Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

MODELLAZIONE E COMPARAZIONE DEL PROTOCOLLO TCP IN SCENARI DI MOBILITÀ

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentata da:
Stefano Pierantozzi

Sessione II
Anno Accademico 2013/2014

A tutti coloro che vorrebbero esserci!

Indice

Elenco delle figure

Introduzione

I progressi scientifici durante l'ultimo secolo hanno provocato un cambiamento forte e radicale nel modo di comunicare. In particolare l'impatto delle tecnologie wireless ha rivoluzionato le comunicazioni lasciando il segno nella vita quotidiana; ciò è confermato anche dall'evidente crescita mostrata nel mercato mondiale, specialmente in ambito della telefonia e dei dispositivi mobili. Tuttavia l'utilizzo di connessioni senza fili e l'idea di mobilità introducono, oltre ad una serie enorme di benefici e innovazioni, anche diversi problemi che dovrebbero essere presi in considerazione per far sì che le funzionalità ed i vantaggi offerti da questa tecnologia si integrino completamente con i sistemi esistenti. Uno tra i protocolli da sempre più usato e conosciuto in ambito di comunicazioni di rete, il TCP, viene sempre più spesso, implicitamente o esplicitamente, utilizzato all'interno di sistemi che prevedono almeno un tratto di rete wireless, poiché offre diverse caratteristiche desiderabili, come tecniche di controllo ed affidabilità, che risultano molto preziose se si considera la difficoltà di ottenerle durante le comunicazioni wireless. Eppure, le versioni di TCP che venivano solitamente utilizzate non prevedevano supporto specifico per questo tipo di comunicazione, che aggiunge diverse problematiche e nuovi requisiti rispetto alle connessioni cablate per cui inizialmente tale protocollo fu progettato. L'adozione perciò di reti wireless che sfruttassero efficientemente il TCP è stata nell'ultimo decennio, e continua ad essere tuttora, materia di forte dibattito e ricerca. In particolare si è cercato di raffinare le versioni esistenti di TCP introducendo caratteristiche orientate al miglioramento delle comunicazioni senza fili, introducendo modi-

fiche che permettessero da un lato di compensare gli svantaggi derivanti dal wireless e dall'altro di mantenere il TCP uno standard adatto ai sistemi per cui era stato da sempre utilizzato. In questo lavoro di tesi verrà considerato uno scenario tipico di rete di comunicazione basata su TCP, in cui un tratto wireless sarà alternato ad uno wired al fine di far comunicare due end-system costituiti da un nodo mobile ed un host fisso. Per studiare ed analizzare i comportamenti tipici di un sistema siffatto, si ricorrerà a metodi matematici consolidati in letteratura, con cui si cattureranno gli aspetti più rilevanti che emergono dall'utilizzo del TCP in contesti di mobilità. Nello specifico verranno analizzate due varianti di sistema derivanti dallo stesso contesto: un primo sistema in cui si utilizzerà una versione TCP classica, ed un secondo che prenderà in esame una versione di TCP modificata con un meccanismo di ritrasmissione anticipata dei pacchetti persi. I modelli matematici esistenti, avvalendosi del metodo basato sulla Teoria del Rinnovamento, saranno lo strumento chiave per delineare e comprendere i comportamenti dei sistemi osservati, per definirli in termini statistici e per confrontarli, con lo scopo di valutarne prestazioni e caratteristiche principali oltre all'impatto che possano avere modifiche più o meno complesse al sistema. Si vedrà come la modellazione matematica riesca ad offrire degli strumenti potentissimi ma allo stesso tempo estremamente difficili da governare ed utilizzare praticamente se non se ne ha la completa padronanza e non ci si serve di condizioni ed ipotesi di semplificazione della realtà che rispettino svariati vincoli di validità. Inoltre, risulterà chiaro come nonostante i meccanismi del TCP combinati a quelli wireless offrano servizi migliori, maggiormente se opportunamente modificati e interconnessi, piccoli cambiamenti nelle logiche di funzionamento e nei parametri interni ed esterni del TCP influenzino pesantemente le dinamiche di comportamento, e di conseguenza le prestazioni del sistema.

L'elaborato è suddiviso in capitoli, che seguono grossomodo una linea temporale degli step d'avanzamento per soddisfare gli obiettivi prefissati, ed è organizzato come desritto di seguito.

- **TCP:** una panoramica sul funzionamento generale del protocollo TCP

e dei principali meccanismi di controllo di cui dispone, seguita da una descrizione delle versioni più conosciute ed utilizzate dalla nascita ai tempi odierni.

- **OBIETTIVI E SCENARIO:** una breve precisazione degli obiettivi primari di questo lavoro, contestualizzati nello scenario proposto per l'analisi dei sistemi; inoltre verrà mostrata ed analizzata l'architettura globale di un sistema classico di comunicazione che utilizza uno tra gli standard wireless più conosciuti , il Wi-Fi, brevemente descritto.
- **SISTEMA ANTICIPATO:** una descrizione delle principali idee e caratteristiche su cui si fonda la modifica del sistema introdotto precedentemente, affiancata da una breve spiegazione del suo funzionamento generale.
- **STATO DELL'ARTE:** un sguardo approfondito nel mondo della modellazione matematica incentrata su sistemi wired e wireless che utilizzano il TCP, in cui si analizzano una serie di approcci rilevanti al nostro scopo e svariati lavori presenti in letteratura.
- **VINCOLI E ASSUNZIONI:** un sottoinsieme dei maggiori problemi affrontati e delle decisioni prese durante lo svolgimento della tesi, che hanno portato all'imbocco di determinate strade piuttosto che altre ed all'adozione di assunzioni necessarie per raggiungere i risultati desiderati.
- **IL MODELLO:** una spiegazione dettagliata del modello utilizzato per l'analisi del sistema classico ed in secondo luogo un confronto con il sistema anticipato di cui si discuteranno le potenzialità, i problemi ed i casi specifici di funzionamento, delineando le linee guida per la rielaborazione e l'adattamento del modello classico.
- **SVILUPPI FUTURI:** una lista contenuta di suggerimenti, potenziali modifiche e aggiunte che potrebbero servire come spunto di possibile evoluzione di questo lavoro di tesi.

Capitolo 1

TCP

Il primo capitolo introduce uno tra i protocolli di trasporto più conosciuti e comunemente usati su Internet: il Transmission Control Protocol. Le successive sezioni delineano il regime operativo e gli aspetti fondamentali comuni alle varie versioni di TCP, discutendo successivamente le principali differenze che contraddistinguono alcune tra le versioni più note, riconducibili essenzialmente alle due generazioni di TCP Tahoe e Reno, e delineando i cambiamenti sostanziali nel tempo.

1.1 Introduzione al TCP

Negli anni 90 Internet rivoluzionò il modo di distribuire le informazioni nel mondo, entrando gradualmente a far parte della vita quotidiana di persone private e compagnie commerciali. Di conseguenza, attualmente è richiesto un alto grado di affidabilità di Internet, spesso accompagnato da requisiti di disponibilità delle risorse, sicurezza ed ottimizzazione delle prestazioni. Tuttavia Internet è composta da sottoreti eterogenee, strutturalmente diverse ed appartenenti a proprietari differenti, inoltre non esiste una singola autorità che abbia responsabilità per l'intera rete. L'utilizzo così vasto e vario di Internet genera dunque numerosi problemi di gestione, tra cui la suddivisione della capacità della rete in maniera equa tra milioni di utenti. I nodi della

rete devono perciò comunicare sfruttando in maniera intelligente il canale di comunicazione e scegliendo ponderatamente i rispettivi tassi di invio.

Il Transmission (o Transfer) Control Protocol (TCP) è un protocollo di rete (livello di trasporto) con la caratteristica chiave di fornire il controllo di trasmissione, offrendo garanzie sulla consegna dei pacchetti trasmessi, affiancati da meccanismi di controllo del flusso e della congestione di rete, al fine di risolvere molti dei problemi suddetti e costruire un canale di comunicazione affidabile tra i nodi della rete. E' un protocollo connection-oriented, cioè prima di poter trasmettere è necessario stabilire una connessione, detta *Handshake*, tra mittente e destinatario che rimane aperta anche in assenza di comunicazione fino ad esplicita segnalazione di chiusura. Il servizio offerto è full-duplex, quindi un flusso di byte bidirezionale tra due applicazioni in esecuzione su host differenti che permette la trasmissione contemporanea in entrambe le direzioni. Inoltre, il protocollo è stream-oriented, ovvero i dati trasmessi sono considerati come un flusso di byte ordinati e numerati. Il TCP non supporta la comunicazione multicast, che consiste nel trasferimento di dati da un mittente a destinatari multipli, in una sola operazione; nel caso un host debba comunicare con altri due o più host, sarà necessario instaurare una connessione TCP separata per ogni destinatario.

1.2 Aspetti fondamentali

In generale, le caratteristiche principali del TCP sono le seguenti e vengono descritte singolarmente nelle sezioni successive:

- **Segmento TCP:** struttura che rappresenta un pacchetto TCP formato da un' intestazione e una sezione dati.
- **Handshake:** algoritmo di apertura e chiusura della connessione.
- **Affidabilità e correttezza nella trasmissione:** tutti i dati trasmessi dovrebbero essere consegnati al destinatario in maniera affidabile, cioè permettendo, grazie ad appositi meccanismi, di rilevare gli errori e le

perdite, ritrasmettendo i pacchetti persi. Inoltre, a causa di congestionamento, path differenti, ed altro ancora, alcuni pacchetti potrebbero arrivare in un ordine diverso da quello stabilito alla partenza; prima di essere inoltrati a livello superiore i pacchetti ricevuti devono essere riordinati in base al numero di sequenza contenuto nell'header TCP.

- **Controllo del flusso:** l'affidabilità della comunicazione in TCP è garantita anche dal cosiddetto controllo di flusso atto a far in modo che il flusso di dati in trasmissione non superi le capacità di ricezione del ricevente, con conseguente overflow del buffer ed eliminazione dei pacchetti. Ciò può accadere nel caso in cui il mittente invii i dati con una frequenza nettamente maggiore del destinatario. Per colmare questo scompenso, il TCP calibra il tasso di invio del mittente in modo da bilanciare le prestazioni.

- **Controllo della congestione:** per garantire ulteriormente l'affidabilità, si cerca di limitare il più possibile fenomeni di congestione nella rete analizzando le informazioni disponibili relative allo stato del canale e anche in questo caso cercando di regolare dinamicamente il tasso di invio del mittente tramite una finestra di congestione.

- **Multiplazione delle connessioni:** il TCP si occupa di smistare i dati tra le connessioni attive ed i relativi processi, associando ad ogni connessione tra due host un numero di porta (per ciascun host) che, affiancato agli indirizzi IP, la identifichino. In questo modo, un server può accettare connessioni da più client contemporaneamente attraverso una o più porte, così come un client può stabilire più connessioni verso più destinazioni, ed è anche possibile che un client stabilisca contemporaneamente più connessioni indipendenti verso la stessa porta dello stesso server.

1.2.1 Segmento TCP

Il pacchetto TCP è detto segmento, viene normalmente incapsulato in un pacchetto IP, ed è costituito da un header ed un payload (carico utile), ovvero dati di livello applicativo. I dati contenuti nell'intestazione costituiscono l'insieme di informazioni necessarie per la comunicazione tra le due entità TCP, sono utilizzate quindi per realizzare le funzionalità dello strato di trasporto e non sono accessibili dagli strati dei livelli superiori. La struttura di un segmento TCP è mostrata nella Figura 1.1.

Bit offset	Bits 0–3	4–7	8–15								16–31
0	Source port [16 bit]										Destination port [16 bit]
32	Sequence number										
64	Acknowledgment number										
96	Data offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size
128	Checksum										Urgent pointer
160	Options (optional)										
160/192+	Data										

Figura 1.1: struttura di un pacchetto (segmento) TCP

- I campi *Source Port* (16 bit) e *Destination Port* (16 bit) memorizzano le porte di mittente e destinatario e combinati con l'indirizzo IP identificano univocamente la connessione TCP e stabiliscono il tragitto del segmento.

- *Sequence Number* (32 bit), *Acknowledgement Number* (32 bit) e *Window Size* (16 bit) concorrono a realizzare il meccanismo di Sliding Window, che permette ad ogni dispositivo di tenere traccia della quantità di dati che ha inviato e di confermare la corretta ricezione degli stessi da parte dell'altro dispositivo. Contengono rispettivamente il numero

di sequenza del primo byte trasportato in quel segmento, il numero di sequenza che il ricevente si aspetta di ricevere con il prossimo pacchetto e la quantità di dati che la destinazione può ancora ricevere.

- Il campo *Data Offset (o Header Length)* (4 bit) specifica la lunghezza totale dell'intestazione del pacchetto, che può essere variabile (da 20 a 60 bytes) considerando l'utilizzo del campo facoltativo Opzioni.

- Il campo *Reserved* (4 bit) non è utilizzato ed è previsto essere usato per sviluppi futuri del protocollo.

- Il campo *Flags* (8 bit) si occupa di trasmettere informazioni di controllo realtive alla comunicazione. Ogni bit può essere trattato indipendentemente e le rispettive funzioni sono la notifica di riduzione della finestra in caso di congestione (CWR), la conferma di ricezione di CWR (ECE), la presenza di dati urgenti (URG), la validità del campo Acknowledgement Number (AKC), la trasmissione immediata senza bufferizzazione dei dati ricevuti ai livelli superiori (PSH), il reset della connessione poiché non valida (RST), la segnalazione di apertura di una connessione (SYN) e la segnalazione di chiusura della connessione (FIN).

- Il campo *Checksum* (16 bit) verifica la validità del segmento tramite operazioni di complemento.

- Il campo *Urgent Pointer* (16 bit) è legato al flag URG e ne indica la posizione.

- Il campo *Options* (40 byte) è facoltativo e finalizzato ad utilizzi avanzati.

1.2.2 Connessione: Way-Handshake

Come detto in precedenza, prima di qualunque operazione di trasferimento e controllo sul canale di comunicazione, il TCP si occupa di instaurare la connessione tra i processi applicativi dei terminali tramite la specifica di una

coppia $< indirizzoIP, porta >$ per ognuno di essi. Tale processo è chiamato Three-Way-Handshake, ed è descritto di seguito insieme al processo analogo di terminazione della connessione detto invece Four-Way-Handshake.

Three-Way-Handshake

Come è presumibile dal nome, il processo implica la necessità di scambiare tre messaggi tra mittente e destinatario affinché la connessione sia instaurata correttamente. Supponendo quindi che il client voglia aprire una connessione con il server, ed abbreviando Sequence Number con SN e Acknowledgement Number con AN, il processo procederà nei seguenti tre step, riportati in Figura 1.2.

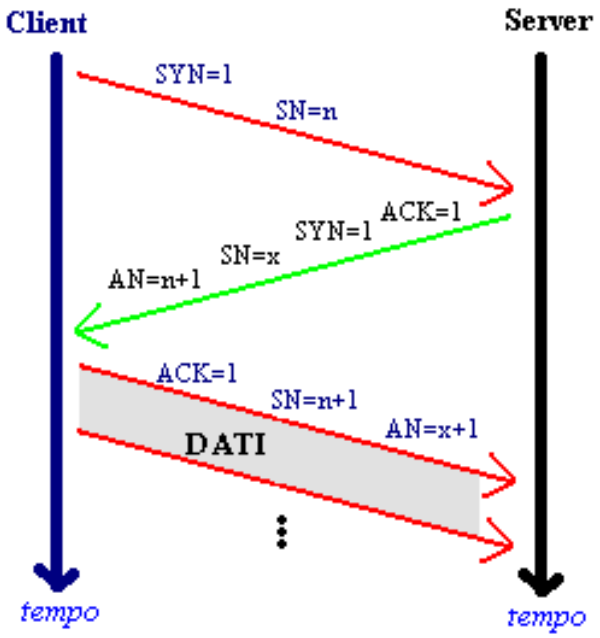

Figura 1.2: apertura connessione TCP (Three-Way-Handshake)

1. il client invia un SYN al server, o meglio un segmento senza dati in cui il flag SYN=1 e il campo SN è inizializzato ad un numero iniziale, SN=n, calcolato in maniera pseudocasuale dal client;

2. ricevuto il segmento, il server risponde al client inviando un nuovo segmento vuoto (SYN/ACK) con i flag SYN=1 ed ACK=1, il campo AN inizializzato al valore ricevuto in SN incrementato di 1, AN=n+1, e il campo SN inizializzato al numero iniziale pseudocasuale del server, SN=x;

3. dopo aver ricevuto l'ACK, il client invia un ultimo segmento vuoto (ACK) con SYN=0, ACK=1, il campo AN inizializzato al valore ricevuto in SN incrementato di 1, AN=x+1 e SN incrementato di 1, SN=n+1.

L'invio del terzo segmento da parte del client risulta non necessario ai fini dell'apertura della connessione in quanto alla ricezione dell'ACK dal server entrambe le parti sarebbero pronti per la comunicazione. Tuttavia la sua utilità deriva dal bisogno di effettuare una stima del timeout iniziale inteso come tempo intercorso tra l'invio di un segmento e la ricezione del corrispondente ACK.

Four-Way-Handshake

La connessione TCP è considerata come un aggregato di due flussi provenienti dai due partecipanti. Per terminare la connessione entrambe le parti devono terminare la propria, di conseguenza possono esistere connessioni monodirezionali. Utilizzando un procedimento analogo a quello esposto in precedenza, la connessione può essere chiusa simultaneamente con un Handshake a 3 vie che utilizza il flag FIN anziché il SYN. Per disconnessioni non contemporanee invece è necessario uno step successivo in cui la parte rimasta connessa invia un segmento in cui indica la volontà di chiudere la connessione anche dalla sua parte. L'algoritmo è mostrato nella Figura 1.3.

1.2.3 Affidabilità e correttezza nella trasmissione

Nonostante il TCP si appoggi su un protocollo come quello IP, incapace di garantire certezze in termini di affidabilità, riesce ad offrire una serie di

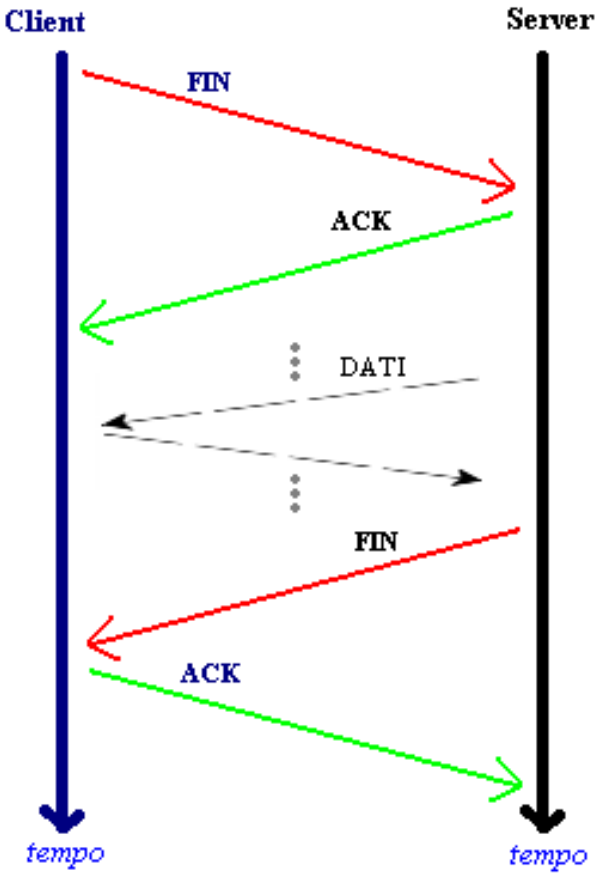

Figura 1.3: chiusura connessione TCP (Four-Way-Handshake)

servizi che lo rendono altamente affidabile. Nello specifico, i principali eventi indesiderabili legati ad un segmento TCP a cui dovrebbe essere posto rimedio sono i seguenti:

- corruzione: il segmento inviato viene ricevuto con degli errori al suo interno (uno o più bit modificati);

- ordine alterato: i segmenti potrebbero non arrivare nello stesso ordine con cui sono stati inviati a causa di percorsi diversi all'interno della rete;

- attese potenzialmente infinite: il segmento viene inviato ma viene ricevuto in tempi eccessivamente lunghi o non viene ricevuto affatto.

I provvedimenti presi dal TCP in queste circostanze sono:

- acknowledgement: quando un pacchetto è consegnato correttamente, il ricevente invia al mittente uno speciale segmento di riscontro, detto ACK (Acknowledgement), con cui lo avverte dell'avvenuta ricezione. La mancata ricezione di un ACK può causare l'invio di pacchetti duplicati o lo scadere di un timeout. Inoltre gli ACK possono essere cumulativi, cioè riscontrare un numero di pacchetti ricevuti correttamente maggiore di uno.

- numeri di sequenza e checksum: per capire se i dati sono arrivati in ordine, quali segmenti sono stati riscontrati e se ci sono errori nei pacchetti, tutti i segmenti TCP (ACK inclusi) hanno dei numeri identificativi univoci nel campo Sequence Number e funzioni di controllo nel campo Checksum.

- timer: sono dei contatori che tengono traccia del tempo che i pacchetti trascorrono nella rete. In caso di perdita o attese eccessive, un timeout impostato ad un tempo prestabilito scatta forzando il mittente ad inviare nuovamente i pacchetti.

Un esempio tipico di comunicazione durante una connessione TCP è mostrato in Figura 1.4 e verrà discusso approfonditamente nella sezione relativa alle differenti versioni del protocollo.

Timer

I diversi timer gestiti dal TCP sono di quattro tipologie:

- timer di ritrasmissione: gestisce i timeout di ritrasmissione che possono scattare tra l'invio di un pacchetto e la ricezione dell' ACK e verificano se il segmento trasmesso è stato effettivamente ricevuto. Il valore del

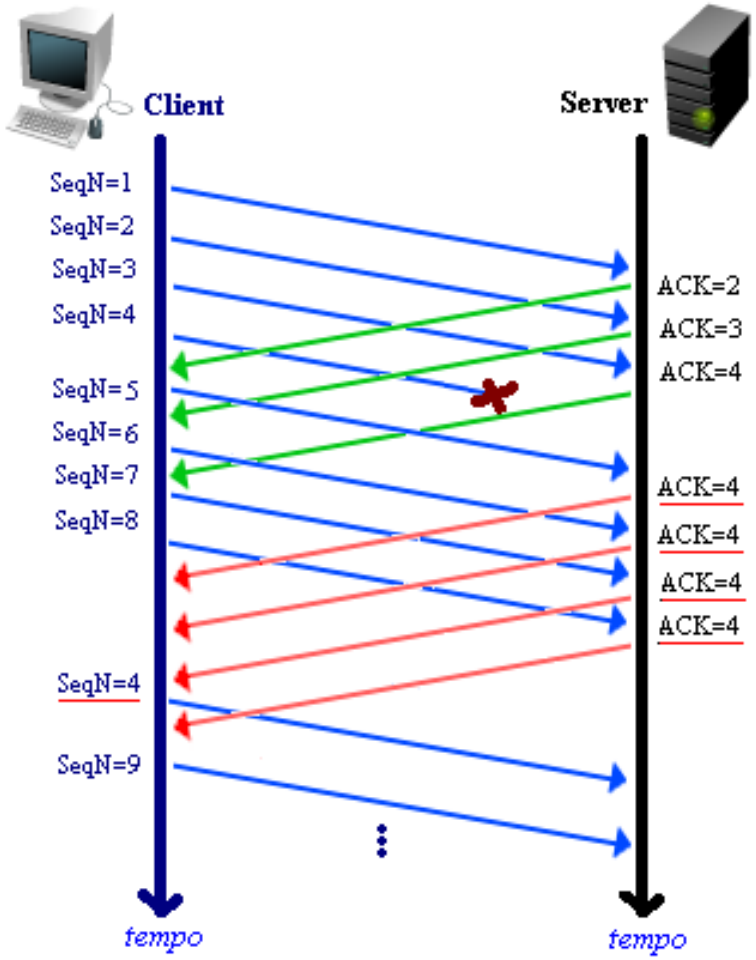

Figura 1.4: esempio tipico di comunicazionei durante una connessione TCP

timeout varia notevolmente in base alla rete, ma deve avere un valore pari almeno al Round Trip Time (RTT), stimato in base alla sua media mobile, ed incrementato fino ad un limite massimo ogni volta che scatta un timeout. La corretta impostazione di questo timer è ardua ma molto importante, in quanto timer troppo corti comportano ritrasmissioni inutili mentre timer troppo lunghi comportano attese superflue.

