_feedback. La formula utilizzata dal sender per regolare la cwnd è la seguente:

$$w = \max (w + \text{feedback} \cdot RTT, MSS)$$

dove:

- **w** è la cwnd corrente di TCP, misurata in bytes.
- **feedback** è il valore riportato nel campo reverse_feedback del pacchetto di ACK; è calcolato in bytes/sec e può essere sia positivo che negativo.
- **RTT** è il round trip time corrente calcolato dal sender in secondi.
- **MSS** è la minima grandezza di un segmento (frame) calcolata in bytes.

Il valore minimo della cwnd è impostato a MSS in modo da evitare la "silly window syndrome" riportato in [RFC0813]. Quest'ultima si presenta quando la minima cwnd diventa troppo piccola per permettere al TCP di inviare dati. Per prevenire che il valore della cwnd sia 0, essa viene settata per essere almeno grande quanto la grandezza minima di un pacchetto.

1.3.2 Reazione alla perdita di pacchetti

Quando il protocollo di trasporto è TCP, un pacchetto perso o la verifica di una situazione di Explicit Congestion Notification (ECN) dovrebbe causare una transizione ad un comportamento standard del controllo di congestione come riportato nel documento [RFC2581]. In altre parole, la cwnd dovrebbe essere dimezzata e vengono applicati gli algoritmi di *fast retrasmission/fast recovery*, *slow start* e *congestion avoidance,* descritti da Jacobson nel documento [Jacobson88], per il resto della connessione o fin quando l'evento di congestione è terminato. In XCP viene assunto che la perdita di pacchetti rivela la presenza di un router non – XCP lungo il percorso.

Notiamo che:

- Il cambiamento dell'algoritmo di controllo di congestione da XCP a TCP, deve essere rinviato fino alla ricezione di tre ACK duplicati (tre DUPACK) in accordo all'algoritmo di *fast retrasmission/fast recovery.*

- Una volta che il controllo di congestione passa al TCP standard, la cwnd dovrebbe essere gestita utilizzando l'algoritmo descritto nel documento [RFC2581].
- Il campo X dei pacchetti in volo dovrebbe continuare a riportare il tempo attuale che intercorre tra un pacchetto e l'altro. Questo permette, alle applicazioni XCP nel router lungo il percorso, di continuare a controllare l'utilizzo della capacità del flusso.
- Saranno necessari ulteriori studi per determinare se sia possibile ritornare ad una connessione a controllo di congestione XCP, una volta che si sono terminate le transizioni degli algoritmi di Van Jacobson.
- Per protocolli di trasporto diversi da TCP, la risposta a pacchetti persi o alla notifica di ECN è ancora in fase di studio.

1.3.3 Stima dell'RTT

Avere una buona stima del round trip time (RTT) è più importante in XCP rispetto ai controlli di congestione spiegati da Van Jacobson. E' evidente che piccoli errori della stima del RTT influiscono di molto sulla stima del throughput e di X. Il rapporto tra cwnd corrente e RTT è solo un'approssimazione del throughput effettivo. In modo analogo, il rapporto tra RTT e cwnd è solo un'approssimazione del tempo, altamente variabile, che intercorre tra un pacchetto e l'altro (X).

1.4 Il Receiver

Quando il sistema di ricezione (receiver) riceve un messaggio XCP, copia il valore del campo Delta_Throughput nel campo riverse_feedback del messaggio di ritorno (ACK). Settando il formato del pacchetto a 0x2 (formato minimo), il receiver impedisce al router XCP di modificare il pacchetto di ritorno destinato al sender. Come detto prima, il receiver dovrebbe inviare il messaggio di ritorno XCP sul messaggio di ACK del TCP, per ogni messaggio ricevuto. Spesso, però, l'implementazione di TCP è tale che non viene inviato un ACK per ogni messaggio ricevuto, ma vengono inviati degli ACK cumulativi. Per adattarsi a questa situazione, il receiver deve accumulare tutti i Delta_Throughput che ha ricevuto in modo da inviare al sender il feedback corretto. Il sender, in questo caso, riceverà pochi messaggi XCP, ma ognuno conterrà il valore del feedback cumulativo dei messaggi XCP multipli inviati.