- timer di persistenza: gestisce i casi in cui la Receive Window è zero ed il messaggio di un byte viene perso, costringendo il mittente ad un'attesa

indefinita. Il timer di persistenza attende un tempo prefissato allo scadere del quale invia ad intervalli regolari il segmento di un byte per assicurarsi che la macchina ricevente sia ancora bloccata.

- timer di keep-alive: invia costantemente (tipicamente 5-45 secondi) un pacchetto vuoto per assicurarsi che la connessione sia ancora attiva. Se non si riceve risposta entro un tempo prefissatto si assume che la connessione sia interrotta.

- timed wait: entra in gioco alla chiusura della connessione per permettere a pacchetti ancora circolanti in rete di raggungere la destinazione al fine di evitare la riapertura della porta a connessione effettivamente terminata. Il timer viene solitamente impostato a valori pari al doppio del tempo di vita massimo di un segmento, che segnano lo scarto dei pacchetti ancora in transito.

Stima RTT

Come suddetto, parametro essenziale per il successo dell'affidabilità delle trasmissioni è la durata del RTT. Si ricorda che questo valore è molto variabile dipendentemente dal traffico della rete, dall'instradamento dei pacchetti, dalla banda e così via. Questo valore viene calcolato inizialmente considerando il tempo che intercorre tra l'invio di un pacchetto con un particolare numero di sequenza e la risposta. Successivamente, per ogni misurazione effettuata, il valore viene aggiornato in base alla seguente formula:

$$RTT_{stimato} = (1 - \alpha) * RTT_{misurato} + \alpha * RTT_{stimato}$$

dove α è un valore variabile tra 0 e 1 (estremi esclusi) che indica il grado di variabilità del RTT stimato. Dal RTT stimato si può assegnare il valore al timer di ritrasmissione come segue:

$$timer_{retrasmission} = RTT_{stimato} * \beta$$

dove β assume valori maggiori di 1 (valore consigliato da specifiche 2). Purtroppo il meccanismo descritto ha diverse limitazioni emerse sempre più con il tempo legate essenzialmente alle ambiguità causate dalla ritrasmissione dei pacchetti e relativi ACK. L'algoritmo di *Karn/Partrige* offre delle soluzioni a questi problemi, ed anche valide alternative introducendo, per esempio, stime basate su *Exponential Backoff*.

1.2.4 Controllo del Flusso

Il controllo del flusso utilizza una tecnica chiamata *Sliding Window*, pensata per cercare di evitare overflow del buffer dell'host ricevente, problema che sorge quando il tasso di invio dei pacchetti è maggiore della capacità di lettura e smaltimento dei pacchetti accumulati nel buffer. Il controllo del fusso interviene dunque tenendo traccia dello spazio libero del buffer di ricezione in una variabile detta *Receive Window*, contenente un valore dinamico recuperato dal campo Window Size, a sua volta impostato con l'ammontare di spazio disponibile durante la fase di invio di un ACK. Tale valore è calcolato come segue:

$$ReceiveWindow = BufferSize - (LastByteReceived - LastByteRead)$$

dove *ReceiveWindow* indica il valore della finestra del ricevente, *BufferSize* indica la dimensione del buffer di ricezione, *LastByteReceived* e *LastByteRead* contengono rispettivamente il valore dell'ultimo byte ricevuto e letto. Il mittente alla ricezione dell' ACK recupera il valore da Window Size e aggiorna la Receive Window. Di conseguenza i pacchetti trasmessi saranno adattati in base al valore ricevuto rispettando il seguente vincolo:

$$LastByteSent - LastByteAcked \leq ReceiveWindow$$

dove *LastByteSent* indica il valore dell'ultimo byte inviato e *LastByteAcked*

indica il valore dell'ultimo byte verificato dalla ricezione dell'ACK. Il sistema è estremamente semplice e garantisce dei risultati consistenti; vi sono comunque casi limite in grado di produrre il blocco della trasmissione come nella situazione in cui il buffer di ricezione si riempie facendo assumere alla finestra di ricezione un valore pari a 0. A questo punto il mittente interrompe immediatamente l'invio dei dati per evitare l'overflow del buffer, mentre l'applicazione destinazione processerà i dati nel buffer che si libererà fino a svuotarsi. Tuttavia il destinatario non potrà notificare lo svuotamento al mittente, per impossibilità di invio di ACK, non permettendo al mittende di riprendere l'invio di dati. Questa criticità è risolta dal protocollo imponendo l'invio da parte del mittente di un pacchetto delle dimensioni di 1 byte che permetta di notificare al destinatario lo stato del buffer.

1.2.5 Controllo della congestione

Il controllo della congestione si manifesta quando la quantità di dati immessi nella rete risulta troppo elevata rispetto alla possibilità di essere smaltita ed è tra le principali cause di declino delle prestazioni delle reti. Senza entrare nello specifico è facile immaginare gli inconvenienti legati al perdurare di una situazione di congestione in diversi nodi, tra cui ritardo dei pacchetti, overflow dei buffer con conseguente eliminazione dei pacchetti, ritrasmissioni (spesso superflue) ripetute, etc. Le diverse versioni di TCP sviluppate nel corso degli anni si diversificano proprio per il modo di reagire a improvvisi degradamenti delle prestazioni della rete. In realtà, nelle prime versioni del TCP, sostanzialmente non era preso alcun provvedimento, ma dal 1986, con *TCP Berkeley*, si cominciarono a vedere i primi veri e propri meccanismi di congestion avoidance. Successivamente, il *TCP Tahoe* aggiunse la fase di Fast Retransmit seguita pochi anni dopo da quella di Fast Recovery introdotta dal *TCP Reno*. Nel 1995 una leggera modifica condusse al *TCP New Reno*, utilizzato moltissimo negli anni. Le successive versioni nacquero principalmente con l'affermazione delle reti wireless per cercare di adattarsi alle condizioni del canale. Per sommi capi, l'approccio genercio per concretizzare

il controllo della congestione è fare in modo che ogni mittente limiti il ritmo di immissione dei dati nella rete in funzione della sua condizione rilevata. La frequenza di trasmissione dunque aumenta o diminuisce in base a specifici parametri messi a disposizione dal protocollo. In particolare, il mittente regola la quantità di dati immessi nella rete utilizzando una variabile chiamata *Congestion Window* (C_{wnd}) che impone una limitazione addizionale a quella già definita con il controllo del fusso. Facendo riferimento a quanto detto in precedenza il vincolo si trasforma quindi in:

$$LastByteSent - LastByteAcked \leq min\{C_{wnd}, ReceiveWindow\}$$

L'evento di congestione è notificato tramite la ricezione di 3 ACK duplicati (DUPACK) o, in estremis, tramite un timeout e gestito diversamente a seconda della versione TCP utilizzata.

1.3 Versioni TCP

La prima versione del TCP fu *Berkeley*, e cercò di gestire le congestioni senza indagare effettivamente lo stato della rete e modificando la frequenza di invio dei dati sulla base di una certa soglia, detta SS_{thresh} (Slow Start Threshold), sotto la quale C_{wnd} cresce in maniera esponenziale in funzione degli ACK ricevuti, mentre al di sopra della quale rallenta diventando lineare. Oltre alla sogliatura, lo scadere di un timeout segnalerà un evento di perdita, quindi il congestionamento della rete, e riporterà la finestra di congestione al valore iniziale, solitamente un *Maximum Segment Size* (MSS), ed alla riduzione di SS_{thresh} (metà del valore di C_{wnd}).

1.3.1 TCP Tahoe

Il controllo della congestione appena introdotto risultò poco specifico ed in molti casi drastico e fu migliorato da una successiva versione del TCP chiamata *Tahoe*. Modifica principale fu l'introduzione dell'algoritmo *Fast*

Retransmit, che definì un nuovo tipo di evento di perdita in aggiunta al timeout: la ricezione di ACK duplicati. Il timeout infatti è un periodo di tempo notevolmente lungo che bisognerebbe attendere soltanto in casi estremi (e.g. perdite multiple in entrambi i lati). Tale attesa può essere ridotta rilevando più rapidamente, lato server, se un pacchetto è andato perso e notificando al mittente tale perdita per mezzo di DUPACK, cioè con uguale numero di sequenza in risposta a pacchetti con numero di sequenza maggiore di quello del pacchetto potenzialmente perso. Il funzionamento generale del protocollo è mostrato in Figura 1.5.

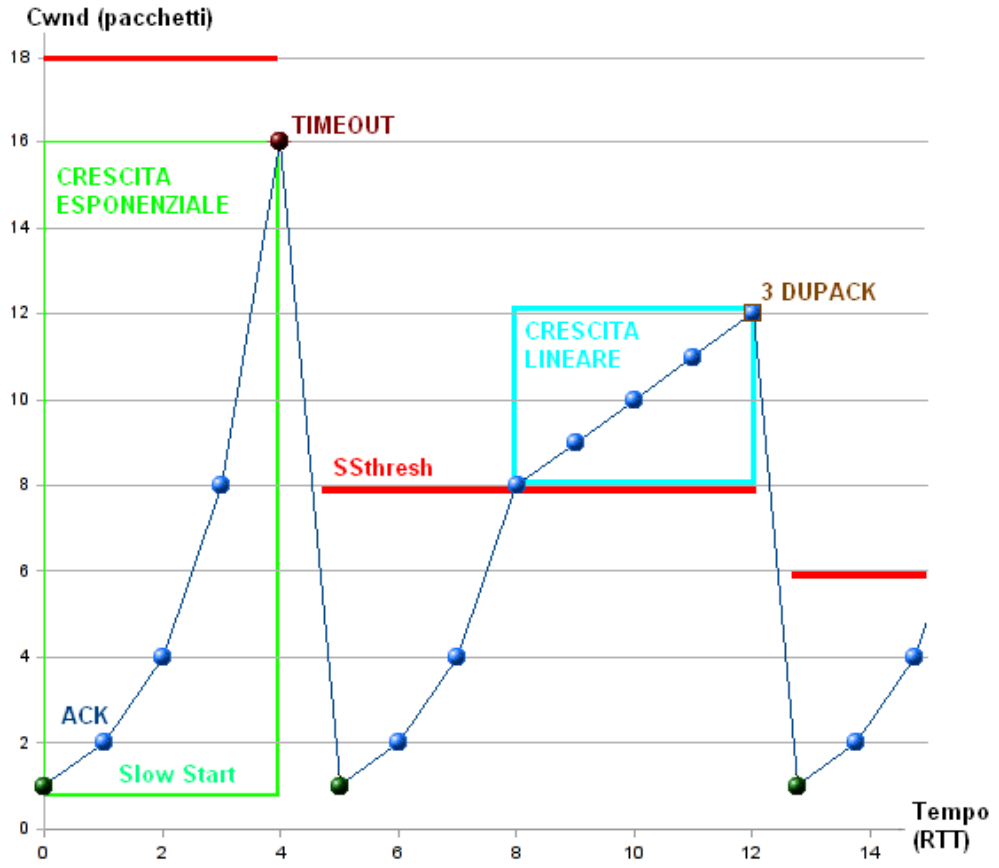

Figura 1.5: TCP Tahoe

All'avvio della connessione, si imposta un valore tipicamente alto per SS_{thresh} ed uno minimo per C_{wnd}, supponiamo 1MSS. Inizialmente il valore

di C_{wnd} viene aumentato esponenzialmente per ogni RTT e, più precisamente, C_{wnd} è incrementata di 1MSS per ogni ACK ricevuto (fase di *Slow Start*). Praticamente, ipotizziamo che il mittente invii il primo pacchetto e riceva il rispettivo ACK, portando C_{wnd} a 2MSS. Nel RTT successivo i pacchetti inviati sono 2: ciò implica 2 riscontri ed un aumento di 2MSS che porta C_{wnd} a 4MSS. La crescita prosegue finché non si verifica uno dei seguenti eventi:

- $C_{wnd} = SS_{thresh}$, si entra dunque nella fase di *Collision Avoidance* che prevede una crescita lineare, quindi ridotta, per la finestra di congestione, $C_{wnd} = C_{wnd} + \frac{1}{C_{wnd}}$;
- ricevuti 3 DUPACK, si entra nella fase di Fast Retransmit che segnala un evento di perdita e comporta l'abbassamento della soglia a $SS_{thresh} = \frac{C_{wnd}}{2}$ ed il reset della finestra a $C_{wnd} = 1MSS$;
- timeout, segnala anch'esso un evento di perdita e si ha lo stesso comportamento adottato per i 3 DUPACK.

Per chiarire il comportamento dei DUPACK, supponiamo che al ricevitore arrivi un pacchetto corretto e nel giusto ordine: egli invierà al mittente un ACK con il campo Acknowledgment Number settato al valore del campo Sequence Number incrementato di uno. All'arrivo del successivo pacchetto, se presenti degli errori, il ricevente invierà un ACK identico al precedente. Stessa situazione se un pacchetto viene perso ed al ricevente viene consegnato un pacchetto con numero di sequenza maggiore di quello aspettato. La trasmissione di diversi ACK duplicati indica quindi al mittente che qualcosa è andato storto. Di conseguenza, dopo il primo DUPACK, appena potrà il mittente proverà a ritrasmettere il pacchetto in questione che potrà arrivare prima del terzo DUPACK, dopo il terzo DUPACK, arrivare corrotto o affatto, con le rispettive conseguenze. I pacchetti spediti dal mittente successivi a quello perduto, ma prima dell'arrivo del terzo DUPACK, possono essere salvati (senza alcun obbligo da parte delle specifiche) nel buffer del ricevente in modo da essere riscontrati tutti insieme nel momento dell'arrivo del pacchetto ristrasmesso.

1.3.2 TCP Reno

La novità introdotta dal *Reno* fu l'algoritmo *Fast Recovery*. Questo consiste essenzialmente in una reazione differente agli eventi di timeout e di 3 DUPACK. Questa distinsione nasce dalla considerazione che analizzando la rete è più probabile che ci si trovi di fronte ad una congestione allo scadere di un timeout (poiché nessun dato è ricevuto per lunghi periodi) piuttosto che alla ricezione di 3 DUPACK, che indica probabilmente pacchetti corrotti o perdite isolate, nonostante non sia sempre così. Il funzionamento del TCP Reno è mostrato in Figura 1.6.

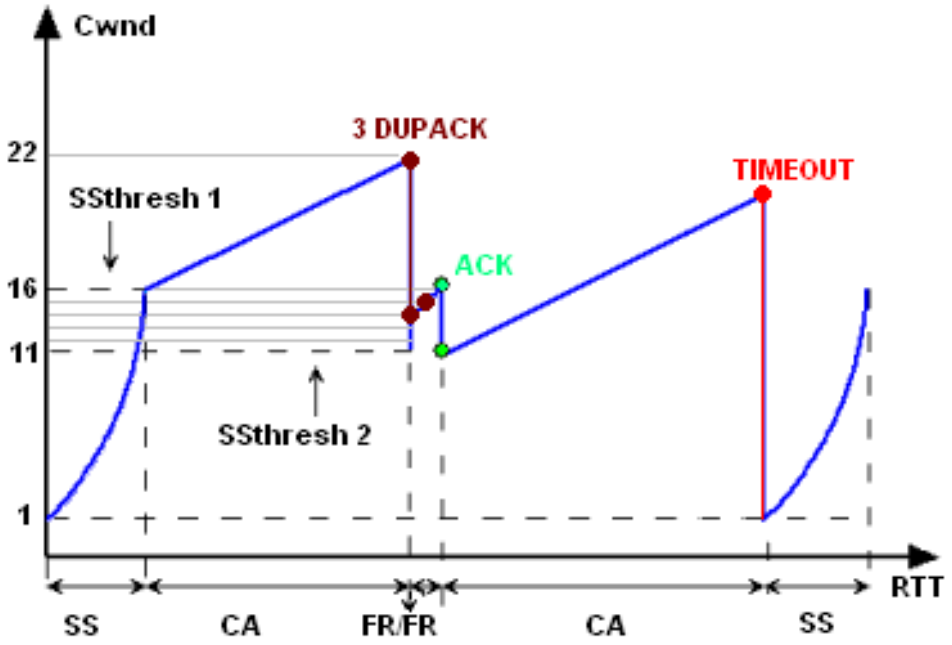

Figura 1.6: TCP Reno

Mentre il comportamento per il timeout risulta lo stesso del Tahoe, la ricezione dei 3 DUPACK conducono al seguente comportamento:

- $SS_{thresh} = \frac{C_{wnd}}{2}$;
- il pacchetto in questione viene ritrasmesso;

- $C_{wnd} = \frac{C_{wnd}}{2} + 3MSS.$

A questo punto la C_{wnd} viene incrementata di 1MSS per ogni ulteriore DUPACK ricevuto, finché non arriva un ACK non duplicato che riscontra tutti i dati spediti conseguentemente al pacchetto perso. C_{wnd} viene allora abbassata nuovamente al valore di SS_{thresh} e si ricomincia con la fase di Collision Avoidance. Si noti che questo meccanismo risulta efficiente nel caso di una singola perdita o corruzione diminuendo significativamente i tempi di attesa ma contemporaneamente riducendo minormente la frequenza di invio. D'altro canto se le perdite fossero multiple accadrebbe che dopo ripetute fasi di Fast Retransmit/Fast Recovery, le fasi di Congestion Avoidance conducano ad invii molto ridotti e quindi a ricezioni di DUPACK in tempi molto lunghi, tanto da portare spesso allo scadere dei timeout.

1.3.3 TCP New Reno

Per risolvere il problema dell'inefficienza dovuta alle perdite multiple all'interno della stessa finestra del TCP Reno e nell'ottica di un utilizzo sempre maggiore di canali wireless, la nuova versione chiamata *TCP New Reno* introduce il meccanismo degli *ACK parziali*, affiancato da una diversa impostazione del valore di SS_{thresh}. Quest'ultimo viene qui calcolato al momento della creazione della connessione utilizzando l'equivalente in bytes del prodotto $ritardo * ampiezza$ della banda, parametri valutati tramite algoritmi specifici (e.g. Packet Pair). Riguardo la gestione di perdite multiple all'interno della stessa finestra, il New Reno introduce il concetto di ACK parziali, ossia ACK che riscontrano solo alcuni dei pacchetti che sono stati inviati prima di entrare in fase di Fast Recovery; il mittente quindi tiene traccia dell'ultimo pacchetto inviato prima dell'ingresso in FR e resta in fase di FR finché tutti i pacchetti persi prima del FR non vengono riscontrati. Alla ricezione di un ACK parziale il mittente suppone una perdita successiva a quella notificata e ritrasmette immediatamente il pacchetto successivo a quello riscontrato dall'ACK parziale rimanendo però nella fase di Fast Recovery ed aumentando di 1 MSS la C_{wnd} per ogni ulteriore ACK duplicato relativo al

secondo pacchetto perso. Il TCP New Reno entrerà in Congestion Avoidance solo quando riceverà l'ACK che riscontrerà tutti i pacchetti inviati prima di entrare in Fast Recovery.

1.3.4 Altre versioni

Per completezza di esposizione e poiché il lavoro svolto in questa tesi si concentra essenzialmente sulle due versioni del protocollo appena descritte, in particolare nell'analisi e modellazione matematica di New Reno, le altre versioni rilasciate saranno descritte sommariamente, rimandando il lettore alle specifiche del protocollo per ulteriori approfondimenti.

Westwood

L'idea principale che caratterizza il TCP Westwood è di ottenere una stima della banda disponibile, intesa come velocità di trasferimento attuale dei dati, misurando la velocità con cui arrivano gli ACK. Tale stima è il parametro chiave per regolare la finestra di congestione e la sogliatura in risposta ad eventi di congestione.

Vegas

Il principio base di questa versione si discosta da quello delle altre, mirando non solo a reagire ad eventi di perdita ma a rallentare la frequenza di invio ancora prima che occorrano, al fine di prevenire la congestione. Analizzando le variazioni dei RTT per i segmenti già inviati, il mittente ha una panoramica dello stato della rete, ed aumenta o diminuisce la finestra di congestione dipendentemente all'accorciarsi o all'allungarsi del RTT.

Sack

Merito principale è l'introduzione dei *Selective Acknoledgment* (SACK), che si occupano di risolvere il problema di perdite non consecutive. Ogni

dispositivo modifica la sua coda di ritrasmissione in modo tale che ogni segmento includa un flag di Selective Acknowledgment. Se impostato a 1, tale flag indica al mittente di non reinviare il segmento corrispondente nel caso vengano persi pacchetti precedenti nella finestra di invio.

Cubic

Cubic fornisce un buon servizio in presenza di vaste reti in cui possono alternarsi stati di forte congestione a periodi di ampia banda disponibile. Questo TCP è l'attuale versione esistente su sistemi linux. CUBIC utilizza una funzione di crescita della finestra determinata dalla funzione

$$W_{cubic} = C * (t - K)^3 + W_{max}$$

dove C è un fattore di scala, t è il tempo intercorso dall'ultima riduzione della finestra, K è il tempo dell'ultimo evento di perdita, W_{max} è la dimensione della finestra appena prima l'ultima riduzione causata da un evento di perdita. In generale, la finestra cresce molto velocemente dopo un evento di perdita fino ad avvicinarsi a W_{max}, mentre rallenta fin quasi a fermarsi nell'intorno di questo valore, superato il quale ricomincia a crescere esponenzialmente (fase di Probing). La crescita lenta nell'intorno di W_{max} assicura la stabilità del protocollo, mentre il successivo aumento repentino la scalabilità. Inoltre, la funzione cubica assicura l'equità tra diversi flussi di dati sia rispetto alla banda che rispetto al RTT.

Veno

Questa versione si propone di apportare migliorie soprattutto per comunicazioni su reti wireless, dove ha evidente importanza riscontrare se il motivo delle perdite è dovuto a situazioni di congestione o a rumore nel canale ed interruzzioni di connessione (in questo caso si parla di *random loss*). Veno rappresenta l'unione di Reno e Vegas, in cui il meccanismo di Reno è perfezionato con la funzionalità di Vegas di stimare il RTT del canale per permet-

tere di intrapredere azioni differenti sulla base di situazioni di congestione o random loss.

Capitolo 2

OBIETTIVI E SCENARIO

Questo capitolo mira a definire e spiegare gli obiettivi principali del lavoro di tesi. Nel contempo si occupa di mostrare uno scenario classico relativo al funzionamento di un semplice sistema di comunicazione wireless, utilizzato come base di analisi e confronto per un sistema modificato con l'aggiunta di un meccanismo di ritrasmissione anticipata che mira all'ottimizzazione delle prestazioni. Verrà inizialmente mostrata l'architettura globale del sistema e definiti i vari attori implicati. Successivamente ci si focalizzerà sulla parte wireless del sistema, illustrando in particolare il funzionamento dello standard Wi-Fi (802.11), evidenziando i più comuni metodi di accesso al canale condiviso e discutendo pregi e difetti. Infine, lo scenario delineato in precedenza verrà riesaminando approfonditamente esplicitando le sezioni più significative che lo compongono, ed, in particolare, i punti di criticità associati al suo reale funzionamento al fine di comprendere nel modo più chiaro possibile il meccanismo fondamentale per le successive analisi.

2.1 Obiettivi

Gli obiettivi che si intendono raggiungere con lo svolgimento di questa tesi possono essere raccolti essenzialmente in tre punti.

- Primo obiettivo e prerequisito fondamentale è la comprensione del funzionamento di un semplice sistema di comunicazione (mostrato in Figura 2.1 e chiamato d'ora in avanti **sistema classico**) basato su reti cablate e wireless che sfruttano il TCP e conseguentemente del corrispettivo sistema modificato con l'aggiunta di una ritrasmissione anticipata dei pacchetti persi (mostrato in Figura 3.2, a cui ci si riferirà d'ora in poi con il nome di **sistema anticipato**).

- In secondo luogo, si vuole trovare un modello matematico di riferimento che descriva il sistema classico permettendo di analizzarlo e comprenderlo secondo un'ottica che astragga da specifiche implementazioni e da particolari applicazioni, ma allo stesso tempo offra un metodo generale per la valutazione di sistemi analoghi.

- Infine, si vogliono confrontare i due sistemi e valutarli in base alle proprie caratteristiche, discutendone i problemi principali, analizzandone il comportamento ed i casi specifichi di funzionamento, definendo e ragionando sugli aspetti matematici comuni e contrastanti.

Il primo obiettivo sarà raggiunto grazie alle prossime sezioni di questo e del successivo capitolo che mostreranno i due sistemi nella loro totalità, il relativo funzionamento generale, i protocolli specifici che li governano ed i componenti principali insieme alle loro interazioni, comportamenti, vincoli e funzionalità. Il Capitolo 4 servirà da preambolo per affacciarsi al mondo della modellazione matematica in ambito delle reti di calcolatori, in particolare considerando il TCP. Successivamente nel Capitolo 5 si entrerà nel vivo della modellazione introducendo il contesto, descrivendo le nozioni principali per la comprensione del modello, esplicitando e motivando scelte ed assunzioni fatte, delineando le problematiche affrontate ed i compromessi raggiunti. Il Capitolo 6 spiegherà il modello vero e proprio descrivendone funzionamento, limiti e applicabilità e getterà le basi per lo sviluppo di un modello derivato che si adatti al sistema anticipato. Infine, nel capitolo 7 saranno raccolte una serie di considerazioni e proposte relative agli sviluppi futuri.

2.2 Architettura Globale del Sistema

A livello globale il sistema considerato è formato dai quattro componenti mostrati in Figura 2.1 e descritti di seguito.

Figura 2.1: componenti principali del sistema

- Un dispositivo (o nodo) mobile (e.g. uno smartphone, un tablet, un netbook, etc) (MN).
- Un Access Point (AP).
- Zero o più router.
- Un server.

Lo scenario consiste in un MN che agisce da client e comunica (utilizzando il TCP) con un AP per mezzo di un canale wireless. L'AP fornisce al MN un punto di accesso alla rete e lo mette in comunicazione con un server facendo affidamento su un canale wired. Tale canale può essere direttamente connesso all'end point server, o può prevedere uno o più router che smistino il traffico e lo instradino correttamente alla destinazione. Spostandosi nello spazio circostante all'AP, il dispositivo mobile attraverserà diversi stadi relativi alla sua connessione, dettati da periodi di ottima stabilità del segnale, periodi di connessione mancante, periodi in cui l'ambiente o altri dispositivi generano interferenza, e così via. L'affidabilità della connessione sarà quindi fortemente influenzata da una serie di eventi caratteristici del mezzo condiviso wireless e che un protocollo TCP classico, pensato essenzialmente per sistemi basati su reti cablate, può arginare limitatamente.

Per comprendere appieno caratteristiche, benefici e vincoli legati alle comunicazioni wireless, la prossima sezione tratterà approfonditamente lo standard Wi-Fi dedicato alle comunicazioni senza filo.

2.3 Standard Wi-Fi (802.11)

Lo standard IEEE 802.11 è una famiglia di standard relativi al Livello Fisico del modello OSI che si occupano di regolare la comunicazione tra dispositivi all'interno di una rete Wireless. Esistono sette diversi tipi di protocolli 802.11, che rispondono alle esigenze dello sviluppo tecnologico in ambito di reti wireless, ognuno di esso basato su un tipo di comunicazione che avviene attraverso frequenze radio ed all'interno di una determinata ampiezza di banda (bandwidth). Fondamentale per il protocollo è il livello MAC (Media Access Control), un sottolivello del Data Link Layer che fornisce funzionalità di indirizzamento e meccanismi di controllo degli accessi a canali di comunicazione condivisi.

2.3.1 Architettura

L'architettura di una rete di comunicazione wireless LAN 802.11 è mostrata in Figura 2.2.

Essa si basa su una struttura a celle (simili al GSM) chiamata *Basic Service Area* (BSA). Un gruppo di *stazioni* comunicanti situate all'interno di una BSA formano un *Basic Service Set* (BSS), controllato da una *Base Station* (BS) o *Access Point* (AP). Molteplici celle possono essere combinate a formare LAN wireless in cui i singoli AP sono interconnessi attraverso *Distribution System* (DS), generalmenti costituiti da dorsali Ethernet, ottenendo così delle *Extended Service Area* (ESA). L'insieme di wireless LAN e delle infrastrutture di interconnessione viene visto come una singola rete 802 dai livelli superiori del modello OSI ed è definito *Extended Service Set* (ESS). Tra i diversi tipi di wireless LAN esistenti, ricordiamo le reti *Ad Hoc*, formate da diverse stazioni in un area limitata che permette facilità di installazione

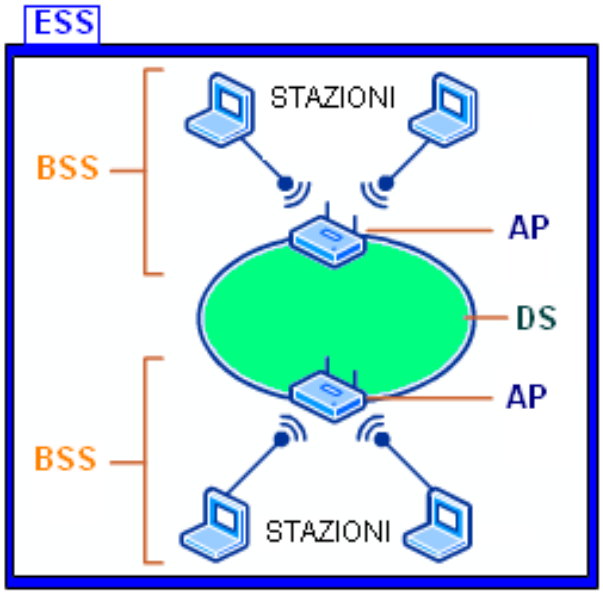

Figura 2.2: architettura 802.11

indipendentemente da infrastrutture precedenti, e le reti *Infrastrutturali*, che includono gli AP ognuno legato ad uno specifico BSS.

2.3.2 Accesso al canale: MAC

Il livello MAC (Medium Access Control) è il responsabile della gestione degli accessi al canale wireless condiviso e mette a disposizione le due seguenti tecniche.

- *Distributed Coordination Function* (PCF), rappresenta la tecnica fondamentale ed impiega un metodo *Carrier Sensing Multiple Access with Collision Avoidance* (CSMA/CA) con un algoritmo di backoff esponenziale binario. Quando una stazione vuole trasmettere, testa il canale di trasmissione: nel caso sia occupato ritarda la trasmissione ad un momento successivo (basato su un random backoff), altrimenti può trasmettere. L'efficienza di questi controlli dipende dal traffico del canale. Tuttavia, quando più stazioni rilevano contemporaneamente che

il mezzo trasmissivo è libero può succedere che inizino a trasmettere simultaneamente, provocando delle collisioni. Per risolvere questi incovenienti, il protocollo 802.11 utilizza un meccanismo di collision avoidance unito ad uno schema di positive acknowledge. Sulla base di tali considerazioni, una stazione che vuole trasmettere, testa il canale e posticipa la trasmissione se occupato, ma, se libero, prima di trasmettere attende un certo periodo di tempo chiamato *Distributed Inte-Frame Space* (DIFS), al termine del quale può iniziare la trasmissione. La stazione ricevente verifica eventuali errori nei dati ricevuti e risponde con un ACK, che se perso causerà una serie limitata di ritrasmissioni da parte del mittente. Inoltre, DCF fornisce un meccanismo opzionale di *Virtual Carrier Sense* basato sullo scambio di piccoli pacchetti di controllo chiamati *Request-to-send* (RTS) e *Clear-to-send* che fungono da coordinatori tra le stazioni comunicanti e attutiscono le collisioni.

- *Point Coordination Function* (PCF), usata per implementare servizi che richiedono maggiore efficienza, come sistemi real-time. Similmente alla DCF, la PCF utilizza un intervallo di tempo chiamato *Point Inter-Frame Space* (PIFS) che fa acquisire priorità di accesso al canale ad alcune stazioni privilegiate e tramite un meccanismo di polling guidato dagli AP si controlla l'accesso al mezzo per la trasmissione, facendo comunque attenzione a lasciare sufficiente tempo alla DCF per la trasmissione delle stazioni non privilegiate.

2.3.3 Livello Fisico: PHY

Il livello fisico è composto da due sottolivelli chiamati *Physical Medium Dependent* (PMD) e *Physical Layer Protocol Convergence* (PLPC). Il PMD si interfaccia direttamente col mezzo fisico, i.e. *Radio Frequency* (RF) o infrarossi, trasformando l'input ricevuto dal PLCP in segnale fisico che possa essere ricevuto da un'altra stazione, e viceversa. Il sottolivello PLCP invece converte i dati in uscita dal MAC in un formato consono per il PMD. I

sottomoduli per le trasmissioni a livello fisico prevedono una serie di tecniche di espansione dello spettro, tra cui *Frequency Hopping Spread Spectrum* (FHSS) e *Direct Sequence Spread Spectrum* (DSSS), affiancate da tecniche di modulazione digitale, come *Phase-Shift Keying* (PSK), e analogica, come *Quadrature Amplitude Modulation* (QAM), che permettono di convertire i segnali nei diversi formati necessari per la comunicazione.

2.4 Analisi approfondita del sistema

Come anticipato nella seconda sezione di questo capitolo, il sistema considerato consta di quattro componenti: Mobile Node, Access Point, router e server. Per lo scopo prefissato, ci si focalizza sull'interazione dei componenti elencati specialmente in prossimità di alcuni punti critici (e.g. buffer) che in determinate circostanze possono diventare dei colli di bottiglia del sistema influenzando negativamente le prestazioni.

Mobile Node

Il punto di partenza del sistema è il MN che attiva la comunicazione innescando una serie di azioni sintetizzate come segue:

- l'applicazione utente effettua una richiesta che viene inoltrata nello stack di rete e ricevuta dal TCP a livello di trasporto;
- il TCP gestisce debitamente la richiesta ed accoda i pacchetti in ordine FIFO memorizzandoli all'interno di un buffer;
- i pacchetti, tramandati attraverso lo stack dai livelli rete e data-link che li elaborano opportunamente (e.g. IP, frammentazione), giungono, nel formato consono, al livello MAC dove vengono accodati nel buffer di trasmissione delle interfacce delle schede di rete (NIC)[1];

[1]Durante queste fasi si assume ragionevolmente l'utilizzo di un meccanismo di gestione che prevenga eventuali buffer overflow, escludendo quindi legami di responsabilità con eventi di congestione dovuti al traffico vero e proprio.

- i pacchetti accodati in ordine FIFO nel buffer del livello MAC, vengono trasmessi sul canale wireless: per essi si attendono degli ack immediati che li confermino; in caso di perdite o attese fuori tempo massimo, prima di essere scartati si tenta di ritrasmettere i pacchetti direttamente fino ad un massimo di 7 volte.

In questa parte della comunicazione il canale potrebbe diventare un collo di bottiglia per le prestazioni del sistema a causa di eventi di congestionamento (per traffico eccessivo) o perdite di pacchetti (interferenza, ostacoli, disconnessioni, etc).

Access Point

L'AP si limita a ricevere i pacchetti dal MN ed inoltrarli al server tramite un link cablato. Il problema relativo a questo nodo potrebbe essere un'ingente quantità di pacchetti ricevuti dal MN, che intaserebbe il buffer di ricezione dell'AP provocando overflow (con conseguente dropping dei pacchetti secondo le diverse politiche di gestione delle code (AQM)).

Router e Server

Questi due componenti hanno un ruolo certamente importante nel contesto considerato, ma non soffrono di criticità rilevanti per lo scopo previsto; ciò deriva dal fatto che si ipotizza che la parte wired della connessione non agisca da collo di bottiglia del sistema, ma abbia invece ampiezza di banda sufficiente (rispetto alla parte wireless) ad evitare la congestione della rete.

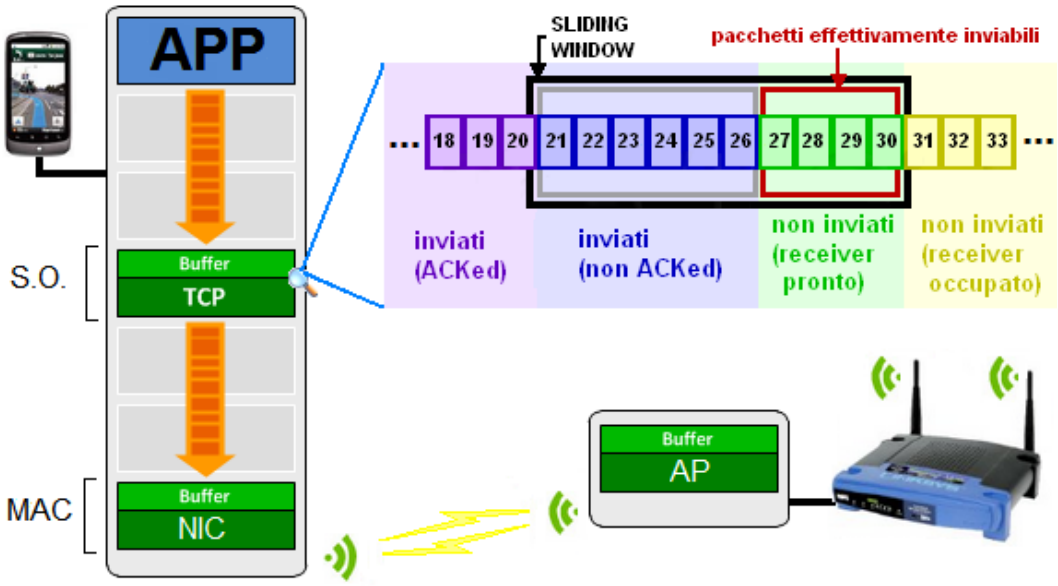

Figura 2.3: potenziali criticità del sistema classico

Capitolo 3

SISTEMA ANTICIPATO

In questo capitolo verrà mostrato un sistema analogo a quello descritto precedentemente ma ripensato per rendere più effciente lo scambio di pacchetti a livello di trasporto fra i due end-point. Prima di entrare nel vivo della descrizione del sistema modificato, si discuteranno alcuni prerequisiti necessari alla realizzazione di questa estensione. Si passerà poi all'esposizione dei componenti del sistema per giungere alle principali modifiche, aggiunte, limitazioni ed al funzionamento globale del sistema ottimizzato.

3.1 Prerequisiti: ABPS e RWMPC

Always Best Packet Switching (ABPS) è uno schema di comunicazione wireless cross-layer che permette a dispositivi mobili di sfruttare concorrentemente le proprie interfacce di rete. Esso è infatti in grado di determinare in ogni momento quale sia la NIC più appropriata per poter usufruire di un servizio di trasmissione ottimale, basato su QoS (Quality of Service). Il sistema utilizza un'architettura software distribuita il cui modulo principale è costituito dal Client Proxy in esecuzione all'interno del MN che comunica con la controparte, Server Proxy, ubicato in un host fisso esterno a firewall e NAT. In linea generale, ABPS funziona nel modo seguente: all'interno del nodo mobile c'è un sistema di monitoraggio avviato in background che si

occupa di identificare nuove possibili connessioni e di configurare le interfacce di rete; quindi il Client Proxy rileva l'interfaccia migliore basandosi su quelle correntemente attive. In realtà tutte le NIC possono essere utilizzate concorrentemente, ed è quindi possibile passare da una rete all'altra ad ogni istante in caso di degrado delle performance piuttosto che di fallimento della connessione. Lo scopo di un simile servizio è di garantire al nodo mobile la fruibilità ininterrotta dei servizi di comunicazione, la riduzione al minimo della perdita di informazioni, elevata autonomia e riduzione dei costi. In particolare, sfruttando l'eterogeneità delle NIC disponibili su ogni MN, si riesce a limitare l'emissione di onde elettromagnetiche, aumentare la banda e l'affidabilità delle trasmissioni, riutilizzare infrastrutture preesistenti, superare eventuali ostacoli imposti da firewall.

Componente fondamentale di ABPS è *Robust Wireless Multipath Channel* (RWMPC), un meccanismo finalizzato a garantire alle applicazioni VoIP un basso rirtardo di trasmissione e una perdita di pacchetti ridotta al minimo. Il design RWMPC è basato su un approccio cross-layer in cui le diverse informazioni provenienti dai diversi livelli dello stack ISO/OSI vengono utilizzate per monitorare attivamente la comunicazione wireless. In breve, un monitor alla base del sistema rileva l'eventuale perdita dei datagrammi UDP durante la trasmissione tra una NIC del nodo mobile e un AP, dopodiché notifica tale perdita all'applicazione da cui proviene quel determinato datagram cosicché tale applicazione può decidere se ritrasmettere il datagram perduto utilizzando una differente interfaccia. L'assunzione che il nodo mobile possegga più interfacce wireless è fondamentale in quanto RWMPC sfrutta le sue capacità di connettersi a più di una rete (multi-homing) per poter stabilire di volta in volta quale sia la più adatta per la ritrasmissione del datagram. Quando si verifica un evento di perdita, RWMPC ritrasmette utilizzando un collegamento alternativo tra quelli disponibili sulle interfacce di rete attualmente connesse alla rete. In pratica, RWMPC è essenzilmenrte un middleware intermediario posizionato su entrambi gli end-system di una comunicazione VoIP e incaricato di incapsulare i dati in datagrammi UDP per trasmetterli,

più volte se necessario, al destinatario.

I tre moduli principali di RWMPC sono mostrati in Figura 3.1 e sono descritti di seguito.

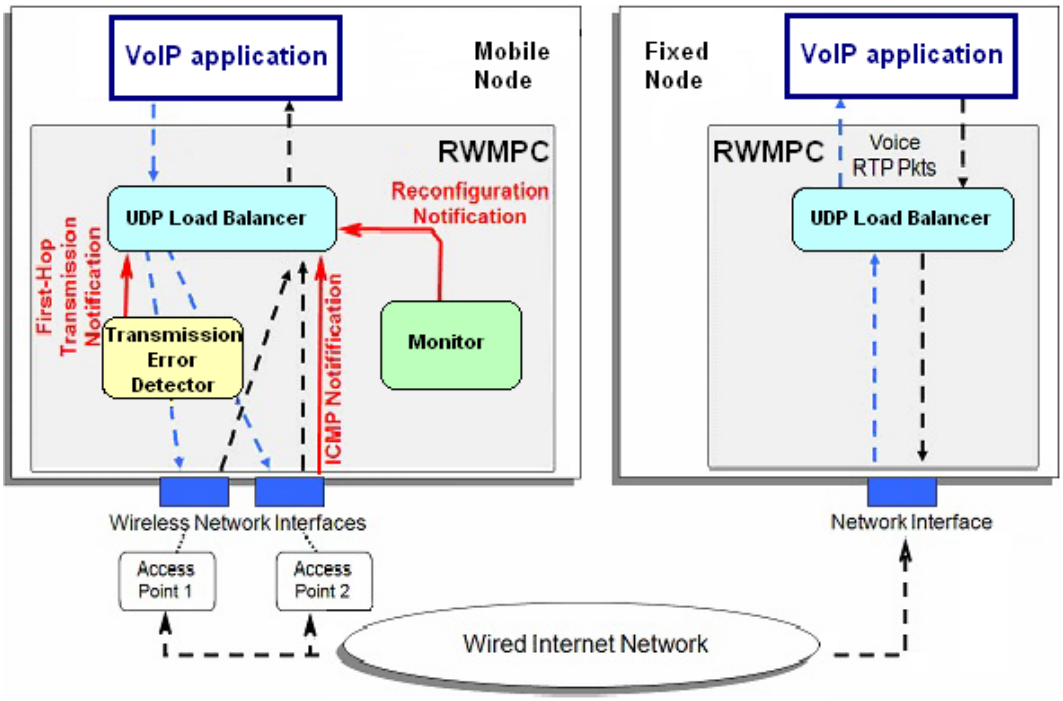

Figura 3.1: architettura RWMPC

- *Trasmission Error Detector* (TED), monitora i datagrammi UDP inviati sul canale wireless e individua se il pacchetto è stato ricevuto dall'AP o è stato scartato dal livello MAC. Successivamente, il TED informa l' ULB del mancato invio del pacchetto inserendo un messaggio di errore all'interno del socket utilizzato per l'invio.

- *UDP Load Balancer* (ULB), riceve i pacchetti generati dall'applicazione VoIP e li incapsula in datagrammi UDP per mandarli al sistema ricevente, oltre a ricevere le notifiche provenienti da TED per decidere se i pacchetti persi debbano essere reinviati.

- *Monitor*, gestisce le interfacce wireless del nodo mobile, configurandole dinamicamente e comunicando a ULB quali sono momentaneamente attive.

TED è il componente chiave per lo sviluppo del sistema anticipato e adeguatamente modificato permette il rinvio di pacchetti persi prima dello scadere del timeout predefinito all'interno del protocollo TCP.

3.2 TCP a ritrasmissione anticipata

La prima modifica del sistema consiste nell'aggiunta di un meccanismo di ritrasmissione anticipata al protocollo TCP, che fornisca il rinvio del messaggio qualora venga notificata una mancata ricezione da parte del destinatario, come mostrato in Figura 3.2.