1.5 Il router XCP

Il compito di un router XCP è calcolare il feedback per permettere al sistema di convergere ad un'ottima efficienza e fairness. In questo modo, XCP non dovrebbe perdere pacchetti. Infatti l'obiettivo di XCP è prevenire, il più possibile, l'incremento della coda di attesa nel momento in cui il pacchetto è stato perso.

Per calcolare il feedback, un router XCP utilizza un controllore per l'efficienza (*efficiency controller*) ed un controllore per la distribuzione fair di banda (*fairness controller).* Entrambi i controllori stimano l'RTT medio dei flussi che attraversano il link, mitigando la burstiness del protocollo di controllo window – based. Infatti, la stima dei parametri su intervalli di tempo più lunghi rispetto all'RTT medio, porta ad una risposta del sistema abbastanza lenta, mentre la stima dei parametri su intervalli di tempo più piccoli porta a stime errate. L'RTT medio è calcolato utilizzando le informazioni della *congestion header.*

I controllori di XCP effettuano singole decisioni di controllo in un RTT medio (l'intervallo di controllo). Questo è dovuto al fatto che c'è la necessità di esaminare i risultati delle precedenti decisioni di controllo prima di effettuare un nuovo controllo. Per esempio, se il router vuole informare la sorgente di incrementare la congestion window, dovrebbe prima accertarsi di quanta banda è disponibile e poi inviare al sender la quantità di incremento.

Il router mantiene per un determinato tempo la stima di controllo per-link, il quale è la più recente stima dell'RTT medio sul link interessato. Allo scadere del tempo di osservazione, il router aggiorna le sue stime e le decisioni di controllo. Vediamo di seguito come avvengono le decisioni di controllo di efficienza e di fairness di banda.

1.5.1 Il Controllore di Efficienza (Efficiency Controller – EC)

Lo scopo del controllore di efficienza è quello di massimizzare l'utilizzo del link di comunicazione minimizzando, allo stesso tempo, il rate di pacchetti persi e l'occupazione della coda persistente. Il controllore osserva solo un traffico aggregato senza preoccuparsi del fairness.

Poiché XCP è window – based, l'EC calcola gli incrementi o decrementi desiderati in numero di bytes che il traffico totale trasmette in un intervallo di controllo (RTT medio). Il feedback totale è calcolato, quindi, in ogni intervallo di controllo come segue:

$$\Phi = \alpha \cdot d \cdot S - \beta \cdot Q \quad ,$$

dove **α** e **β** sono parametri costanti, i quali valori sono rispettivamente 0.4 e 0.226, settati basandosi sull'analisi di stabilità che verrà di seguito descritta [Katabi03]. Il termine **d** è l'RTT medio, **S** indica la quantità di banda disponibile definita dalla differenza tra l'input rate e la capacità del link di comunicazione (notiamo che **S** può essere anche negativa), ed

infine, **Q** rappresenta la grandezza della coda persistente (coda che non si svuota in un ritardo di propagazione round trip), diversa dalla coda transitoria che è il risultato della natura bursty di tutti i protocolli window – based. Il calcolo di **Q** avviene prendendo in considerazione la minima coda vista dall'arrivo di un pacchetto durante l'ultimo ritardo di propagazione.

Dall'equazione di **Φ** su scritta, otteniamo il feedback proporzionale alla quantità di banda disponibile poiché, quando $\mathbf{S} \geq 0$, il link di comunicazione è sottoutilizzato e quindi si invierà un feedback positivo, mentre quando $\mathbf{S} < 0$, il link è congestionato e si invierà un feedback negativo. Comunque, l'utilizzo di questa politica è insufficiente perché significherebbe non dare feedback quando il traffico in input eguaglia la capacità del link, e in questo modo la coda non si svuoterebbe. Per svuotare la coda persistente, viene generato un feedback totale proporzionale alla coda persistente in eccesso. Infine, visto che il feedback è in byte/sec, quando arriverà al sender, verrà prima moltiplicato per l'RTT corrente relativo al sender e poi sommato alla cwnd.