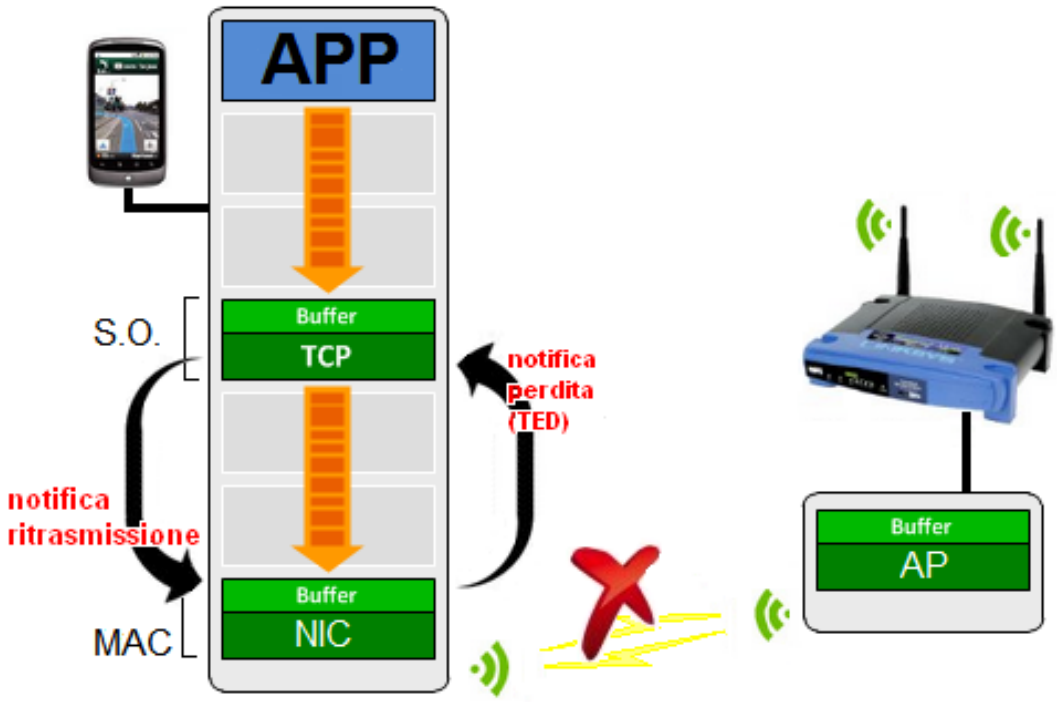

Figura 3.2: funzionamento e criticità del sistema anticipato

L'idea di base fornita da ABPS per un TED con UDP, è stata adattata per realizzare una versione funzionante con il TCP, integrando ed estendendo i requisiti di affidabilità già presenti. Al fine di minimizzare i tempi di ritrasmissione dei pacchetti è stato inserito un modulo di controllo a livello MAC tra il nodo trasmittente e l'AP che implica l'invio di una notifica anticipata allarmando sulla perdita di un pacchetto. In particolare, i pacchetti segnalati come persi vengono segnalati a livello TCP, recuperati tramite un identificativo creato appositamente per renderli tracciabili durante la trasmissione lungo lo stack di rete, e rispediti immediatamente al livello MAC. La concretizzazione del meccanismo di TED per il TCP è resa possibile dall'utilizzo del timer, il cui modulo di gestione è stato modificato per permettere la segnalazione anticipata della perdita e l'inserimento della gestione dell'evento di ritrasmissione anticipata. Come anticipato, tra le maggiori difficoltà di questo approccio c'è l'identificazione del pacchetto perduto, tracciabile percorrendo i livelli dello stack di rete verso il basso ma non viceversa. Per sormontare tale ostacolo in fase di creazione del segmento TCP è stato aggiunto un ulteriore identificativo che, propagato attraverso lo stack, permette di mantenere un'associazione tra i frame a livello data-link e i segmenti TCP incapsulati al loro interno. Questo sistema è implementato a livello di kernel del sistema operativo e permette di ridurre notevolmente i tempi agendo sul primo hop della comunicazione, quindi fra il mittente e l'AP, migliorando significativamente le prestazioni specialmente su canali wireless.

Si nota che tale meccanismo potrebbe essere implementato direttamente a livello MAC portando i tentativi di ogni ritrasmissione da, per esempio, 7 a 14. Teoricamente una modifica siffatta risulterebbe molto più immediata e semplice a livello implementativo, nonostante in linea di massima comporterebbe diversi svantaggi a livello di funzionamento globale del sistema (e.g. inadeguata priorità delle attività di scheduler e NIC, inefficacia con cluster di perdite, ed altro ancora).

Capitolo 4

STATO DELL' ARTE

Il quarto capitolo offre una panoramica dello stato dell'arte riguardante un sottoinsieme cospicuo dei lavori e delle pubblicazioni che, dagli anni '90 in poi, hanno affrontato le problematiche arzigogolate relative allo sviluppo di modelli matematici per l'analisi e la valutazione del comportamento del TCP su sistemi di reti. In particolare verranno approfonditi i principali lavori da cui si è preso maggiore spunto, chiarendone il contenuto e discutendone il ruolo intrapreso nello svolgimento della tesi.

4.1 Cos'è un modello matematico?

Un modello matematico è un modello costruito attraverso il linguaggio e gli strumenti della matematica. Similmente a tutti gli altri modelli usati nella scienza, il suo scopo è quello di rappresentare e descrivere il più efficacemente possibile degli oggetti o dei fenomeni reali, spesso cercando di ricavare previsioni sullo stato futuro, escludendo la necessità di compiere esperimenti. Un modello è generalmente legato ad un insieme di dati iniziali sulla base dei quali descrive la probabile evoluzione del sistema, generando in output dei dati finali. Gli strumenti matematici potenzialmente utilizzati sono i più disparati, andando per esempio dal calcolo combinatorio a quello infinitesimale. Le fasi per la costruzione di un modello matematico sono tipi-

camente: l'analisi del problema reale con seguente formulazione di ipotesi e dati sperimentali; costruzione del modello ottenendo delle relazioni matematiche (funzioni, equazioni, ...) che correlino le variabili in gioco; risoluzione delle relazioni matematiche individuate e previsioni sul fenomeno; infine, validazione del modello tramite confronti con dati sperimentali. Tuttavia un modello matematico non è mai una rappresentazione esatta della realtà ma una versione di essa sufficientemente semplificata nei limiti di validità.

4.2 Principali metodi di modellazione

Sebbene vi siano numerosi lavori relativamente differenti emersi negli anni per modellare il comportamento del TCP, i metodi più noti ed efficaci rientrano in un insieme ristretto, formato dai cinque descritti di seguito.

4.2.1 Renewal Theory

Probabilmente uno dei più noti e usati, si basa prevalentemente su sorgenti single source in cui il throughput è calcolato tramite metriche di performance ben note, come tasso di perdita dei pacchetti e ritardo. Il metodo s'incentra sullo studio dell'evoluzione della C_{wnd} in termini di cicli (i.e. periodo tra due eventi di perdita), considerati come periodi di rinnovamento in cui il sistema, supposto in uno stato stazionario, torna a ripetersi in lassi di tempo aleatori. L'usabilità, la semplicità e l'efficacia di questo modello lo rendono tra i più utilizzati e derivano dal fatto che i risultati offrono spesso soluzioni in forme algebriche chiuse, quindi esprimibili analiticamente in termini di un numero finito di funzioni bene note (tipicamente elementari come costanti, variabili, operazioni algebriche, radici, logaritmi, etc) facilmente implementabili, utilizzabili e scalabili. Inoltre questo metodo permette di modellare diversi aspetti delle dinamiche del TCP, come Slow Start, Collision Avoidance, Timeout, e molti altri, offre stime accurate e distingue le diverse versioni esistenti. Alcune delle espressioni conosciute come pietre miliari all'interno della comunità del TCP modeling sono la *square root law* e la *PFTK-formula*,

che forniscono una relazione tra il throughput e la probabilità di perdita dei pacchetti, considerando nel caso della PFTK-formula gli eventi di timeout. Uno tra gli svantaggi principali di questo metodo è certamentet la necessità di conoscere a priori il tasso di perdita dei pacchetti e l'RTT medio, che conducono quindi ad assunzioni sulla topologia della rete, statistiche di traffico, etc. Inoltre, questo metodo rischia di essere poco espressivo e indicato in caso di scenari molto complessi, dove l'utilizzo di sorgenti single source non permettono una stima sufficientemente accurata (e.g. Internet). Infine, in molti casi è necessaria l'assunzione di *long lived flows*, ovvero trasferimento persistente dei dati, che a volte può risultare limitativa.

4.2.2 Fixed-point

La crescita delle reti e la presenza di scenari sempre più complessi ha accresciuto nei ricercatori la necessità di considerare sorgenti multiple ed eterogenee, operanti in network arbitrarie. La creazione di questo tipo di modelli nasce dall'aggregazione di altri modelli TCP separati e più semplici, che descrivono situazioni atomiche o basilari e costituiscono i blocchi di un sistema più grande. Tuttavia esiste una relazione fondamentale che lega i diversi blocchi che compongono il framework totale: la dipendenza di ritardo e perdita dei pacchetti dal tasso di invio degli stessi, adattato di conseguenza. La metodologia di analisi dell'interazione tra modelli separati per trovare il regime di funzionamento delle reti è chiamata fixed-point method. Questo metodo è indubbiamente uno strumento potentissimo; presenta tuttavia degli svantaggi tra cui il bisogno di un grande lavoro implementativo e la necessità di poter analizzare e verificare separatamente l'accuratezza dei modelli di sorgente e di rete.

4.2.3 Fluid

I pacchetti TCP sono approssimati come un "fluido" che fluisce attraverso la rete, i cui parametri, come la dimensione della finestra di congestione

e la lunghezza della coda, sono assunti cambiare continuamente. Sistemi di equazioni differenziali descrivono tali cambiamenti e permettono di derivare le informazioni desiderate. Il tasso di perdita dei pacchetti è modellato come un processo stocastico che governa le dinamiche di evoluzione della finestra di congestione e permette di derivare le prestazioni in termini di sue proprietà. Le soluzioni delle equazioni differenziali forniscono quindi l'evoluzione nel tempo della dimensione della finestra di congestione e della lunghezza della coda, da cui è possibile ricavare metriche di performance come il throughput medio. Meriti principali di questo modello sono il fatto che le proprietà statistiche del processo di perdita sono direttamente incluse nel modello (poiché processo stocastico) oltre alla possibilità di sfruttare alcuni risultati analitici utili nelle simulazioni. L'altra faccia della medaglia mette in evidenza il dramma maggiore di questo metodo che riguarda il notevole aumento della complessità in caso si modellino aspetti dettagliati del TCP che non rientrino nella fase di Congestion Avoidance e dimezzamento della finestra di congestione dovuto a eventi di perdita.

4.2.4 Processor Sharing

Questo metodo studia le metriche di performance a livello di flusso, definito come il numero di pacchetti che corrisponde al download di uno specifico oggetto, per esempio un video, un'immagine, etc. Ciò è motivato dall'idea che tale metrica è quella maggiormente percettibile e significativa per l'utente, supponendo un contesto in cui il traffico sia vario. Si considera un comportamento per il flusso basato su processi di Poisson e si assume un utilizzo equo del canale da parte delle sorgenti. Le interazioni tra sorgenti e network sono modellate tramite dei sistemi a coda[1] (e.g. M/M/1, M/M/∞, M/G/R, ...) che soddisfano le suddette assunzioni ed, in determinate reti e scenari di traffico, permettono il calcolo dei tempi di download per uno specifico flusso

[1]Teoria delle code: studio matematico delle code di attesa e dei processi legati ad esse come il processo di arrivo dei clienti nella coda, il processo di attesa per essere serviti, ed il processo di servizio.

ed il numero di flussi simultanei nel sistema. Chiaramente, la scelta di rete e scenario di traffico ha un impatto molto elevato sull'usabilità del modello, ed in determinati casi ciò implica la perdita dell'obiettivo primario, cioè la modellazione del TCP, a favore della stima dei tempi di download. Ciononostante, come risultato si ha comunque il beneficio di ottenere una visione generale delle performance del sistema senza dover entrare in specifici dettagli del TCP, estrapolando delle linee guida per la configurazione di diversi parametri della rete, come capacità ideale dei link e stima del download di file specifici. Il modelllo si presta inoltre ad essere esteso con facilità senza introdurre elevata complessità aggiuntiva. D'altro canto, alcuni requisiti da tener conto per ottenere buoni risultati sono l'uso di lunghi trasferimenti di file (causa slow-start) ed un basso utilizzo del sistema. Limitazione importante e direttamente collegata alla genericità del framework nei confronti del TCP, è la scarsa efficacia del sistema nel modellare specifiche versioni del protocollo TCP e dei meccanismi di controllo del flusso e congestione che le caratterizza. Infine i vincoli di equità e set-up omogeneo delle sorgenti rendono il modello in molti casi poco realistico o nella peggiore delle ipotesi non valido.

4.2.5 Control-Theoretic

Metodo basato sugli approcci classici della teoria del controllo automatico (Control Theory). In questo contesto, una rete con sorgenti a flusso controllato è vista come un sistema di feedback largamente distribuito, dove i feedback sono meccanismi che permettono di controllare stati ed output di sistemi dinamici. I parametri di input di questo sistema sono la topologia della rete, il numero di sorgenti e la versione del TCP, mentre quelli di output sono il tasso di perdita ed il ritardo, che constituiscono un insieme rilevante e ordinario. Il contributo maggiore di questo metodo consiste sicuramente nell'affiancare all'analisi di sorgenti e reti l'obiettivo esplicito di progettare e proporre nuovi regimi di controllo del flusso stabili e scalabili per nuove reti ad alta capacità. In aggiunta, risultati positivi sono stati riscontrati anche

durante l'utilizzo di diversi protocolli di controllo del flusso e politiche di accodamento. Per contro, non tutti gli scenari ed i tipi di analisi possono essere trattati, in particolare perché molti dei risultati accettabili riguardano flussi long lived e sono transitori, sebbene molti dettagli possano essere introdotti a discapito di un incremento significativo della complessità.

4.3 Lavori in letteratura

Ci si soffermerà soprattutto sui lavori in ambito di Renewal Theory, che hanno un ruolo centrale per il raggiungimento dell'obiettivo prefissato, dedicando minor attenzione ai lavori basati sugli altri metodi, trattati all'incirca ugualmente ad eccezione del metodo Fixed-Point, lievemente più importante nel contesto considerato viste le sue potenzialità.

4.3.1 Renewal Theory in letteratura

In ambito di Renewal Theory model, vi sono numerose pubblicazioni e la comunità dei ricercatori è risultata e tuttora risulta molto attiva. Tra i primi lavori maggiormente affermati e punto di partenza per estensioni future c'è senz'altro quello di *Mathis et al.*[10]. Esso si occupa di modellare un TCP generico, riconducibile a Reno con qualche aggiunta, in un ambiente con tasso di perdita (costante) ridotto, escludendo la possibilità di timeout ed assumendo RTT costante e trasferimento di massa (i.e. la sorgente ha sempre dati da inviare). Il throughput è definito in termini di $\sqrt{p}$*–law*, già citata precedentemente e spiegata in seguito, che lo mette in relazione con il tasso di perdita dei pacchetti. Un importante lavoro di rifinitura di questo modello è stato portato avanti da *Padhye et al.*[11, 12]. La versione di TCP analizzata è il Reno ed anche in questo caso il throughput è modellato in relazione al tasso di pacchetti persi ed al RTT, per sorgenti a trasferimento di massa. Tuttavia stavolta il tasso di perdita considerato può variare all'interno di range molto più ampi e viene introdotta la gestione dei timeout, rendendo il modello nettamente più realistico e accurato. Risultato principale di questo

lavoro è la cosiddetta formula *PFTK*[2], concetto estremamente significativo e ripetutamente usato in letteratura. Questi due modelli verranno adeguatamente approfonditi nella seguente sottosezione. *Kumar*[13] utilizza un approccio leggermente diverso modellando l'evoluzione ciclica della finestra di congestione per mezzo di *Discrete Time Markov Chain* (DTMC)[3]. Sono confrontate diverse versioni di TCP (Tahoe, Reno e NewReno) in uno scenario che introduce random loss su un tratto di canale wireless in una LAN. Lavoro simile a quello di Padhye et al. fu portato avanti da *Cardwell et al.*[14]. L'estensione della PFTK è motivata dall'introduzione del trasferimento di file di piccole dimensioni e dall'analisi degli effetti di inizializzazione del TCP (i.e. Slow Start e instaurazione della connessione), ritenuti fenomeni di grande impatto sulle prestazioni del TCP all'interno di reti come Internet. Autori di vari lavori significativi, anche *Kaj & Olsén*[15, 16] si ispirano ai lavori in [10, 11, 12], ma si concentrano in particolare su due versioni di TCP, Tahoe e NewReno, analizzando diversi aspetti che li caratterizzano sotto le usuali ipotesi di RTT costante, trasferimento persistente di massa, sorgente singola, etc. La differenza sostanziale emerge nell'analisi di due diverse politiche di AQM[4], e più precisamente nell'utilizzo di code RED affiancate a DropTail, che cambiano le regole del gioco influenzando notevolmente la probabilità di perdita dei pacchetti in corrispondenza di eventi di perdita. Più recente è l'opera di *D. J. Leith*[35] che indaga sulle interazioni che avvengono tra flussi TCP che competono per l'utilizzo del canale wireless. L'obiettivo principale

[2]Il nome deriva dalle iniziali dei cognomi degli autori: Padhye, Firoiu, Towsley, Kurose.

[3]Catena di Markov: processo stocastico che dato uno spazio degli stati del sistema assume dei valori al suo interno, passando da uno stato all'altro secondo una matrice di transizione i cui elementi indicano le probabilità di transizione. Tale processo gode della proprietà di Markov secondo cui il passaggio da uno stato ad un altro dipende unicamente dallo stato immediatamente precedente.

[4]Active Management Queue: politiche di gestione delle code che utilizzano diverse metriche di misurazione della congestione e metodi di scarto dei pacchetti al fine di informare il mittente sullo stato del canale. I due tipi di AQM maggiormente conosciuti sono *DropTail* e *RED*, due varianti speculari in termini di regime operativo (rispettivamente attivo e passivo).

consiste nel calcolare e valutare due caratteristiche fondamentali in ambito di reti Wi-Fi: il fairness, cioè l'equità del throughput per i flussi delle diverse sorgenti, ed il responsiveness, cioè il tasso di convergenza o reattività con cui il sistema tende a raggiungere il punto di equilibrio. Lo scenario considerato consiste in una rete WAN in modalità infrastruttura, collegata ad una LAN in cui la variabilità di banda del link wired permette di spostare il collo di bottiglia del sistema tra il canale wired e wireless, cosicché entrambi i casi possano essere testati in situazioni di upload/download e sotto diversi valori di parametri per strategie di Additive Increase/Multiplicative-Decrease (AIMD), utilizzate essenzialmente per il Congestion Avoidance.

Si discostano lievemente dai lavori finora descritti, quelli basati sull'indagine di ambienti DiffServ[5], che si poggiano sui risultati dei precedenti ed altri modelli utilizzandoli come building block. *Yeom & Reddy*[17] modellano il throughput in funzione del tasso contrattuale, del tasso di perdita e del RTT. Poggiandosi sulla formula PFTK la estendono considerando flussi aggregati e caratterizzando il traffico in base a livelli di priorità (Two-Drop e Three-Drop Precedence) che smistano il traffico scartando determinati pacchetti sulla base del traffico medio di invio. Il modello di *Malouch & Liu*[18] propone un meccanismo di "bucket token" per la marcatura dei pacchetti (IN e OUT) sui bordi della rete al fine di ricavare, tramite la consueta formula PFTK, un'espressione in closed form che relazioni tasso di perdita e parametri di configurazione per DiffServ, assumendo trasferimento di massa e politica di accodamento RED.

Altri lavori consultati in maniera superficiale e non centrali per questa tesi non sono menzionati, ma risultano comunque presenti nella bibliografia per consultazione ed approfondimenti.

[5]Differential Service: architettura che fornisce differenti livelli di servizi mantenendo la QoS del sistema.

4.3.2 Fixed-Point, Fluid, Processor Sharing e Control-Theoretic in letteratura

Uno dei lavori iniziali molto interessante, nonostante la sua semplicità e genericità, è quello di *Misra et al.*[19], in cui si considerano flussi TCP multipli e persistenti che interagiscono tramite canali single-bottleneck e politiche di accodamento RED. L'evoluzione della finestra di congestione è modellata tramite DTMC e sono esclusi diversi comportamenti specifici del TCP come Slow Start, timeout e dimensione massima della finestra. Basandosi sul lavoro appena citato, *Firoiu & Borden*[20] considerano uno senario simile ma si concentrano maggiormente sull'analisi dei parametri che influenzano le code di tipo RED al fine di fornire delle valutazioni sul miglior tipo di configurazione per la stabilità del sistema, utilizzando un approccio sistematico. Ulteriori estensioni al lavoro in [19] furono proposte da *Bu & Towsley*[21] e *Firoiu et al.*[22] che modificarono diversi aspetti, tra cui trasferimento di file di dimensione finita, marcatura dei pacchetti, DiffServ e flussi UDP, ed in particolare adattarono il modello all'introduzione di trasferimenti persistenti multipli su topologie di rete arbitrarie. *Casetti & Meo*[23, 24] utilizzano un approccio che si discosta leggermente dai precedenti, sfruttando due diversi modelli per sorgenti TCP e rete, e combinandoli insieme per trovare il regime operativo stazionario della rete. Specificatamente, il traffico verso la rete, catalogato come ON/OFF (i.e. sistema BUSY/IDLE, in cui le sorgenti generano o meno traffico), è modellato tramite catene di Markov a tempo continuo, mentre ogni nodo della rete è rappresentato da un modello a coda. Le versioni di TCP trattate sono Tahoe e Reno, e la soluzione di punto fisso è trovata attraverso un'analisi iterativa dei due modelli, e fornisce risultati molto accurati ma richiede sforzi e tempi elevati.

In ambito di Fluid method, *Misra et al.*[25] utilizzano le SDE[6] per modellare l'interazione tra sorgenti TCP persistenti e code RED, in cui il traffico è modellato come un fluido e gli eventi di perdita sono descritti da processi

[6]Stochastic Differential Equations: equazioni differenziali, cioè relazioni tra funzione non nota e sue derivate, in cui almeno un termine risulta essere un processo stocastico.

di Poisson. La topologia della rete considerata è arbitraria e può contenere colli di bottiglia multipli. *Altman et al.*[26] si focalizzano principalmente su come le diverse proprietà statistiche legate agli eventi di perdita influenzino le prestazioni di un TCP generico con sorgenti persistenti. Diversamente, *Altman et al.* in [27] analizzano una rete in cui due sorgenti, che competono per la sua capacità totale, sono assunte essere asimmetriche, nel senso che soltanto una delle due risente degli eventi di perdita modificando di conseguenza la propria finestra di congestione. Formule esplicite di throughput sono ricavate quando le sorgenti hanno stesso RTT, altrimenti sono proposti delle approssimazioni basate su upper e lower bound.

Per quanto riguarda i modelli Processor Sharing, il primo di essi fu introdotto da Kleinrock negli anni '60 per modellare il time sharing nei mainframe multi-utente. *Riedl et al.*[28] presentano un modello che può essere usato per il dimensionamento delle reti con traffico elastico (i.e. adattivo) basato su TCP. Per il calcolo della capacità della rete è utilizzato un modello a coda M/G/R esteso per tenere conto dei vari aspetti del TCP. *Fredj et al.*[29] studiano reti in cui sorgenti simmetriche ed omogenee condividono la capacità del canale in modo equo. Si assume che un numero casuale di flussi siano raggruppati in sessioni utente i cui tempi di arrivo sono processi di Poisson; dunque il throughput è valutato in funzione del processo degli arrivi. *Vranken et al.*[30] analizzano il comportamento del TCP a livello di flusso tramite un modello Processor Sharing multiple-server in cui il traffico TCP è diviso in due classi di priorità. Obiettivo principale è affrontare il problema del dimensionamento di reti TCP/IP che supportino QoS DiffServ su link single-bottleneck utilizzando un sistema a coda M/G/C.