Per ottenere efficienza, viene allocato sul singolo pacchetto il feedback totale nel campo *reverse_feedback* della *congestion_header*. Il fatto che l'EC (efficiency controller) tratta solo traffico totale significa che non si preoccupa se i pacchetti portano feedback e di quanto cambia la congestion window di ogni singolo flusso. Il controllore di efficienza impone che, i cambiamenti del traffico totale dovuti a **Φ**, avvengano nell'intervallo di controllo. La suddivisione del feedback tra i vari pacchetti avviene attraverso il controllore fairness (*fairness controller)*.

1.5.2 Il Controllore Fairness (Fairness Controller – FC)

Il compito del controllore fairness (FC) è ripartire il feedback su pacchetti singoli in modo da raggiungere l'esatta distribuzione di banda. L'FC si basa sullo stesso principio usato da TCP per convergere all'imparzialità, cioè sulla politica denominata *Additive Increase – Moltiplicative Decrease (AIMD)* in cui si ha un' incremento additivo, e un decremento moltiplicativo della cwnd. In questo modo si può calcolare il feedback per – pacchetto in accordo alla seguente teoria:

- ✓ se $\mathbf{\Phi} > 0$, l' incremento del throughput di tutti i flussi è lo stesso.

- ✓ se **Φ** < 0, il decremento del throughput di un flusso è proporzionale al suo throughput corrente.

Questo assicura continua convergenza all'imparzialità finché il feedback totale non è zero. Per prevenire una situazione di stallo della convergenza quando l'efficienza è circa ottimale (**Φ** ≈ 0), viene introdotto il concetto di *bandwidth shuffling*. Questa è l'allocazione e il decremento simultaneo della larghezza di banda tale che il rate del traffico totale (e di conseguenza l'efficienza) non cambi, anche se il throughput di ogni flusso cambia gradualmente per ottenere un flusso condiviso fair. Lo *shuffled traffic* è calcolato come segue:

$$h = \max(0, \kappa \cdot y - |\Phi|) ,$$

dove ***y*** è il traffico di input in un RTT medio e ***k*** è una costante settata a 0.1. Questa equazione assicura che, ad ogni RTT medio, almeno il 10% del traffico è ridistribuito in accordo alla politica AIMD.

Il calcolo del feedback per – pacchetto, che permette al *fairness contoller* di applicare la politica suddetta, è effettuato nel modo seguente in riferimento al documento [Katabi03]. Dalla legge dell'incremento additivo e del decremento moltiplicativo, è conveniente calcolare il feedback assegnato ad ogni pacchetto ***i*** come combinazione del feedback positivo p_i , e di quello negativo n_i :

$$reverse_{feedback} = p_i - n_i .$$

Consideriamo, per prima cosa, il caso in cui il feedback è positivo (**Φ**>0). In questo caso si dovrà incrementare il throughput di tutti i flussi allo stesso modo. Quindi, il cambio del throughput di alcuni flussi ***i*** è proporzionale alla stessa costante ($\frac{\Delta throughput_i \propto ¿}{¿}$ cost).

Visto che il protocollo in questione è window – based, viene considerato il cambiamento della cwnd piuttosto che quello del throughput e quindi l'evoluzione della cwnd del flusso ***i*** deve essere proporzionale all'RTT del flusso ($\Delta cwnd_i \propto rtt_i$).

Il passo successivo è tradurre questo cambiamento richiesto della cwnd in feedback per – pacchetto che verrà riportato nel *congestion header*. Il cambiamento complessivo della cwnd di un flusso è la somma dei feedback ricevuti per pacchetto. Di conseguenza, il feedback per pacchetto si ottiene dividendo il cambiamento della cwnd con il numero previsto di pacchetti del flusso ***i*** che il router riceve in un intervallo di controllo **d**. Questo valore è proporzionale alla cwnd del flusso diviso la grandezza di un singolo pacchetto ($\frac{cwnd_i}{s_i}$, numeratore e denominatore in bytes), e inversamente proporzionale al suo round

trip time, **rtt$_i$**. Quindi, in conclusione, il feedback positivo per – pacchetto è proporzionale al quadrato dell'RTT del flusso, e inversamente proporzionale al rapporto tra cwnd e la grandezza del singolo pacchetto ($p_i \propto \frac{rtt_i^2}{cwnd_i / s_i}$). Il feedback positivo $\boldsymbol{p_i}$ sarà:

$$p_i = \varepsilon_p \frac{rtt_i \cdot s_i}{cwnd_i} ,$$

dove ε_p è una costante. L'incremento totale del rate del traffico complessivo è $\frac{h + \max(\Phi, 0)}{d}$, dove $\max(\Phi, 0)$ assicura che si è calcolato il feedback positivo. Questa quantità è uguale alla somma dell'incremento dei rates di tutti i flussi in complessivo, che è la sommatoria del feedback positivo di ogni flusso diviso il suo RTT :

$$\frac{h + \max(\Phi, 0)}{d} = \sum^{L} \frac{p_i}{rtt_i} ,$$

dove $\boldsymbol{L}$ è il numero di pacchetti ricevuti dal router in un RTT medio. Da questa eguaglianza, può essere ricavato ε_p come segue:

$$\varepsilon_p = \frac{h + \max(\Phi, 0)}{d \cdot \sum \frac{rtt_i \cdot s_i}{cwnd_i}} .$$

In modo analogo si può calcolare il feedback negativo per - pacchetto $\boldsymbol{n_i}$, ottenuto quando il feedback cumulativo è negativo ($\boldsymbol{\Phi}$<0). In questo caso, si vorrà decrementare il throughput del flusso ***i*** proporzionalmente al suo throughput corrente ($\Delta throughput_i \propto throughput_i$). Di conseguenza, la variazione desiderata della congestion window del flusso è proporzionale alla congestion window corrente ($\Delta cwnd_i \propto cwnd_i$). D'altronde, il feedback desiderato per pacchetto è praticamente la variazione della congestion window diviso il numero di pacchetti inviati dal flusso che il router riceve in un intervallo di controllo **d**. Quindi si può concludere che, il feedback negativo per – pacchetto è proporzionale al prodotto tra la grandezza del singolo pacchetto è l'RTT del flusso ***i***-esimo ($n_i \propto rtt_i \cdot s_i$). Il feedback negativo $\boldsymbol{n_i}$ è:

$$n_i = \varepsilon_n \cdot rtt_i \cdot s_i ,$$

dove ε_n è una costante. Come per il caso dell'incremento, il decremento totale in rate di traffico cumulativo, è la somma dei decrementi in rate di tutti i flussi complessivi come segue:

$$\frac{h+\max(-\Phi,0)}{d}=\sum^{L}\frac{n_i}{rtt_i}\ .$$

Da questa uguaglianza, si può derivare la costante ε_n come:

$$\varepsilon_n=\frac{h+\max(-\Phi,0)}{d\cdot\sum s_i}\ ,$$

dove la sommatoria di s_i indica la somma di tutti i pacchetti in un intervallo di controllo (cioè RTT medio).

1.5.3 Note sui controllori di Efficienza e di Fairness

Vediamo ora alcuni aspetti importanti sul design dei controllori di efficienza e di imparzialità. Come detto prima, i due controllori sono separati. In particolare, il controllore di efficienza utilizza un politica *Moltiplicative – Increase Moltiplicative – Decrease* (MIMD), in cui si incrementa il rate del traffico proporzionalmente alla quantità di grandezza di banda disponibile nel sistema (piuttosto che incrementare di un pacchetto ogni RTT per ogni flusso come nel TCP). Questa legge di controllo permette all'XCP di utilizzare più velocemente la quantità di banda disponibile, sfruttando esattamente la capacità del link. Il controllore di imparzialità utilizza, invece, la legge di controllo *Additive – Increase Moltiplicative – Decrease* (AIMD), in modo da convergere all'imparzialità. Quindi, la separazione permette ad ognuno dei controllori di utilizzare la legge di controllo più opportuna.