Per concludere, relativamente all'approccio Control-Theoretic, *Low et al.*[31, 32] presentano inizialmente un'ottimizzazione teorica del controllo del flusso in cui sorgenti e link tentano di massimizzare l'utilizzo totale della rete, in contesti sincroni ed asincroni. In secondo luogo propongono un'implementazione pratica, utilizzando un algoritmo di AQM chiamato Random Exponential Marking (REM) che porta una stabilità crescente nella rete ed

equità tra le sorgenti, dimostrando grande scalabilità nei confronti di variazioni di capacità della rete e ritardi di propagazione. Ulteriore estensione dei precedenti lavori fu introdotta da *Low*[33] dove diverse versioni di TCP ed algoritmi di AQM vengono combinati per mostrare i vari comportamenti. *Hollot et al.*[34] svilupparono invece un insieme di linee guida per il dimensionamento in ambienti RED, considerando sorgenti TCP, ritardi della rete e controller RED, al fine di raggiungere la stabilità nel sistema di feedback control.

4.4 Lavori maggiormente considerati

Per raggiungere lo scopo prefissato sono stati presi in considerazione e valutati, con grado variabile di approfondimento, diverse decine di articoli scientifici e pubblicazioni ed alcuni lavori di tesi. Non tutti però hanno contribuito in maniera sostanziale alla scelta del modello finale, essenzialmente a causa dei vincoli dettati dal sistema modellato. Di seguito verrà descritto un insieme di modelli scelti tra quelli analizzati e considerati i più significativi a livello di contributo attivo o passivo, inteso come influenza ed applicabilità nel contesto considerato.

Ad eccezione dei lavori relativi al metodo Renewal Theory, gli altri non verranno descritti, non perché meno importanti nella comunità scientifica ma poiché non centrali al raggiungimento dell'obiettivo finale.

4.4.1 *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*

Il lavoro di *Mathis et al.*[10] è analizzato di seguito.

OBIETTIVO: modellare le performance dell'algoritmo Congestion Avoidance per un TCP generico in sistemi end-to-end.

ASSUNZIONI: distribuzione stazionaria della finestra di congestione per una CA ideale; tasso di perdita dei pacchetti moderato, periodico e co-

stante; sorgenti persistenti e trasferimento di massa; RTT costante; finestra di congestione con crescita lineare e riduzione esponenziale costanti rispettivamente prima e dopo un evento di perdita (AIMD); pacchetti indipendenti fino al primo evento di perdita; perdita multipla trattata come unica perdita; timeout non modellati (recovery sufficientemente breve).

MODELLO: definita p la probabilità costante di perdita, durante ogni RTT i pacchetti consegnati sono circa $\frac{1}{p}$ e la dimensione della finestra acquisisce una forma a "dente di sega" periodica nel tempo. Fissata W la dimensione massima della finestra di congestione in pacchetti, durante i periodi liberi da perdite la finestra crescerà di $\frac{1}{W}$ per ogni ack ricevuto (quindi di 1 ogni W ACK) mentre verrà immediatamente dimezzata al verificarsi di un evento di perdita. Raggiunto lo stato di equilibrio, la dimensione minima della finestra sarà dunque $\frac{W}{2}$.

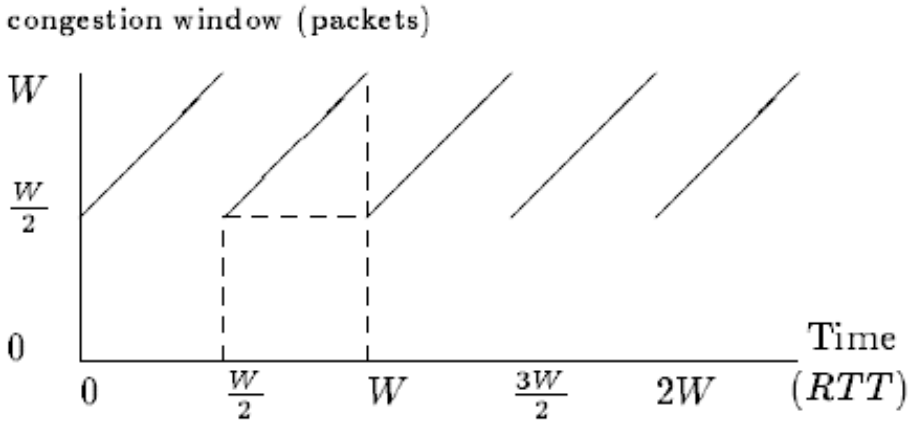

Figura 4.1: andamento finestra di congestione nel tempo

Ispezionando la Figura 4.1 si nota che ogni ciclo (periodo tra due eventi di perdita) è lungo $\frac{W}{2}$ RTT. Durante ogni ciclo il numero di pacchetti consegnati corrisponde all'area sottesa al dente di sega, data da $(\frac{W}{2})^2 + \frac{1}{2} \cdot (\frac{W}{2})^2 = \frac{3}{8} \cdot W^2$. Per ipotesi si ha che in ogni ciclo sono inviati $\frac{1}{p}$ pacchetti, quindi $\frac{3W^2}{8} = \frac{1}{p}$, da cui:

$$W = \sqrt{\frac{8}{3p}} \tag{4.1}$$

Dall'equazione (4.1) si ricava il throughput (TP), definito come quantità di dati per ogni ciclo (in bytes) diviso la durata del ciclo (in secondi):

$$TP = \frac{MSS \cdot \frac{3}{8} \cdot W^2}{RTT \cdot \frac{W}{2}} = \frac{MSS \cdot \frac{1}{p}}{RTT \cdot \sqrt{\frac{2}{3p}}} \tag{4.2}$$

Raggruppando le costanti in un unico termine, $C = \sqrt{\frac{3}{2}}$, si ottiene la $\frac{C}{\sqrt{p}}$–*law*:

$$TP = \frac{MSS \cdot C}{RTT \cdot \sqrt{p}} \tag{4.3}$$

La costante di proporzionalità C varia in base alle diverse considerazioni effettuate, come la versione del TCP, la gestione degli ACK, i meccanismi di perdita, etc.

VALUTAZIONE: le simulazioni sono effettuate tramite il simulatore di rete *ns versione 1* (ns-1). Lo scenario principale consiste in due nodi che implementano il TCP congiunti da un link associato alle proprietà del canale (code, ritardi, banda, etc). Il primo esperimento considera ritardo, MSS e tasso di perdita costanti, su link senza code. Vengono simulate diverse conmbinazioni di valori e disegnati grafici di confronto tra i risultati delle simulazioni e le stime del modello. Per tassi di perdita molto piccoli ($p < 0.01$) le stime del modello risultano plausibili, altrimenti lo scadere dei timeout rendono le stime notevolmente inaccurate poiché la fase di CA risulta meno predominante. Introducendo la gestione del link come coda DropTail e simulando un TCP Reno, si notano risultati simili ai precedenti, fuorché per alcune eccezioni dovute alle attese di accodamento che inibiscono l'algoritmo di Delayed ACK. Altre valutazioni considerano le versioni di TCP messe a disposizione dal simulatore, i test live di trasferimento da siti internet, code RED, ed altro ancora, dimostrando come per moderati tassi di perdita il modello resti accurato, specialmente nei casi di random loss ed esclusione di timeout.

4.4.2 *Modeling TCP Throughput: A Simple Model and its Empirical Validation*

Il lavoro di *Padhye et al.*[11] è analizzato di seguito.

OBIETTIVO: creare un modello analitico per il throughput di un TCP Reno che consideri i timeout e valori più ampi per il tasso di perdita.

ASSUNZIONI: sorgenti persistenti e trasferimento di massa; TCP Reno (3 ACK duplicati (TD) e timeout (TO) con exponential backoff); ogni ciclo è suddiviso in round (tempo intercorso tra l'invio di W pacchetti e la ricezione del primo ACK di conferma, con W dimensione corrente della finestra di congestione) lunghi esattamente un RTT indipendentemente da W; delayed ACK ogni b pacchetti; perdita dei pacchetti indipendente tra i round ma correlata in ogni round (politica DropTail); tempo trascorso in Slow Start trascurabile; throughput misurato in pacchetti inviati (i.e. comprese perdite).

MODELLO: dato N_t il numero di pacchetti trasmessi nell'intervallo $[0, t]$ e $B_t = \frac{N_t}{t}$ il throughput su tale intervallo, si definisce B il throughput di stato stazionario su lungo termine come:

$$B = \lim_{t\to\infty} B_t = \lim_{t\to\infty} \frac{N_t}{t} \tag{4.4}$$

Definita la probabilità di perdita dei pacchetti p, si vuole stabilire una relazione tra p e B, $B(p)$. Si esaminano inizialmente solo gli indicatori di perdita basati su TD: sia TDP_i l'i-esimo periodo di TD (i.e. periodo tra due eventi di TD); siano Y_i il numero di pacchetti inviati in TDP_i, A_i la durata di TDP_i e W_i la dimensione della finestra alla fine del TDP_i; considerando $\{W_i\}_i$ un processo rigenerativo di Markov[7] con ricompensa $\{Y_i\}_i$. Si può mostrare che:

$$B = \frac{E[Y]}{E[A]} \tag{4.5}$$

[7]Processo rigenerativo: processo stocastico che ha proprietà di ripetitività in un tempo aleatorio. Tale processo può quindi essere suddiviso in cicli indipendenti ed identicamente distribuiti (i.i.d.). Un processo rigenerativo di Markov è una generalizzazione di un processo rigenerativo.

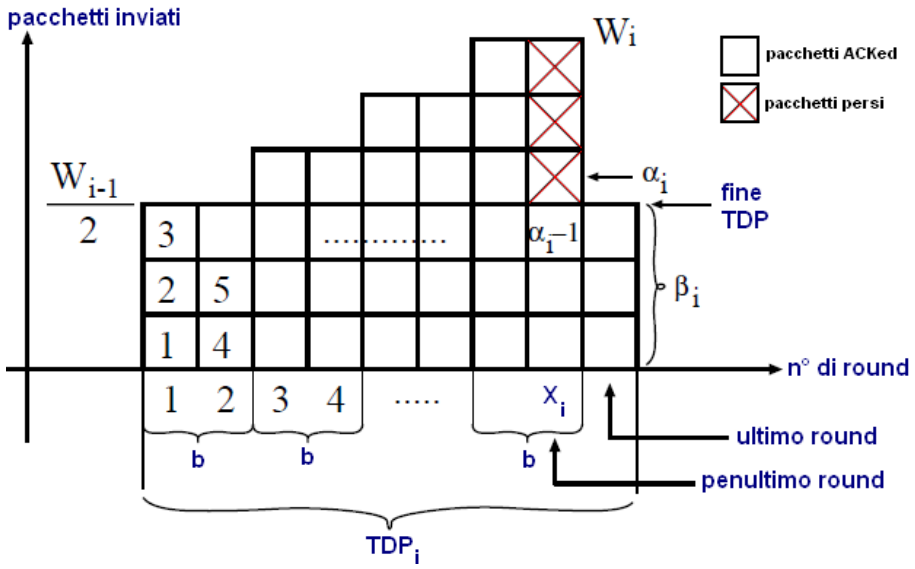

Figura 4.2: evoluzione della finestra di congestione durante un TDP

La Figura 4.2 mostra un esempio di evoluzione della finestra di congestione durante un TDP; la finestra viene incrementata di $\frac{1}{b}$ ogni round (dovuto ad ACK cumulativi o delayed) e dimezzata alla fine del TDP. X_i indica il round in cui occorre l'evento di perdita, α_i indica il numero di pacchetti inviati fino al primo perso (incluso), mentre β_i corrisponde al numero di pacchetti inviati l'ultimo round. Ispezionando la Figura 4.2 si deduce che in $X_i + 1$ round sono inviati $Y_i = \alpha_i - 1 + W_i$ pacchetti, e poiché α ha distribuzione geometrica[8] di media $\frac{1}{p}$, si ha:

$$E[Y] = E[\alpha] - 1 + E[W] = \frac{1-p}{p} + E[W] \tag{4.6}$$

Per derivare $E[A]$ si nota che la durata di un TDP_i è data dalla sommatoria dei round trip time di tutti i round; definendo RTT il valore medio di round trip time, segue che:

$$E[A] = RTT \cdot (E[X] + 1) \tag{4.7}$$

[8]Distribuzione Geometrica: distribuzione di probabilità discreta sui numeri naturali che indica la probabilità che un evento si verifichi all' i-esimo tentativo: $P(i) = (1-p)^{i-1} \cdot p$, dove $1 - p$ indica la probabilità di insuccesso.

Osservando che W aumenta tra $\frac{W_{i-1}}{2}$ e W_i durante l'i-esimo TDP, introducendo aclune assunzioni ed approssimazioni, si ricava:

$$E[W] = \frac{2}{b} \cdot E[X] \tag{4.8}$$

Facendo uso di diverse manipolazioni algebriche e sostituzioni (non riportate poiché esulano dallo scopo proposto), si ottengono:

$$E[W] = \sqrt{\frac{8}{3bp}} + o\left(\frac{1}{\sqrt{p}}\right) \tag{4.9}$$

$$E[X] = \sqrt{\frac{2b}{3p}} + o\left(\frac{1}{\sqrt{p}}\right) \tag{4.10}$$

da cui:

$$\begin{aligned} B(p) &= \frac{\frac{1-p}{p} + E[W]}{RTT \cdot (E[X]+1)} \\ &= \frac{1}{RTT} \cdot \sqrt{\frac{3}{bp}} + o\left(\frac{1}{\sqrt{p}}\right) \end{aligned} \tag{4.11}$$

approssimata a $B(p) \approx \frac{1}{RTT} \cdot \sqrt{\frac{3}{bp}}$ per piccoli valori di p.

La situazione si complica nel momento in cui subentra la gestione del timeout. In questo caso l'equazione (4.5) diventa:

$$B = \frac{E[M]}{E[S]} = \frac{E[Y] + Q \cdot E[R]}{E[A] + Q \cdot E[Z^{TO}]} \tag{4.12}$$

dove Z^{TD} e Z^{TO} sono rispettivamente la durata di una sequenza di timeout e la durata tra due sequenze di timeout (i.e. intervallo tra la fine di uno o più TO consecutivi e l'inizio di un altra sequenza di TO), $S_i = Z^{TD} + Z^{TO}$ indica l'intervallo di tempo tra la fine di una sequenza di timeout e la fine della successiva sequenza di timeout, Q è la probabilità che si verifichi un timeout alla fine di un TDP ed R denota il numero di pacchetti inviati durante una sequenza di timeout Z^{TO}.

Un esempio di evoluzione della finestra di congestione in presenza di eventi di TD e TO è mostrata in Figura 4.3. T_0 indica il valore del timeout, cioè il

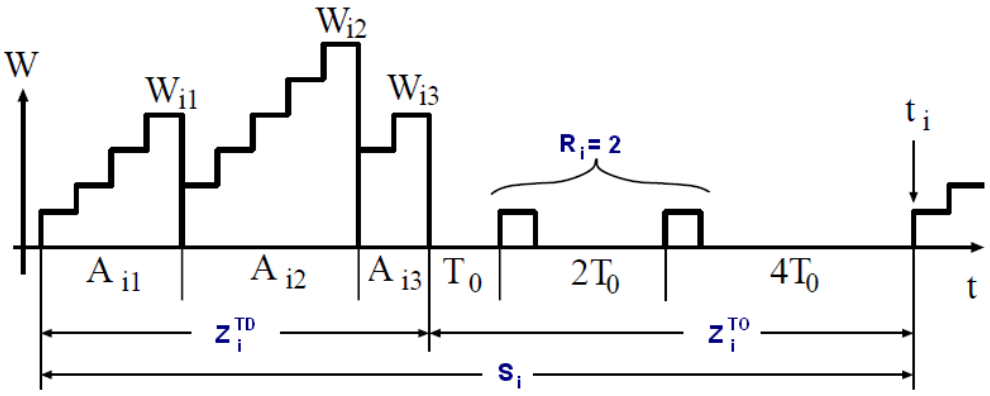

Figura 4.3: esempio di evoluzione della finestra considerando TD e TO

periodo di tempo che il mittente attende prima di ritrasmettere i pacchetti non confermati; nel caso in cui i pacchetti non vengano nuovamente ricevuti, il valore del timeout è raddoppiato per ogni ritrasmissione senza successo, fino al valore massimo di $64 \cdot T_0$, dopodiché la lunghezza resta fissa a 64. Per derivare Q è necessaria un'analisi approfondita del penultimo ed ultimo round prima di un segnale di perdita, dalla quale si ricava numericamente una soluzione approssimata per Q:

$$Q \approx \min\left(1, \frac{3}{E[W]}\right) = \min\left(1, 3 \cdot \sqrt{\frac{3bp}{8}}\right) \tag{4.13}$$

Per derivare $E[R]$ si nota che in genere soltanto un pacchetto è inviato tra due timeout ed il numero di timeout in una sequenza di TO è distribuito geometricamente, quindi ha media $\frac{1}{1-p}$. $E[Z^{TO}]$ si deriva sapendo che i primi 6 timeout in una sequenza hanno lunghezza $2^{i-1} \cdot T_0$ con $i = 1...6$, ed i successivi valgono $64T_0$, quindi la durata di una sequenza con k timeout è

$$L_k = \begin{cases} (2^k - 1) \cdot T_0, & \text{con } k \leq 6 \\ (63 + 64 \cdot (k-6)) \cdot T_0, & \text{con } k \geq 7. \end{cases}$$

ed il valore medio di Z^{TO} risulta

$$\begin{aligned} E[Z^{TO}] &= \sum_{k=1}^{\infty}(L_k \cdot P[R=k]) \\ &= T_0 \cdot \left(\frac{1+p+2p^2+4p^3+8p^4+16p^5+32p^6}{1-p}\right) \end{aligned} \tag{4.14}$$

Combinando i risultati ottenuti si ricava infine la relazione per il throughput in funzione del tasso di perdita, approssimata da

$$B(p) \approx \frac{1}{RTT \cdot \sqrt{\frac{2bp}{3}} + T_0 \cdot min\left(1, 3 \cdot \sqrt{\frac{3bp}{8}}\right) \cdot p \cdot (1+32p^2)} \tag{4.15}$$

Come ultima considerazione, si vuole vincolare la dimensione massima della finestra di congestione in base al limite del buffer del ricevente, affinché nei periodi liberi da perdite la finestra non cresca in maniera eccessiva. Definito W_{max} il valore massimo della finestra di congestione, l'equazione (4.15) diventa

$$B(p) \approx min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT \cdot \sqrt{\frac{2bp}{3}} + T_0 \cdot min\left(1, 3 \cdot \sqrt{\frac{3bp}{8}}\right) \cdot p \cdot (1+32p^2)}\right) \tag{4.16}$$

VALUTAZIONE: il modello è validato empiricamente tramite misurazioni di dati effettuate su 37 connessioni TCP stabilite tra 18 host, con diversi domini e sistemi operativi, sparsi negli Stati Uniti ed in Europa. 24 delle 37 connessioni TCP sono testate ognuna per un'ora mentre le rimanenti soltanto 100 secondi, con vuoti di 50 secondi tra la fine di una e l'inizio della successiva. I dati sono analizzati da programmi sviluppati dagli autori ed ulteriormente verificati dai simulatori *tcptrace* e *ns*. Emerge che in ogni prova i timeout costituiscono una parte significativa del numero totale degli indicatori di perdita (exponential backoff inclusi). Ogni connessione è testata con valori specifici per i parametri di RTT (medio), TO (medio) e W_{max}, ed i risultati vengono confrontati con quelli del modello precedente [10], evidenziando notevoli miglioramenti nelle stime del throughput, specialmente in presenza di timeout.

4.4.3 *Stochastic Equilibrium Modeling of the TCP Dynamics in Various AQM Environments*

Il lavoro di *I. Kaj, J. Olsén*[16] è analizzato di seguito.

OBIETTIVO: valutare le dinamiche di due versioni di TCP (Tahoe e NewReno) in ambienti che utilizzano due diverse politiche di AQM (RED e DropTail).

ASSUNZIONI: sorgente singola persistente e trasferimento di massa; RTT costante; analisi in cicli (intervallo tra due riduzioni della finestra di congestione) che consistono in un numero casuale di round (intervallo tra l'invio di W pacchetti e la ricezione del primo ACK di conferma o timeout); throughput definito come tasso asintotico di trasmissioni di successo su lunghi peiodi (in pacchetti).

MODELLO: questo modello è descritto approfonditamente nel Capitolo 6 poiché scelto come modello di analisi.

VALUTAZIONE: i risultati sono ottenuti utilizzando il simulatore *ns-2* ed effettuando trasferimenti di file FTP long-lived. Gli scenari considerati sono due. Il primo consiste in un ambiente ideale formato da una connessione WAN persistente in cui i dati sono inviati dalla sorgente A alla destinazione D attraverso due gateway B e C che formano il collo di bottiglia e sono configurati per agire in maniera duplice, soddisfando in un caso la politica RED e nell'altro quella DropTail. Il secondo scenario delinea una topologia di rete più realistica composta da 62 nodi eterogenei e traffico misto. I file sono trasferiti tra i nodi A_1 e D_1, A_2 e D_2, ... , A_{30} e D_{30} ed ogni connessione condivide un link bottleneck tra i nodi B e C, in cui si utilizza la politica RED o DropTail. Inoltre i nodi sono divisi in tre gruppi in base al ritardo di propagazione dei link non condivisi (10, 30 e 50 ms). Nello scenario ideale, il modello mostra un'accuratezza elevata nella stima del throughput per tassi di perdita p appartenenti al range (0.001 ... 0.35). Si nota che la politica di accodamento ha forte impatto sulle occorrenze dei timeout, decisamente più numerose per DropTail rispetto a RED; inoltre NewReno mostra un'attitudine maggiore ai timeout rispetto a Tahoe (dovuto eviden-

temente alla fase di ritrasmissione di tutti i pacchetti persi). Si sottolinia anche come una scelta grossolana degli intervalli di timeout possa avere un impatto molto negativo sulle prestazioni. In generale, il throughput stimato rispecchia quello simulato e mostra che in ambiente RED NewReno e Tahoe hanno un comportamento simile, diversamente da quanto accade invece in DropTail in cui NewReno risulta maggiormente penalizzato. Nello scenario realistico, la stima di Q per Tahoe è molto buona con RED, sia in termini di accuratezza che di vicinanza ai risultati simulati. Utilizzando invece DropTail il grafico mostra una stima molto alta che si discosta significativamente dai risultati delle simulazioni (comunque aventi valori nettamente superiori rispetto al caso RED); ciò è motivato dal fatto che l'assunzione di scartare tutti i pacchetti consecutivi alla prima perdita è considerata eccessivamente pessimistica. Circa la stima del throughput, c'è una chiara distinsione tra DropTail e RED; infatti per quest'ultimo viene stimato accuratamente come nello scenario ideale, mentre per DropTail risulta sottostimato per valori di p minori del 3 percento. I risultati relativi al NewReno in questo scenario non sono riportati. Infine, si osserva dalle simulazioni che l'assunzione di indipendenza dei pacchetti per RED è meno sensibile alla topologia e al traffico misto rispetto a DropTail.

4.4.4 *Modelling TCP Dynamics in Wireless Networks*

Il lavoro di *D. J. Leith*[35] è analizzato di seguito.

OBIETTIVO: sviluppare un modello analitico per il comportamento di flussi TCP concorrenti in reti wireless, considerando l'impatto delle loro interazioni sull'equità del sistema.

ASSUNZIONI: N stazioni wireless concorrenti ognuna con n flussi; stazioni destinazione wireless o wired ma almeno un hop wireless condiviso che funge da collo di bottiglia; RTT arbitrario; strategia $AIMD$ per CA; perdite causate da buffer overflow della coda sul bottleneck link o random loss sul canale wireless; i buffer delle stazioni wireless non si svuotano mai (ci sono sempre dati da inviare); il tasso di servizio del canale ed il tasso di

perdita sono stocasticamente indipendenti dalla dimensione della finestra dei flussi individuali; flussi della stessa sorgente condividono lo stesso buffer di accodamento (bottleneck) mentre flussi di sorgenti diverse buffer separati.

MODELLO: data la sorgente $s \in [1, N]$ ed il flusso $i \in [1, n]$ si denota con $w_{s,i}(k)$ il valore della finestra di congestione, del flusso i per la sorgente s, all'inizio della fase di CA, nel k-esimo periodo di CA. La strategia $AIMD$ durante CA prevede un incremento della finestra di $\alpha_{s,i}$ pacchetti per ogni RTT fino alla prima perdita, a cui segue una riduzione di $\beta_{s,i}$. L'RTT del flusso i varia in base a $RTT_{s,i} = T_{d_{s,i}} + \frac{(1+q_s(t))}{B_s(t)}$, in cui $T_{d_{s,i}}$ denota il ritardo di propagazione del tratto wired, $B_s(t)$ indica il tasso di servizio della stazione s al tempo t e $q_s(t)$ conta il numero di pacchetti nella coda della stazione s ($0 \leq q_s(t) \leq q_{s,max}$). In caso di congestione, non necessariamente tutte le stazioni sono affette da perdite. Da quanto detto si ricava per la regola di $AIMD$:

$$w_{s,i}(k+1) = b_{s,i}(k) \cdot w_{s,i}(k) + a_{s,i}(k) \cdot T_s(k) \tag{4.17}$$

dove $b_{s,i}(k) \in \{\beta_{s,i}, 1\}$ corrisponde a $\beta_{s,i}$ se i ha una perdita nel periodo k, 1 altrimenti, $a_{s,i}(k)$ è l'incremento effettivo (pacchetti al secondo) e $T_s(k)$ è la durata del k-esimo periodo di CA. Valutando inizialmente il caso in cui il canale sia esente da rumore (i.e. perdite causate da buffer overflow), si ha che il throughput aggregato per s nel periodo k deve essere uguale al tasso di servizio:

$$B_s(k) = \sum_{i=1}^{n_s} \frac{w_{s,i}(k)}{RTT_{s,i,max}(k)} \tag{4.18}$$

in cui max indica la coda piena (i.e. buffer overflow) e di conseguenza fine del k-esimo periodo di CA. Segue che

$$B_s(k+1) = \sum_{i=1}^{n_s} \frac{b_{s,i}(k) \cdot w_{s,i}(k) + a_{s,i}(k) \cdot T_s(k)}{RTT_{s,i,max}(k)} \tag{4.19}$$

Poiché per ipotesi si ha che il tasso di servizio (e in particolare la sua media) è indipendente dal periodo di congestione, fissato $\gamma_{s,i} = \frac{1}{E[RTT_{s,i,max}]}$, si ottiene

$$\begin{aligned} E[B_s] &= \sum_{i=1}^{n_s} \gamma_{s,i} \cdot E[w_{s,i}(k)] \\ &= \sum_{i=1}^{n_s} \gamma_{s,i} \cdot (E[b_{s,i}] \cdot E[w_{s,i}(k)] + E[a_{s,i}] \cdot E[T_s]) \\ &= E[B_s] \end{aligned} \tag{4.20}$$

con indipendenza tra $b_{s,i}$ e w_i, $a_{s,i}$ e T_s, ed entrambe $b_{s,i}$ e $a_{s,i}$ indipendenti da k. Ricavando $E[T_s]$ da (4.20), sostituendo a (4.17) e utilizzando (4.20) si ha il throughput per un flusso singolo a fine CA:

$$\begin{aligned} \gamma_{s,i} \cdot E[w_{s,i}(k+1)] = E[b_{s,i}] \cdot \gamma_{s,i} \cdot [w_{s,i}(k)] + \frac{\gamma_{s,i} \cdot a_{s,i}}{\sum_{j=1}^{n_s} \gamma_{s,j} \cdot E[a_{s,j}]} \\ \cdot \left(\sum_{j=1}^{n_s} (1 - E[b_{s,i}]) \cdot \gamma_{s,i} \cdot E[w_{s,i}(k)] \right) \end{aligned} \tag{4.21}$$

e le dinamiche di una collezione di flussi sono descritte da

$$W_s(k+1) = A_s \cdot W_s(k)^9 \tag{4.22}$$

D'altro canto, se ci fossero random loss sul canale wireless, le code non sarebbero piene al verificarsi di una congestione, e, per assunzione, non sarebbero mai vuote. Il throughput aggregato dei flussi resta comunque uguale al tasso di servizio, $\sum_{i=1}^{N_s} \frac{w_{s,i(k)}}{RTT_{s,i}(k)} = B_s(k)$, in cui però nel RTT il valore di $q_s(t)$ non è più il massimo e la sua media varia in modo inversamente proporzionale al tasso di perdita.

VALUTAZIONE: le simulazioni sono effettuate attraverso il simulatore *ns* e come individuato anche in precedenti studi empirici, la scarsa equità tra i throughput raggiunti dai flussi concorrenti è evidente (causa principale è l'interazione negativa tra il meccanismo di contesa gestito dal MAC e quello di controllo della congestione del TCP). Tuttavia sono proposte alcune azioni per allentare questo conflitto e ripristinare l'equità basate sulla prioritarizzazione degli ACK sugli AP e sulle stazioni wireless e sul meccanismo *TXOP*

[9] Vedere Appendice A, sezione A.2 per i dettagli.

(Transmit Opportunity). La topologia di rete considerata è formata da una WAN connessa ad una LAN in infrastructure mode, con 4 nodi sorgenti (2 wired e 2 wireless). Sono considerati download ed upload e la banda del link wired connesso all'AP può essere variata affinché questo agisca da collo di bottiglia o viceversa (i.e. wireless bottleneck). In caso di bottleneck wireless, si mostra come il variare dei parametri di $AIMD$ non influisce sull'equità del throughput per l'upload (bottleneck su stazioni wireless) mentre esso diventa proporzionale a $\frac{E[a_{s,i}]}{1-E[b_{s,i}]}$ per il download (bottleneck su AP). Differentemente, se il collo di bottiglia è il link wired, gli upload riducono significativamente l'equità come risultato dei differenti RTT. Altri test sono condotti relativamente al tasso di convergenza (responsiveness) nei vari casi. In definitiva, l'accuratezza elevata del modello è fermamente confermata dalle simulazioni.

Capitolo 5

VINCOLI E ASSUNZIONI

Il capitolo descrive i problemi affrontati e le decisioni prese durante lo svolgimento della tesi, esplicitando molti dei ragionamenti che hanno portato alla scelta di alcune strategie piuttosto che altre, al superamento di determinati ostacoli, ed infine allo schieramento di assunzioni e definizioni su cui il lavoro è stato edificato.

5.1 Vincoli e Limitazioni

I principali vincoli emersi durante la tesi possono essere raggruppati nelle 3 categorie descritte di seguito.

- Vincoli di sistema: l'architettura dei sistemi, le politiche di funzionamento del TCP e le modifiche apportate al sistema per ottenere la ritrasmissione anticipata hanno limitato la possibilità di scelta nella risoluzione di diversi problemi, conducendo in particolare all'utilizzo di un modello particolarmente ricco di contenuti, a politiche di ritrasmissione ed accodamento differenti, all'uso di assunzioni più stringenti.

- Vincoli e limiti dei modelli: la modellazione matematica, sebbene molto affascinante e potente, risulta estremamente insidiosa; la maggioranza dei modelli studiati sono scarsamente documentati e solitamente applicati a sistemi che lavorano in condizioni ideali. Una delle difficoltà

maggiori è stata quella di cogliere nei comportamenti ed interazioni dei sistemi piccole differenze e sfaccettature che provocassero cambiamenti significativi al fine di comprenderle, valutarle, approssimarle o comunque ricondurle a situazioni ragionevoli senza intaccare la validità e la qualità dei sistemi.

- Limiti temporali e conoscitivi: il tempo insieme alla limitata esperienza in ambito matematico e statistico, hanno giocato a sfavore del lavoro intrapreso smorzandone in parte gli ambiziosi obiettivi fissati inizialmente, prolungandone lo svolgimento e purtroppo non permettendo l'espletamento di aspetti tecnici prettamente matematici, che avrebbero richiedesto tempistiche e sforzi non idonei a quelli previsti per una tesi di laurea magistrale.

5.2 Assunzioni

Le principali scelte ed assunzioni che hanno condotto all'adozione di un modello che rappresentasse il sistema in maniera corretta ed efficace sono puntualizzate e motivate di seguito.

- Renewal Theory: ci si è focalizzati sul metodo che sfrutta la Teoria del Rinnovamento, dando importanza ad un meccanismo che potesse esprimere in maniera completa gli aspetti principali del TCP ed al contempo producesse un modello efficace.

- Trasferimento di massa: relativamente al metodo di modellazione scelto, l'assunzione di avere sorgenti persistenti con infiniti dati da inviare è di basilare importanza per testare il degrado di prestazioni durante periodi di traffico intenso nella rete; questo permette inoltre di astrarre dalle problematiche legate alla fase di set-up della connessione (e.g. Three-Way-Handshake) i cui tempi diventano trascurabili rispetto a quelli della fase di trasferimento vera e propria; infine rende possi-

bile parlare di assunzioni relative a stato di equilibrio del sistema e distribuzioni stazionarie per le variabili aleatorie implicate nel modello.

- TCP NewReno[1]: si assume che la versione TCP sia NewReno; tale versione, nonostante non sia la più recente, risulta certamente la più conosciuta, analizzata e matura dal punto di vista del funzionamento; inoltre la sua caratteristica di raccogliere all'incirca la totalità degli aspetti salienti per il controllo della congestione conduce ad un'analisi completa e qualitativa delle comunicazioni TCP.

- Throughput: rilevando un abuso di terminologia e grande confusione legata alla definizione di throughput in letteratura, si precisa che in questo contesto tale termine indica la "quantità di dati (in pacchetti) effettivamente inviati nell'unità di tempo (round)", riferendosi quindi ai soli pacchetti consegnati con successo al destinatario; tale quantità è calcolata in funzione del tasso di perdita e nello stato di equilibrio del sistema, considerando i valori medi delle variabili aleatorie implicate e sfruttando il teorema renewal-reward.

- Timeout e Fast Recovery: sia i meccanismi di timeout che di recovery sono entrambi considerati poiché aspetto di rilevante importanza nel controllo di congestione del TCP e parametri che giocano un ruolo indispensabile nell'analisi delle prestazioni.

- Finestra massima mittente: si è deciso di non limitare esplicitamente la finestra di congestione con una dimensione massima, lasciando coincidere nel migliore dei casi tale dimensione con quella supportata dal canale e delegando totalmente la responsabilità di gestione ai meccanismi di controllo del TCP.

[1]Inizialmente si era deciso di modellare le dinamiche della versione Cubic, poiché versione attualmente in uso per default su sistemi operativi linux; purtroppo principalmente per mancanza di documentazione e fonti sufficientemente adeguate si è dovuto abbandonare l'idea.

- DropTail vs RED: le due politiche di AQM, descritte nei capitoli precedenti, sono entrambe tenute in considerazione ed utilizzate durante l'analisi del sistema; queste infatti possono essere strettamente collegate ai due principali tipi di eventi di perdita che possono colpire il sistema: random loss e buffer overflow. In effetti le due politiche non hanno un confine ben separato in relazione a tali eventi ed in un certo senso coesistono; considerando per esempio random loss, si può attribuire una politica RED (che occasionalmente scarta dei pacchetti in previsione di congestionamento) a eventi di perdita isolati, mentre in altri casi si può attribuire una politica DropTail (che elimina tutti i pacchetti successivi alla prima perdita fino al recovery) a interi blocchi di perdite (frequenti in scenari wireless, per esempio in caso di shadowing, in cui degli ostacoli bloccano il segnale interrompendo la connessione).

- Sorgente singola: l'utilizzo di un sistema a sorgente singola è stata diretta conseguenza della scelta del metodo Renewal Theory; un'estensione futura del modello che supporti l'uso di sorgenti multiple è tuttavia ritenuta molto importante e discussa sinteticamente nel capitolo riguardante gli sviluppi futuri.

- RTT: il round trip time è supposto costante ed abbastanza lungo da permettere (in condizioni normali) l'invio di un'intera finestra di pacchetti e ovviamente la ricezione di almeno un primo ACK (cumulativo) di conferma; si può pensare a questa approssimazione come ad un valore medio in cui i singoli componenti variano in maniera casuale entro range contenuti ed i rari picchi isolati che fuoriescono da tale range non influenzano significativamente il valore atteso.

- Probabilità di perdita: come per l'RTT, anche la probabilità di perdita dei pacchetti si assume costante e generalmente acquisisce valori tipici all'interno di range di valori solitamente minori di 0.4 e stimati in base a risultati simulativi.

- ACK: gli ACK sono cumulativi e la loro dimensione molto piccola comporta una probabilità di perdita vicinissima allo zero; ciò permette di assumere ragionevolmente che la perdita degli ACK sia trascurabile.

- Coda AP: l'AP tiene traccia dei pacchetti ricevuti, bufferizzati ed inviati; nei casi in cui più ritrasmissioni dello stesso pacchetto raggiungano il buffer dell'AP, se questo ha già inoltrato il pacchetto in questione si assume che non venga nuovamente ritrasmesso ma scartato.

- Finestra massima end-point ricevente: come anticipato nel capitolo 2, assumendo che la parte wired della comunicazione non sia un collo di bottiglia, limitare esplicitamente la finestra di ricezione dell'end-point destinazione non avrebbe senso; ad ogni modo, prendendo spunto dal lavoro in [11], includere tale vincolo significherebbe considerare il minimo tra il throughput risultante e la massima dimensione della finestra calcolata nell'unità di tempo (i.e. $\frac{W_{max}}{RTT}$).

- Buffers canale wired: i buffer degli eventuali router e dell'end-point destinazione non vengono considerati colli di bottiglia poiché come già detto la parte wired del sistema si assume non essere un punto critico della comunicazione.

- Delay: i tipi di ritardi che possono crearsi durante la comunicazione (e.g. accodamento, trasmissione, accesso al canale, etc) risultano ininfluenti in quanto già considerati nella stima dell'RTT medio.

5.3 Scelta del modello

La fase di scelta del modello ha avuto come fase primaria la selezione del metodo di modellazione. Come precedentemente indicato, la scelta è ricaduta sul metodo basato sulla Renewal Theory per i motivi già discussi. In linea di massima, gli altri metodi sono stati esclusi per tre ragioni principali: inadeguatezza, incompletezza ed eccessiva espressività, che sfociavano in risultati

inappropriati, incompleti o estremamente complessi portando confusione e sovrabbondanza di informazioni superflue per gli obiettivi prefissati. Inoltre, una prima idea di modellare il sistema tramite reti di Petri aveva preso piede nella fase embrionale del lavoro, ma essenzialmente per scarsità di materiale e fonti, relativamente al TCP, l'opzione è stata immediatamente scartata.

Di forte impatto sono stati i modelli in [10] e [11] che sono serviti soprattutto per affacciarsi al mondo della modellazione del TCP e hanno aiutato sensibilmente alla formazione di un background di conoscenze basilari ma indispensabili per permettere la comprensione e l'utilizzo di modelli decisamente più complessi e raffinati, quale quello utilizzato per il modello classico.

Capitolo 6

IL MODELLO

Questo capitolo spiega dettagliatamente il modello utilizzato per l'analisi del sistema classico e descrive i comportamenti tipici ed i problemi chiave rilevati che hanno un forte impatto sulle prestazioni e sui cambiamenti del sistema. In secondo luogo, sulla base del metodo di analisi acquisito tramite il modello applicato al sistema classico, si affronterà una discussione incentrata sull'analisi del sistema anticipato, descrivendo le ripercussioni delle modifiche apportate, discutendo potenzialità e casi specifici di funzionamento, e delineando le linee guida per la rielaborazione e l'adattamento del modello al sistema anticipato.

6.1 Modello per il sistema classico

Come introdotto nel capitolo relativo allo stato dell'arte, il modello *Stochastic Equilibrium Modeling of the TCP Dynamics in Various AQM Environments*[16] si occupa di valutare le dinamiche di due versioni di TCP (Tahoe e NewReno) in ambienti che utilizzano due diverse politiche di AQM (RED e DropTail). Alla luce dei vincoli e delle assunzioni analizzati finora, tale modello si applica perfettamente al sistema classico. Segue una descrizione dettagliata del modello proposto da *I. Kaj & J. Olsén*, a cui ci si riferirà d'ora in avanti con *modello classico*.

6.1.1 Introduzione al modello

Si riepilogano assunzioni e definizioni fondamentali utilizzate:

- sorgente singola e trasferimento di massa;
- probabilità di perdita e RTT costante;
- analisi in funzione dei cicli, definiti come intervallo tra due riduzioni della finestra di congestione (i.e. timeout (TO) o fast recovery (FR));
- ogni ciclo è formato da un numero variabile di round (di lunghezza 1 RTT) definiti come l'intervallo tra l'invio di W pacchetti e la ricezione del primo ACK cumulativo o lo scadere di un TO (i.e. un round termina allo scorrimento della sliding window);
- il throughput è definito come il tasso asintotico di trasmissioni riuscite su lunghi periodi (misurato in pacchetti).

Similmente al lavoro in [11], lo studio di ogni ciclo n è legato all'analisi delle seguenti variabili aleatorie.

- A_n: il numero di pacchetti trasmessi con successo nel ciclo fino alla prima perdita (esclusa).
- B_n: il numero di pacchetti trasmessi con successo nel ciclo dopo la prima perdita (esclusa) e fino all'inizio del FR (ricezione terzo DUPACK) o il verificarsi di un TO.
- J_n: indica se il ciclo termina con un FR riuscito ($J_n = 1$) o un TO ($J_n = 0$).
- Y_n: il numero di round da inizio ciclo fino al round in cui si verifica la prima perdita (incluso).
- Z_n^{FR}: il numero di round dal round successivo alla prima perdita fino alla terminazione del FR (con successo).

- Z_n^{TO}: il numero di round dal round successivo alla prima perdita fino allo scadere di un TO.

- Z_n: il numero di round del ciclo, $Z_n = Y_n + Z_n^{FR} + Z_n^{TO}$; per ogni ciclo, uno soltanto tra Z_n^{FR} e Z_n^{TO} si verifica (i.e. > 0).

- K_j: il numero di cicli tra due TO consecutivi, occorsi al tempo $j-1$ e j.

6.1.2 Throughput generico

I TO sono punti di rigenerazione per le dinamiche della finestra che viene resettata alla dimensione iniziale, quindi attraverso l'analisi renewal-reward si ricava la relazione di throughput. Utilizzando le variabili definite si possono calcolare facilmente il numero di round tra due TO consecutivi, il numero di pacchetti inviati con successo tra due TO consecutivi ed il numero medio di pacchetti inviati con successo tra due TO consecutivi prima dell'evento di perdita (descritti dettagliatamente nella sottosezione A.1.1 dell'appendice A).

Assumendo il processo di rinnovamento stazionario, K_∞, ed il verificarsi di TO entro un tempo finito, $E[K_\infty] < \infty$, si ottiene una relazione di throughput generica, valida quindi per diverse varianti di TCP nei diversi ambienti di AQM.

$$Throughput = \frac{E[K_\infty] \cdot E[A_1] + E[\sum_{k=1}^{K_\infty} B_k]}{E[\sum_{k=1}^{K_\infty} Y_k] + E[\sum_{k=1}^{K_\infty} Z_k^{FR}] + E[Z_\infty^{TO}]} \tag{6.1}$$

Nel caso del TCP Tahoe, le sequenze (B_n), (Y_n) e (Z_n) sono stazionarie, quindi i valori attesi delle sommatorie possono essere scritti come $E[\sum_{k=1}^{K_\infty} B_k] = E[K_\infty] \cdot E[B_\infty]$ (lo stesso vale per Y e Z).

Diversamente, nel caso del TCP NewReno esse dipendono dalla dimensione della finestra all'inizio del ciclo (resettata o dimezzata), quindi, per semplicità, si assume che tali equazioni, se non esatte, possano essere derivate utilizzando delle approssimazioni.

Considerando inoltre la probabilità di TO (i.e. la probabilità che un ciclo termini con un TO), $Q = \frac{1}{E[K_\infty]}$, l'equazione (6.1) può essere riscritta come:

$$Throughput_{TCP} = \frac{E[A_1] + E[B_\infty]}{E[Y_\infty] + (1-Q) \cdot E[Z_\infty^{FR}] + Q \cdot E[Z_\infty^{TO}]} \tag{6.2}$$

6.1.3 Derivazione quantità throughput

Il procedimento dettagliato per derivare le singole variabili aleatorie dell'equazione (6.2) è complesso e oneroso sia in termini di tempo che di spazio e verrà descritto concisamente di seguito, demandando le parti esplicative e puramente di calcolo alla sottosezione A.1.2 dell'Appendice A.

Derivazione $E[A_\infty]$

Si parte dal tasso di perdita p, definito tramite la legge dei grandi numeri come

$$p = \lim_{n\to\infty} \frac{n}{\sum_{k=1}^{n}(A_k + 1)} = \frac{1}{E[A_\infty] + 1} \tag{6.3}$$

Infatti durante n cicli di trasmissione persistente sono persi n pacchetti (come primi nel ciclo) su un totale di $\sum_{k=1}^{n}(A_k + 1)$; da ciò si ricava quindi la relazione

$$E[A_\infty] = \frac{1-p}{p}$$

Si noti che A_n è indipendente dalla politica di accodamento e dalla versione di TCP.

Derivazione $E[Z_\infty^{TO}]$

Il discorso è analogo a quello trattato nell'articolo di *Padhye et al.*[11], il cui risultato è descritto dall'equazione (4.14); in questo caso l'approssimazione finale è leggermente diversa e soddisfatta per valori di $p_0 \leq 0.3$.

$$E[Z_\infty^{TO}] \approx \frac{T_0}{(1-2p_0)} \tag{6.4}$$

T_0 è una costante ed indica la lunghezza dell'intervallo del primo TO mentre $p_0 = P(A_n = 0)$ denota la probabilità che il primo pacchetto dopo un timeout venga nuovamente perso.

Si noti che anche Z_n^{TO} risulta indipendente dalla politica di accodamento e dal TCP adottato.

Derivazione W_n

Si introduce un parametro $a \in [0, 1)$ che distingue il caso del TCP Tahoe ($a = 0$) dal NewReno ($a = \frac{1}{2}$), per differenziare il comportamento tra FR e TO. Ogni b round la finestra è incrementata di un'unità fino all'istante della prima perdita, quindi ad ogni round l'incremento è di $\frac{1}{b}$.