Si può notare, però, che le leggi di controllo utilizzate dai due controllori non sono le uniche scelte possibili. Per esempio, nel documento[Katabi01] viene descritto un controllore di imparzialità che utilizza una legge binomiale simile alla legge descritta nel documento [Bansal01]. Comunque, in questa tesi sono utilizzate le leggi di controllo utilizzate da Katabi poiché dall'analisi effettuata in [Katabi03] risultano avere migliori prestazioni.

Notiamo che il controllore di efficienza soddisfa le richieste descritte nel paragrafo 1.1. La dinamica del traffico totale è specificata dal feedback cumulativo e rimane indipendente dal numero di flussi che attraversano il link. In aggiunta, in contrasto con il TCP dove la decisione dell'incremento/decremento è indifferente al grado di congestione della rete, il feedback cumulativo inviato dal controllore di efficienza è proporzionale al grado di utilizzo della capacità del link (sotto – utilizzo, sovra – utilizzo). Inoltre, dato che il feedback cumulativo è calcolato su un' RTT medio, XCP diventa poco reattivo al crescere del round trip delay.

Sebbene il controllore di imparzialità utilizzi una legge AIMD, la convergenza all'imparzialità di banda è più veloce rispetto al TCP. Notiamo che utilizzando l'AIMD, tutti i flussi incrementano il loro throughput ugualmente indipendentemente dal loro rate corrente. Di conseguenza, è la politica moltiplicative decrease che aiuta a convergere all'imparzialità. In TCP, la legge moltiplicative decrease (MD) è dipendente alla notifica di perdita. In contrasto, in XCP la MD no dipende dalle perdite ed è effettuata ogni RTT medio.

XCP è abbastanza robusto per quanto riguarda la stima degli errori. Per esempio, è stato stimato il valore di ε_p ogni **d** ed è stato usato durante il successivo intervallo di controllo (cioè il successivo **d**). Se si sottostimasse il valore di [illegible], non si riuscirebbe ad allocare abbastanza feedback positivo nell'intervallo di controllo corrente. Inoltre, non si riesce ad allocare banda sulla successiva stima del traffico di input relativo alla banda disponibile, che sarà allocata (o parzialmente allocata) nel successivo intervallo di controllo. Quindi, in ogni intervallo di controllo, è allocata una porzione di banda disponibile fin quando non ci sarà più disponibilità di banda. La sottostima di [illegible] causa un'allocazione ridotta e la convergenza all'efficienza è più lenta rispetto alla situazione in cui [illegible] viene stimato correttamente. Nonostante ci sia errore, XCP raggiungerà comunque la situazione in cui sfrutta la capacità massima del link. Allo stesso modo, se si sovrastimasse [illegible] si allocherebbe più feedback per flusso al variare dell'intervallo di controllo esaurendo il feedback cumulativo velocemente. Questo spreco di feedback sull'intervallo di allocazione non influisce sull'utilizzo di banda, ma rallenta la convergenza alla fairness. Si possono effettuare simili considerazioni su altre stime di errori; comunque si avrebbe effetto sul tempo di convergenza ma non sulla sua esattezza.

I parametri costanti utilizzati in XCP (cioè **α** e **β)** sono scelti indipendentemente dal numero di sorgenti, dal ritardo e dalla capacità del link condiviso. Questo è un miglioramento importante per quanto riguarda precedenti studi dove valori specifici di alcuni parametri dipendevano dalla situazione in cui si trovava il sistema, o meglio la scelta dei parametri dipendeva dal numero di sorgenti, dalla capacità e dal ritardo. Nel capitolo seguente vedremo con quali criteri sono stati scelti i valori delle costanti **α** e **β** e come è composto il codice che implementa i controllori di efficienza e di imparzialità riferendosi al documento [draft – falk – spec – 01].