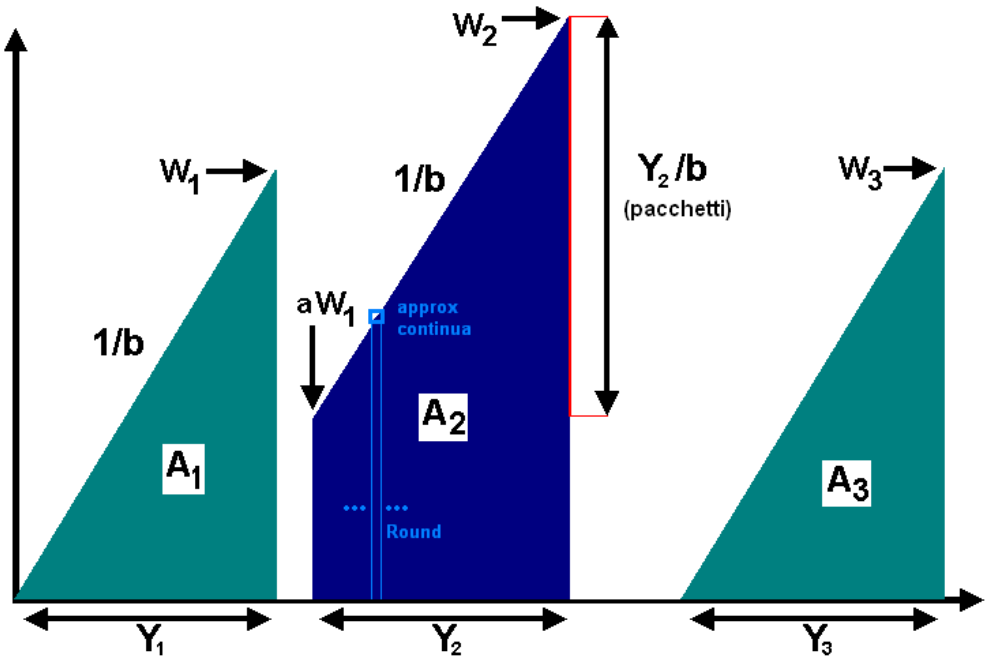

Figura 6.1: dinamiche della finestra con approssimazione continua

La relazione per la finestra può essere compresa esaminando la Figura 6.1. A_n corrisponde all'area sotto l'evoluzione della finestra (trapezio/triangolo)

al ciclo n:

$$A_n = a \cdot W_{n-1} \cdot J_{n-1} \cdot Y_n + \left(\frac{Y_n^2}{2} \cdot \frac{1}{b} \right)$$

mentre W_n corrisponde alla dimensione della finestra alla fine del ciclo precedente, dimezzata o resettata se NewReno mentre soltanto resettata se Tahoe, ed incrementata di $\frac{Y_n}{b}$

$$W_n = a \cdot W_{n-1} \cdot J_{n-1} + \frac{Y_n}{b} \tag{6.5}$$

Dopo alcune elaborazioni matematiche riportate nella sottosezione A.1.2 dell'Appendice A, si ottiene la relazione generica:

$$W_n = \sqrt{\frac{2}{b} \cdot \left(A_n + \sum_{k=1}^{n-1} a^{2\cdot(n-k)} \cdot A_k \cdot (J_k \cdot \ldots \cdot J_{n-1}) \right)} \tag{6.6}$$

in cui $J_1 \ldots J_{n-1}$ valgono tutti 1 per indicare FR terminati con successo.

Indipendenza dei pacchetti fino alla prima perdita

A questo punto si introduce l'assunzione di indipendenza dei pacchetti fino alla prima perdita, da cui $\{A_n\}$ diventa una sequenza di variabili aleatorie *i.i.d.* geometricamente distribuite. Tuttavia, ricorrendo all'utilizzo di approssimazioni continue, la distribuzione analoga alla geometrica nel continuo è la distribuzione esponenziale, dove il tempo di attesa del verificarsi del primo evento corrisponde alle prove necessarie affinché si verifichi. Per il teorema di convergenza delle distribuzioni si approssima perciò la geometrica all'esponenziale, semplificando l'analisi numerica tramite l'identità

$$A_n = \frac{V_n}{p}, \quad n \geq 1 \tag{6.7}$$

in cui $\{V_n\}$ è una sequenza di variabili aleatorie *i.i.d.* con distribuzione esponenziale di media $\frac{1}{\lambda}$, con $\lambda = 1$, e p tende a zero assumendo valori minori di 0.4.

Derivazione B_n

Per derivare B_n è necessario definire ulteriori variabili aleatorie che permettano la distinzione tra RED e DropTail.

- C_n, il numero di pacchetti trasmessi con successo fino alla prima perdita nel rond n in cui essa avviene.
- $D_n = W_n - C_n - 1$, il numero di pacchetti trasmessi con successo dopo la prima perdita nel round n in cui essa avviene.

Nel caso RED dunque, gli invii dopo la prima perdita sono $D_n + C_n = W_n - 1$, cioè i pacchetti inviati dopo la prima perdita nel round n, più quelli inviati nel round $n+1$ in risposta ai pacchetti ACKed nel round n fino alla prima perdita; d'altra parte per DropTail i pacchetti sono soltanto C_n.

Dall'assunzione di indipendenza tra i pacchetti fino alla prima perdita e condizionando a W_n, si osserva che B_n ha distribuzione Binomiale:

$$B_n \sim Bin(X_n, 1-p), \qquad X_n = \begin{cases} C_n & \text{(DropTail)} \\ W_n - 1 & \text{(RED)} \end{cases} \tag{6.8}$$

Al fine di applicare approssimazione continua, si abusa leggermente delle precedenti considerazioni effettuando la normalizzazione $D_n + C_n = W_n$; per semplicità di modellazione si considera poi una sequenza di variabili aleatorie *i.i.d.* con distribuzione Uniforme nell'intervallo $(0,1)$ e si definiscono quindi[1]

$$\begin{aligned} C_n &= W_n \cdot U_n \\ D_n &= W_n \cdot (1 - U_n) \end{aligned} \tag{6.9}$$

dove U_n indica la probabilità che un pacchetto venga trasmesso con successo.

Derivazione $E[W_\infty]$ con assunzione d'indipendenza dei pacchetti

Partendo dall'equazione (6.6), si applica l'approssimazione continua con l'ipotesi di indipendenza dei pacchetti inviati fino alla prima perdita, ottenen-

[1] Per le proprietà della distribuzione Uniforme, la distribuzione di una variabile aleatoria X sull'intervallo $(0,1)$ è uguale a quella di $1-X$ sull'intervallo $(0,1)$.

do, per $n \to \infty$, la versione stazionaria di W_n. Utilizzando $\{V_n\}_{n\geq 0}$ definita precedentemente e sostituendo A_k con $A_k = \frac{V_{n-k}}{p}$, per $k = 1, \ldots, n$, si ottiene

$$W_\infty = \sqrt{\frac{2}{b \cdot p} \cdot \sum_{k=0}^{K^0} a^{2k} \cdot V_k} \tag{6.10}$$

dove K^0 indica la distribuzione stazionaria per il numero di eventi di 3 DUPACK occorsi dopo il TO più recente.

Il caso del TCP Tahoe è descritto dettagliatamente in [15] e sintetizzato nell'Appendice A, sottosezione A.1.2.

Riguardo NewReno, la situazione è più complicata ed un'approssimazione ragionevole si consegue applicando il metodo dei momenti[2] sui primi due momenti e approssimando in base all'upper bound:

$$E\left(\sqrt{\sum_{k=0}^{K^0} a^{2k} \cdot V_k}\right) \leq \sqrt{E\left(\sum_{k=0}^{K^0} a^{2k} \cdot V_k\right)} \approx \sqrt{\frac{1 - E[a^{2\cdot(K^0+1]}}{1 - a^2}}$$

dove il lato destro dell'approssimazione è dato dallo sviluppo della serie geometrica.

Poiché $\frac{1}{K^0}$ cicli terminano con un TO si deduce che K^0 ha distribuzione

[2]Metodo dei momenti: metodo per la stima puntuale di parametri tramite ricerca di stimatori che devono soddisfare una condizione relativa a uno o più momenti campionari; in particolare nel metodo del secondo momento si utilizza la disuguaglianza di Chebyshev e la varianza. Nel caso specifico, la sommatoria è posta uguale a W^2.

geometrica di parametro q. Quanto appena detto permette di calcolare

$$\begin{aligned} E[a^{2K^0}] &= \sum_{k=1}^{n} a^{2k} \cdot P(K^0 = k) \\ &= q \cdot \sum_{k=1}^{n} a^{2k} \cdot (1-q)^{k-1} \\ &= \frac{q}{1-q} \cdot \sum_{k=1}^{\infty} (a^2 \cdot (1-q))^k \\ &= \frac{q}{1-q} \cdot \left(\frac{1}{1-a^2 \cdot (1-q)} - 1\right) \\ &= \frac{q}{1-q} \cdot \left(\frac{a^2 \cdot (1-q)}{1-a^2 \cdot (1-q)}\right) \\ &= \frac{q \cdot a^2}{1-a^2 \cdot (1-q)} \end{aligned}$$

da cui segue il risultato approssimato $\sqrt{E(\sum_{k=0}^{K^0} a^{2k} \cdot V_k)} \approx \frac{1}{\sqrt{1-a^2 \cdot (1-q)}}$. Ponendo infine $q = Q$ e sostituendo $a = \frac{1}{2}$ si giunge a

$$E[W_\infty] \leq \sqrt{E[W_\infty^2]} \approx \sqrt{\frac{2}{b \cdot p}} \cdot \frac{2}{\sqrt{3+Q}} \tag{6.11}$$

Derivazione $E[Y_\infty]$

Si ricava semplicemente dall'equazione (6.5) considerando lo stato stazionario di Y_n, sostituendo la (6.11) e approssimando:

$$\begin{aligned} E[Y_\infty] &= b \cdot E[W_\infty - a \cdot J_\infty \cdot W_\infty] \\ &\approx b \cdot (E[W_\infty] - a \cdot (1-Q) \cdot E[W_\infty]) \\ &\approx b \cdot \left(1 - \sqrt{\frac{2}{b \cdot p}} \cdot \frac{1-Q}{\sqrt{3+Q}}\right) \end{aligned} \tag{6.12}$$

Si noti che come $E[A_\infty]$ e $E[Z_\infty^{TO}]$ anche $E[Y_\infty]$ è indipendente dalla politica di AQM.

Derivazione Q

Innanzitutto si evidenzia come il parametro b relativo agli ACK cumulativi non sia utilizzato per il calcolo di questa probabilità, e in generale durante

le fasi di ritrasmissione dei pacchetti persi, dato che nel corso del recovery ogni singolo pacchetto fuori ordine deve essere confermato singolarmente.

Relativamente alla versione Tahoe, il calcolo di Q è descritto sinteticamente nell'Appendice A, sottosezione A.1.2.

Nel caso di NewReno, evitare un TO significa ritrasmettere con successo tutti i pacchetti persi dopo il primo (incluso), oltre che ricevere almeno k DUPACK come in Tahoe. Il totale dei pacchetti da ritrasmettere con successo nel round della prima perdita saranno quindi $1 + Bin(D_n, p)$ per RED (i.e. il primo pacchetto perso più tutti i pacchetti persi con probabilità p su D_n pacchetti inviati), mentre per DropTail $1 + D_n$, poiché dopo il primo perso tutti i successivi pacchetti dello stesso round sono scartati. Definita dunque F_n come il numero di pacchetti ritrasmessi senza successo, segue che

$$Q_n = 1 - P(B_n \geq k, F_n = 0) \tag{6.13}$$

Sfruttando la funzione generatrice $E[z^S] = ((1-p) + z \cdot p)^{D_n}$ per la binomiale, con $z = 1-p$ ed $S \sim Bin(D_n, p)$, e condizionando a D_n, si ha che la probabilità media di ritrasmettere con successo i pacchetti persi (in RED) risulta

$$\begin{aligned} E[(1-p)^{Bin(D_n,p)} \mid D_n] &= (1 - p + (1-p) \cdot p)^{D_n} \\ &= (1 - p^2)^{D_n} \end{aligned}$$

da cui

$$P(F_n = 0 \mid D_n) = \begin{cases} (1-p) \cdot (1-p^2)^{D_n} = (1-p^2)^{1+D_n} & \text{(RED)} \\ (1-p) \cdot (1-p)^{D_n} = (1-p)^{1+D_n} & \text{(DropTail)} \end{cases}$$

Denotando con $E[(Z; A)] = E[Z \cdot 1_A]$ il valore atteso di Z sull'insieme A, con $1_A : Y \to \{0, 1\}$ funzione indicatrice di A, per NewReno in ambiente RED si avrà

$$1 - Q_n = (1-p) \cdot E[(1-p^2)^{D_n}; B_n \geq k] \tag{6.14}$$

Allo stesso modo per DropTail:

$$1 - Q_n = (1-p) \cdot E[(1-p)^{D_n}; B_n \geq k] \tag{6.15}$$

Ricordando che nelle equazioni (A.1) ed (A.2) si è ottenuto che $1 - Q_n = P(B_n \geq k) = P(X_n \geq k, B_n \geq k)$ e che $P(B_n \geq k \mid X_n = x) \approx (1 - \binom{x}{k-1} \cdot p^{x-k+1})$, per ogni $r \in (0,1)$ si ha:

$$\begin{aligned} E[r^{D_n}; B_n \geq k] &= E[r^{D_n}; X_n \geq k, B_n \geq k] \\ &= E\left[\left(1 - \binom{X_n}{k-1} \cdot p^{X_n - k + 1}\right) \cdot r^{D_n}; X_n \geq k\right] \end{aligned} \tag{6.16}$$

Dalla (6.16), si distinguono infine i due casi RED e DropTail. Per il primo, sostituendo $D_n = (1 - U_n) \cdot W_n$, $r = 1 - p^2$ e sfruttando la funzione generatrice per la distribuzione uniforme su $(0,1)$, si ricava

$$\begin{aligned} E[r^{(1-U_n)\cdot W_n} \mid W_n = v] &= \int_0^1 r^{u \cdot v}\, du \\ &= \int_0^1 e^{u \cdot v \cdot \log r}\, du \\ &= \frac{1 - r^v}{-v \log r} \end{aligned} \tag{6.17}$$

che combinata con l'equazione (6.16), posto $k = 3$ e ricordando $X_n = W_n - 1$, risulta in

$$\begin{aligned} &E\left[\left(1 - \binom{W_n - 1}{2} \cdot p^{W_n - 3}\right) \cdot r^{(1-U_n)\cdot W_n}; W_n \geq 3\right] \\ &= \int_3^\infty \left(1 - \frac{1}{2} \cdot (v-1) \cdot (v-2) \cdot p^{v-3}\right) \cdot \frac{1 - r^v}{-v \log r} \cdot f_W(v)\, dv \end{aligned} \tag{6.18}$$

In ambiente DropTail, $X_n = C_n = U_n \cdot W_n$ e $D_n = (1 - U_n) \cdot W_n$, quindi posto $r = 1 - p$ e $k = 3$ si ha

$$\begin{aligned} &E\left[\left(1 - \binom{C_n}{2} \cdot p^{C_n - 2}\right) \cdot r^{(1-U_n)\cdot W_n}; C_n \geq 3\right] \\ &= \int_0^1 \int_{\frac{3}{u}}^\infty \left(1 - \binom{u \cdot v}{2} \cdot p^{u \cdot v - 2}\right) \cdot r^{(1-u)\cdot v} \cdot f_W(v)\, dv\, du \\ &\approx \int_0^1 \int_{\frac{3}{u}}^\infty r^{(1-u)\cdot v} \cdot f_W(v)\, dv\, du \\ &= \int_3^\infty \frac{1 - (1-p)^{v-3}}{-v \log(1-p)} \cdot f_W(v)\, dv \end{aligned} \tag{6.19}$$

Derivazione $E[Z_n^{FR}]$

Nel caso di Tahoe, la durata dell'intervallo di fast recovery è banalmente $E[Z_n^{FR}] = 1$, in quanto la ritrasmissione è relativa unicamente al primo pacchetto perso ed avviene quindi in un round sia per RED che DropTail.

D'altro canto per NewReno la situazione è più ingarbugliata: l'intervallo di recovery dipende dal primo pacchetto perso e da eventuali pacchetti persi successivamente al primo, $E[Z_n^{FR}] = 1 + N$; N è condizionata a D_n ed è geometricamente distribuita con troncamento a D_n in ambiente DropTail e troncamento a $Bin(D_n, p)$ usando RED. Poiché $N \subseteq D_n$, per un generico troncamento $1 + d$ si ha

$$P(N = k \mid D_n) = \frac{(1-p)^{k-1} \cdot p}{1 - (1-p)^{1+d}}, \quad 1 \leq k \leq 1 + d \tag{6.20}$$

Sviluppando in base al primo termine della serie di Taylor si ricava la seguente uguaglianza per il valore atteso di N

$$E[N] = \frac{1}{p} - \frac{(1+d) \cdot (1-p)^d}{1 - (1-p)^{1+d}}$$

che facendo tendere p a 0 è asintoticamente equivalente alla seguente quantità

$$E[N] \sim \frac{d}{2} - \frac{(d-4) \cdot d}{12} \cdot p, \quad p \to 0$$

Perciò, asintoticamente, al tendere di p a 0, si ha:

$$E[Z_n^{FR}] \approx \begin{cases} 1 + \frac{1}{2} \cdot E[D_n] + \frac{1}{3} \cdot E[D_n] \cdot p - \frac{1}{12} \cdot E[D_n^2] \cdot p & \text{(DropTail)} \\ 1 + \frac{5}{12} \cdot E[D_n] \cdot p - \frac{1}{4} \cdot E[D_n] \cdot p^2 + \frac{1}{12} \cdot E[D_n^2] \cdot p^3 & \text{(RED)} \end{cases}$$

Considerando le derivazioni per C_n e D_n nell'ottica della distribuzione ricavata per W_∞, si possono ordinare i termini dominanti per piccoli valori di p ricavando:

$$E[Z_n^{FR}] \approx \begin{cases} 1 + \frac{1}{4} \cdot E[W_n] & \text{(DropTail)} \\ 1 + \frac{5}{24} \cdot E[W_n] \cdot p & \text{(RED)} \end{cases}$$

Un compendio dei risultati finali per entrambe le versioni di TCP, nei due ambienti di AQM, è riportato nell'Appendice A, sottosezione A.1.3.

6.2 Analisi teorica del sistema anticipato

Uno degli aspetti iniziali analizzati minuziosamente riguarda la modalità di ritrasmissione dei pacchetti, cioè il procedimento, i tempi e gli effetti dell'aggiunta della ritrasmissione anticipata. Quando un pacchetto *pkt* arriva dal livello TCP al livello MAC esso viene accodato nel buffer che eventualmente smaltisce i pacchetti in coda prima di inviarlo. Arrivato il proprio turno, *pkt* viene trasmesso: in caso di trasmissione non riuscita, prima di decretarne la perdita definitiva il sistema effettua automaticamente altri 6 tentativi di ritrasmissione; se tutti i tentativi falliscono, si attiva il meccanismo di ritrasmissione anticipata e tramite il TED si notifica al livello TCP che *pkt* è andato perso, mentre a livello MAC si prosegue con l'invio del successivo pacchetto in coda.

Una prima questione importante è la stima media del numero di pacchetti che intercorre tra l'invio di un pacchetto ed il rinvio (tramite ritrasmissione anticipata) dello stesso pacchetto a livello TCP, poiché in caso di successo delle ritrasmissioni tale stima potrebbe influenzare il verificarsi o meno del recovery. In particolare tale quantità dipende da:

- tempo di propagazione attraverso lo stack di rete;
- lunghezza dei tentativi di ritrasmissione;
- tempi di elaborazione ed invio;
- numero di pacchetti presenti nella sliding window inviabili durante il round.

Sulla base di un insieme ristretto di esperimenti pratici, la misura di tale stima è risultata alquanto variabile dovendo considerare situazioni in cui le condizioni di funzionamento del sistema portano a cambiamenti drastici del suo comportamento e delle prestazioni; di conseguenza la decisione più sensata è stata quella di non assegnare a tale stima un valore puntuale ma considerare un lower bound. Si assume perciò, tra una trasmissione e la

rispettiva ritrasmissione l'invio di un minimo di 3 pacchetti per le seguenti ragioni:

- la probabilità che sia minore di 3 è quasi nulla, quindi trascurabile;
- permettere alla fase di FR di avere luogo (tramite i 3 DUPACK);
- mostrare che, nonostante questa assunzione, il sistema anticipato riesca, sotto determinate condizioni, ad evitare il recovery.

Una seconda questione fondamentale riguarda la gestione della notifica di ritrasmissione ricevuta a livello TCP. Indipendentemente dall'implementazione reale del sistema anticipato, concettualmente sono state delineate 3 modalità di gestione:

- ritrasmissione con aggiunta;
- ritrasmissione con subentro;
- ritrasmissione ibrida.

La prima (più vicina all'attuale versione funzionante del sistema reale) consiste nel ritrasmettere anticipatamente i pacchetti persi in aggiunta ai pacchetti effettivamente inviabili nel round in cui è avvenuta la notifica di perdita. In questo caso, alla ricezione della notifica TED, un pacchetto da ritrasmettere non è accodato nel buffer della sliding window ma è identificato e rinviato immediatamente al livello MAC. I vantaggi principali di una gestione siffatta sarebbero:

- una diminuzione significativa dei tempi di attesa, dovuta ad una probabilità maggiore di evitare recovery e timeout;
- un miglioramento delle prestazioni poiché la finestra verrebbe dimezzata o resettata un numero minore di volte (supponendo la rete non congestionata).

Si consideri infatti il caso tipico in cui un'ostacolo sul canale comporti la perdita dei pacchetti di un'intera finestra (dopodichè il canale torni ad essere libero): l'attesa della scadenza del timeout risulterebbe inevitabile; di conseguenza la possibilità di ritrasmettere i pacchetti persi in aggiunta a quelli già inviati nel round permetterebbe al sistema di uscire dallo stato di attesa senza comportare alcun azzeramento della finestra di congestione. Tra gli svantaggi più evidenti di questa gestione si avrebbe il problema di appesantire il carico della rete in caso di effettiva congestione: infatti se le perdite sono dovute a congestione, le ritrasmissioni risulterebbero inefficaci e dannose. Inoltre tale approccio influenzerebbe il RTT dilungandone sostanzialmente la stima a causa del maggior numero di pacchetti (in caso di reinvio per perdite) inviabili durante un round.

La modalità di ritrasmissione con subentro, invece, al ricevimento della notifica di ritrasmissione prevede il reinserimento del pacchetto in testa alla coda dei pacchetti da trasmettere, e più precisamente un pacchetto viene inserito dopo l'ultimo pacchetto inviato ma non ancora ACKed ed il primo pacchetto ancora da inviare, subentrando così all'ultimo pacchetto inviabile in quel round che passerà al round seguente (e così via per i successivi). Vantaggi e svantaggi principali di questo approccio sono essenzialmente duali a quelli precedenti. Il subentro infatti non appesantisce la rete con invii extra all'interno di un round e non tende ad incrementare le stime dei RTT; d'altro canto blocchi di perdite in condizioni di rete non congestionata porterebbero molto più facilmente il sistema in stati di stallo o in fasi di recovery. Si noti che questa modalità risulta più coerente con la politica di controllo della congestione del TCP, in quanto dispone i pacchetti in esubero nei round successivi rispettando il limite di trasmissione imposto dalla finestra di congestione.

Principalmente per i motivi elencati, la modalità di ritrasmissione con subentro verrà considerata come meccanismo di default durante l'analisi del sistema anticipato, fornendo dei casi di confronto con l'altro approccio ove reputato interessante.

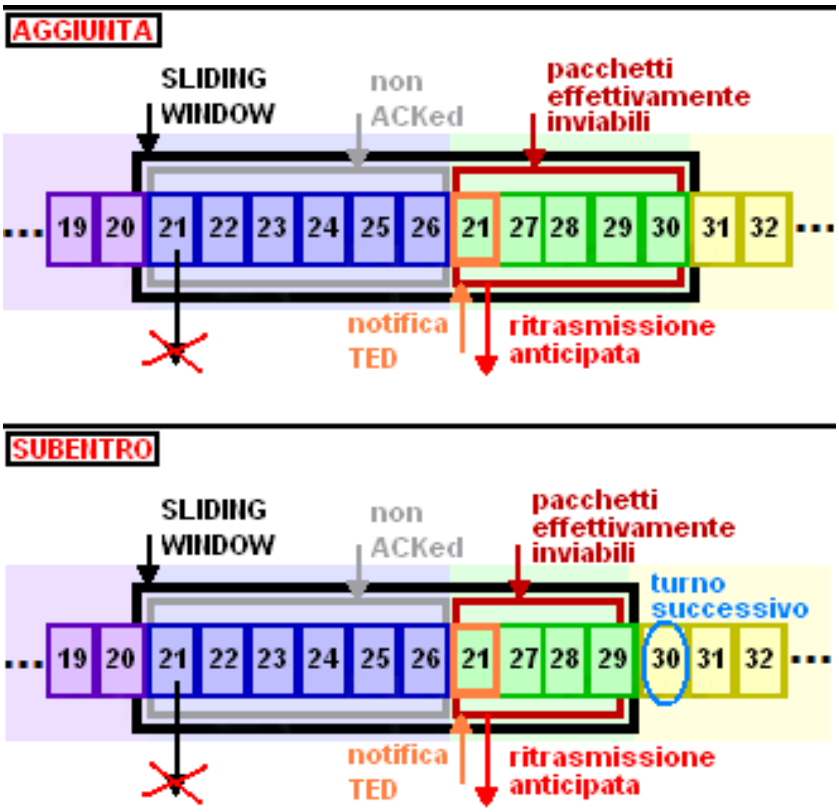

Figura 6.2: Confronto tra ritrasmissione con aggiunta e con subentro

I primi due metodi di ritrasmissione sono confrontati in Figura 6.2. Si noti che, come preannunciato, nella sliding window relativa alla ritrasmissione con aggiunta la finestra contiene un pacchetto in più rispetto alla modalità con subentro.

Un meccanismo di gestione ibrido potrebbe essere pensato per circoscrivere comportamenti sfavorevoli migliorando gli svantaggi che emergono dai due precedenti approcci. Ad esempio mantenendo una tipologia di ritrasmissione con subentro per la gestione di perdite isolate e passando alla modalità con aggiunta in caso di perdite multiple o in blocco, si potrebbero raggiungere compromessi per cui la finestra di congestione non venga appesantita dall'aggiunta di pacchetti addizionali da ritrasmettere, ma allo stesso tempo molte attese dovute ai tempi di recovery e timeout potrebbero essere evitate.

6.2.1 Analogie e differenze con il sistema classico

In generale, l'utilizzo di un sistema anticipato rispetto ad un sistema classico porterebbe una serie di conseguenze positive tra cui:

- probabilità di consegna individuale dei pacchetti maggiore;
- probabilità di entrare nella fase di FR minore;
- quantità media di tempo trascorso nella fase di recovery minore.

Questi ed altri comportamenti risultano evidenti analizzando esempi tipici di casi specifici di funzionento del sistema in diverse situazioni di utilizzo e condizioni[3].

CASO 1 si consideri la situazione mostrata nella Figura 6.3.

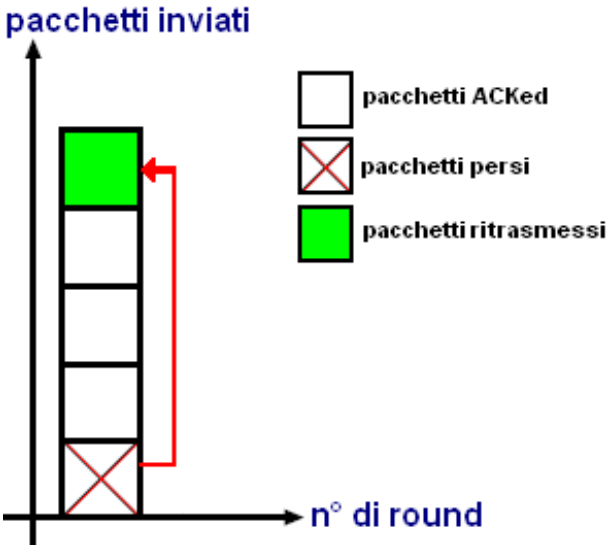

Figura 6.3: caso di perdita e ritrasmissione di un pacchetto

La ritrasmissione del primo pacchetto perso offre una nuova possibilità di consegna dello stesso pacchetto: risulta chiaro quindi come nonostante il numero di pacchetti consegnati nel round sia comunque 4,

[3]NOTA: nelle figure dei vari casi si suppone che i pacchetti siano numerati in ordine di invio (dal basso verso l'alto e da sinistra a destra) da 1 a n.

esiste una probabilità maggiore, dovuta alla ritrasmissione e data da $1 - p^2$, che il pacchetto perso venga consegnato[4]. Si noti invece che la probabilità globale (intera finestra) di consegna dei pacchetti non cambia poiché il pacchetto ritrasmesso sostituisce un altro pacchetto che sarebbe ugualmente inviato con successo. Inoltre, si noti che la finestra di congestione ha dimensione 5 ma la ritrasmissione del pacchetto perso fa scivolare il quinto pacchetto (che sarebbe dovuto essere stato inviato nel round) al round successivo; di conseguenza il numero di pacchetti diversi inviati nel round scende da 5 a 4, mentre il numero di pacchetti consegnati resta invariato a 4.

CASO 2 si consideri la situazione mostrata in Figura 6.4.

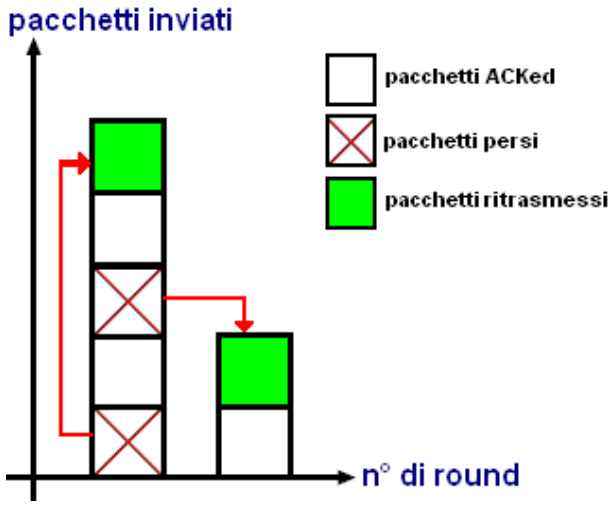

Figura 6.4: caso di ritrasmissioni con esclusione FR

La serie di perdite e ritrasmissioni combinate come in figura mostrano un aspetto interessante del sistema anticipato: la possibilità di evitare

[4]ASSENZA DI MEMORIA: nel valutare statisticamente l'impatto delle ritrasmissioni un aspetto da tenere in cosiderazione è la proprietà di assenza di memoria, anche detta Fallacia dello Scommettitore, che riguarda la totale ininfluenza degli eventi passati su eventi futuri; ciò implica ad esempio che un evento casuale non ha maggiore/minore probabilità di verificarsi perché non si è verificato per un lungo periodo di tempo.

la fase di FR nonostante le perdite e l'invio di un numero sufficiente di pacchetti (≥ 3) per la ricezione dei 3 DUPACK. Dopo la prima perdita, infatti, sono inviati 3 pacchetti prima della ritrasmissione effettuata con successo del primo pacchetto perso; tuttavia la perdita del terzo pacchetto fa in modo che la ritrasmissione avvenga prima della ricezione del terzo DUPACK, escludendo il recovery. La consegna del primo pacchetto tramite ritrasmissione azzera il conteggio dei DUPACK, che quindi riparte da 1, permettendo alla seconda ritrasmissione di avvenire nuovamente dopo l'invio di 3 pacchetti ma prima dei 3 DUPACK. Il risultato di tutto ciò è una ripresa nettamente più rapida della comunicazione che non risente della fase di FR, evitando un calo di prestazioni dovuto al dimezzamento della dimensione della finestra di congestione. Si noti che quella delineata in figura è soltato una delle svariate possibilità di combinazione di perdite e ritrasmissione che porta il sistema a comportarsi nella maniera descritta. Si noti inoltre che, come anticipato e contrariamente all'utilizzo della ritrasmissione con aggiunta, il sistema anticipato che utilizzi ritrasmissione con subentro non contempla situazioni in cui le ritrasmissioni riescano ad evitare il timeout causato da blocchi di perdite corrispondenti all'intera dimensione della finestra di congestione.

CASO 3 si consideri la situazione mostrata in Figura 6.5.

In questo caso, la ritrasmissione avviene oltre l'invio di 3 pacchetti che provocano i 3 DUPACK e l'avvio del FR. Ciononostante il pacchetto perso giunge a destinazione prima del round successivo necessario alla ritrasmissione dovuta al recovery; ciò implica nella migliore delle ipotesi (i.e. RTT sufficientemente lungo per la ricezione dell'ACK) una durata pressoché nulla del round in recovery mode mentre nella peggiore un accorciamento significativo della stessa, poiché l'end-pint destinatario riceve anticipatamente il pacchetto originariamente perso. In una situazione in cui perdite multiple causano una fase di recovery suddivisa in diversi round, le ritrasmissioni anticipate potrebbero favo-

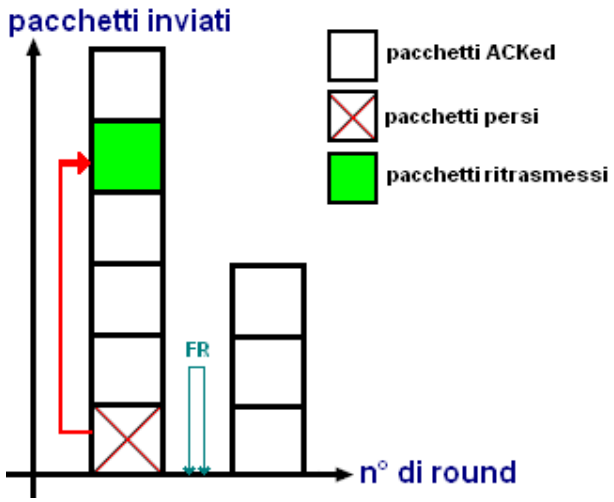

Figura 6.5: caso di ritrasmissioni con FR

rire significativamente l'interattività del sistema tramite la riduzione di tali intervalli di tempo. Tuttavia, differentemente dal caso precedente e nonostante il guadagno in termini di tempo, il dimezzamento della finestra è invece inevitabile.

CASO 4 si consideri la situazione mostrata in Figura 6.6.

Considerando la perdita dei pacchetti ritrasmessi anticipatamente in un contesto di perdite in blocco, ci si accorge che sebbene il comportamento sia diverso in confronto al sistema classico (i.e. le ritrasmissioni coprono i pacchetti persi sostituendo quelli inviabili), il risultato è il medesimo poiché le ritrasmissioni non sovraccaricano il sistema con invii extra e lo scadere del timeout lo riporta allo stato iniziale. Si noti che le ritrasmissioni anticipate, anche in caso di fallimento, non influenzano negativamente il throughput poiché in accordo con la definizione data precedentemente si considerano unicamente i pacchetti inviati con successo nell'unità di tempo.

Dai casi descritti emergono alcune delle potenzialità del sistema anticipato. Come già evidenziato, un vantaggio molto importante da considerare

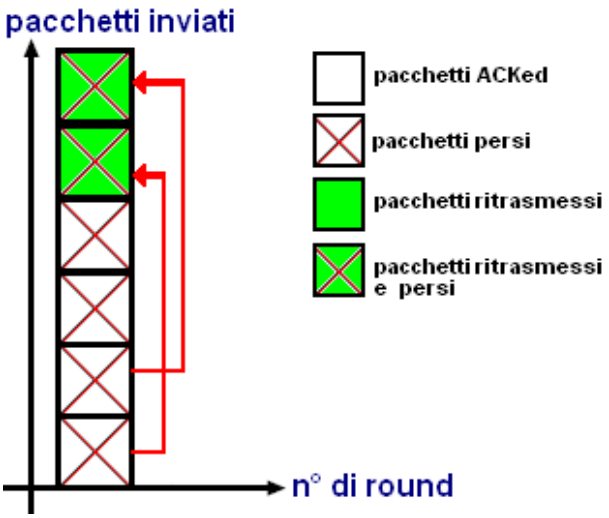

Figura 6.6: caso di ritrasmissioni perse in blocco

è l'interattività, fondamentale in alcune applicazioni: le ritrasmissioni infatti incrementano l'interattività del sistema facendo in modo che i pacchetti persi arrivino prima a destinazione, a discapito dei successivi. Un simile comportamento migliorerebbe il tempo di risposta del sistema e la qualità della comunicazione, fornendo supporto, ad esempio, ai processi di riassemblaggio di dati frammentati velocizzando la ricostruzione. Vi sono tuttavia dei casi in cui le ritrasmissioni anticipate potrebbero peggiorare la situazione. Un caso lampante è quello in cui i tempi tra le ritrasmissioni anticipate e l'inizio della fase di recovery fossero troppo brevi; può accadere che il destinatario non abbia abbastanza tempo per inviare l'ack di ricezione del pacchetto ritrasmesso e di conseguenza il mittente ritrasmetta i pacchetti persi durante il recovery, intasando ulteriormente il canale specialmente in presenza di congestione. Inoltre evitare un FR in determinati casi potrebbe influenzare negativamente il sistema: in una situazione in cui più sorgenti inviino più flussi di dati sovraccaricando la rete, il meccanismo di riduzione della finestra ridurebbe il livello di congestione; si noti tuttavia che si tratta di un caso molto raro e sfortunato poiché a causa, appunto, della congestione difficilmente le ritrasmissioni avrebbero successo.

6.2.2 Discussione sul modello

Quanto detto può essere una solida base su cui lavorare per realizzare un modello adeguato al sistema anticipato partendo dal modello classico. Risulta subito evidente come una modifica relativamente semplice ed in un certo senso naturale nel funzionamento di sistemi wireless basati su TCP rivoluzioni profondamente i comportamenti del sistema. Di conseguenza, per realizzare un modello che utilizzi il metodo basato sulla teoria del rinnovamento e descriva tutte le dinamiche del TCP viste finora, è necessario rivisitare e adattare al sistema anticipato il modello classico, individuando e studiando i problemi che li accomunano e li differenziano.

Uno dei punti chiave da riconsiderare per adattare il modello classico al sistema anticipato è la gestione degli eventi di perdita. Dai casi discussi si è infatti visto che la regolarità delle comunicazioni del sistema può essere comunque mantenuta grazie alle ritrasmissioni, nonostante gli eventi di perdita dei pacchetti; dopo diversi round in cui durante la comunicazione non si verificano perdite, la prima perdita non necessariamente porta il sistema in modalità di recovery. Tale comportamento invalida una delle ipotesi fondamentali su cui il modello poggia: l'evento della prima perdita di un pacchetto come punto di terminazione del ciclo e ripristino della comunicazione tramite FR o TO. Nel sistema anticipato quindi non c'è più una separazione netta tra il comportamento del sistema precedente e successivo alla prima perdita, che invece dipende dai vari casi derivanti dalla particolare combinazione di perdite e ritrasmissioni e può portare il sistema in uno stato diverso dal recovery. In particolare, gli stati in cui il sistema può trovarsi dopo la prima perdita sono essenzialmente 2 e vengono descritti di seguito.

- *stato di FR o TO immediato*: come nel sistema classico, la prima perdita porta il sistema nella fase di recovery o ad attendere lo scadere di un timeout; si ricade perciò nella stessa modalità di comportamento del sistema classico. Tuttavia la probabilità che questa situazione si verifichi diminuisce rispetto a quella presente nel sistema classico per via della possibilità di evitare il recovery grazie alle ritrasmissioni

anticipate. Ovviamente il modello classico tratta unicamente questo caso.

- *stato di perdite e ritrasmissioni ripetute*: le cose cambiano quando, dopo la prima perdita, le ritrasmissioni anticipate e successive perdite si mescolano in maniera tale che i pacchetti precedentemente persi vengano consegnati evitando nell'immediato di entrare in FR o TO. In tale evenienza, il sistema potrebbe attraversare un lasso di tempo in cui perdite e ritrasmissioni si combinino sfociando infine in un FR o TO, oppure evitando il FR e tornando a trasmettere senza perdite. Tale periodo di tempo è tuttavia una quantità arbitraria difficilmente quantificabile, che influenza direttamente il calcolo del throughput così come ricavato nel modello classico, rendendo necessaria l'introduzione di profonde modifiche al fine di modellare questa situazione temporanea e decretare la fine o meno di un ciclo. Intuitivamente, si può affermare che questo dovrebbe essere il caso più comune, poiché fonde insieme il comportamento del sistema classico e di quello anticipato sfruttando al massimo il canale di comunicazione prima di ridurre la capacità di trasmissione.

La soluzione più ovvia al problema descritto potrebbe essere quella di individuare la perdita del pacchetto che eventualmente porti al FR o al TO. Le difficoltà principali derivano, appunto, dal fatto che tale perdita non coinciderebbe obbligatoriamente con la prima del ciclo ma corrisponderebbe all'ultima perdita legata direttamente alla produzione dei 3 DUPACK, oppure alla prima perdita di un blocco che interromperebbe la comunicazione fino allo scadere del timeout. In aggiunta, le perdite potrebbero non avere influenza sui meccanismi di recovery, poiché non necessariamente sarebbero motivo di FR o TO dal momento che le ritrasmissioni riuscirebbero ad evitarli.

Una seconda possibilità che potrebbe aggirare il problema sarebbe considerare l'ipotesi in cui, anche con le ritrasmissioni, la prima perdita implichi sempre e comunque l'inizio di una situazione di FR o TO. Tale assunzione di semplificazione potrebbe sembrare molto stringente ma, considerandone i

tradeoff, potrebbe risultare comunque ragionevole: da un lato, la valutazione del sistema anticipato diverrebbe incompleta, dato che si considererebbero soltanto i casi in cui le ritrasmissioni anticipate migliorerebbero l'interattività tralasciando i benefici derivanti dalla capacità di evitare il recovery; il rovescio della medaglia sarebbe lo sviluppo di un modello molto più vicino a quello classico, che necessiterebbe di modifiche certamente più semplici legate soltanto al calcolo di alcune quantità per la stima del throughput.

L'ultima soluzione pensata, probabilmente la più estrema e poco pratica, riguarda la modifica del sistema anticipato al fine di adattarlo all'utilizzo con la logica di recovery della specifica versione di TCP; per esempio, supponendo TCP NewReno, si potrebbe introdurre un meccanismo che posticipi le ritrasmissioni anticipate dopo la ricezione dei 3 DUPACK, facendo in modo che la prima perdita inneschi inevitabilmente i meccanismi di recovery.

Si noti che, anche considerando l'utilizzo della modalità di ritrasmissione con aggiunta, la situazione non cambia e l'evento di prima perdita nel ciclo allo stesso modo non funge più da delimitatore certo per l'avvio del recovery; restano tuttavia i problemi delineati precedentemente a inizio sezione che comportano comunque dei metodi di analisi differenti e conseguentemente l'introduzione di modifiche sostanziali al sistema classico.

Potenziali interventi al modello

Sotto questa nuova prospettiva, si notano diversi problemi tecnici che è necessario risolvere per adattare il modello classico al sistema anticipato. Nello specifico si rivaluteranno gli aspetti matematici predominanti determinando la validità o meno delle parti considerate e, ove possibile, si tenterà di fornire dei suggerimenti o formulare delle proposte per lo sviluppo futuro di un modello idoneo al sistema anticipato.

In primo luogo, si riesaminano le singole variabili aleatorie utilizzate come quantità nel calcolo del throughput generico.

- A_n: poiché il primo pacchetto perso non indica più con certezza l'immediato ingresso del sistema in FR o TO, quindi la fine di un ciclo, questa

variabile aleatoria avrebbe senso introducendo opportune modifiche:

- una prima alternativa potrebbe essere quella di individuare la perdita che effettivamente produca un FR o un TO e ricalcolare A_n in base a tale evento;
- una seconda alternativa (presa in considerazione durante l'analisi che segue) comporterebbe l'introduzione di un'ulteriore variabile, M_n, interposta tra A_n e B_n che indichi il numero di pacchetti trasmessi con successo dalla prima perdita fino alla perdita che effettivamente scateni un FR o un TO;

in entrambi i casi potrebbe essere necessario un parametro che indichi, a priori, la probabilità con cui la combinazione di perdite e ritrasmissioni conduca ad uno degli stati descritti nella precedente sottosezione; inoltre, la scelta di considerare A_n separata da M_n potrebbe favorire il riutilizzo di alcune assunzioni e sviluppi già adottati nel modello classico.

- B_n: supposta l'introduzione di M_n, ed escludendo la decisione di inglobarla in B_n, la definizione di B_n rimarrebbe la stessa con l'ovvia accortezza di considerare come evento iniziale non obbligatoriamente la prima perdita del ciclo, ma l'effettiva perdita che causa FR o TO.

- J_n: questo parametro dovrebbe essere mantenuto e potrebbe essere esteso con eventuali valori utili (e.g. 2 per indicare, esclusivamente, che la fase di FR è stata evitata da una particolare combinazione di perdite e ritrasmissioni anticipate e quindi la finestra di congestione resta invariata).

- Y_n: la definizione di Y_n sarebbe riutilizzabile a patto che il limite superiore del calcolo si sposti dal round della prima perdita del ciclo al round della perdita che effettivamente provochi il recovery.

- Z_n^{FR}: similmente a Y_n, la definizione sarebbe riutilizzabile considerando come limite inferiore del calcolo il round della perdita che effettivamente provochi il FR.

- Z_n^{TO}: il discorso è analogo a Z_n^{FR}.

- Z_n: la definizione potrebbe essere ancora utilizzata e sarebbe conseguenza delle precedenti, essendo Z_n la somma di Y_n, Z_n^{FR} e Z_n^{TO}.

In accordo con l'analisi proposta, ciò che emerge finora, è il fatto che in linea di massima gran parte delle definizioni del modello classico potrebbero essere riutilizzate con qualche modifica ed aggiunta; tuttavia, come ulteriormente chiarificato in seguito, una valutazione più attenta evidenzia che alcune modifiche e processi di derivazione delle quantità introducono dei cambiamenti significativi nella logica di funzionamento ed analisi del sistema, dovuti essenzialmente all'incertezza di entrare nello stato di FR o TO che stravolge alcuni dei requisiti ed assunzioni alla base del modello classico.

In secondo luogo, dopo aver riesaminato le variabili, riconsiderando più in generale l'impostazione del modello ed entrando nello specifico delle principali derivazioni delle singole quantità utilizzate per il calcolo del throughput, si possono comprendere più approfonditamente i problemi che dovrebbero essere risolti. Per determinare la fine di un ciclo, non è più sufficiente basarsi esclusivamente sulla prima perdita. Introducendo per esempio M_n, come descritto precedentemente, l'assunzione di indipendenza dei pacchetti, valida unicamente per A_n definita secondo il modello classico, diverrebbe parzialmente valida per via dei diversi stati che potrebbe attraversare il sistema prima della fine del ciclo; di conseguenza la distribuzione geometrica descriverebbe in maniera incompleta l'evolvere della comunicazione, modellando solo il caso in cui ogni ciclo termini necessariamente con l'evento della prima perdita. La stessa definizione della probabilità di perdita p tramite la legge dei grandi numeri (equazione (6.3)) dovrebbe essere rivista al fine di tener conto da un lato degli eventuali ulteriori pacchetti persi successivamente al

primo all'interno del ciclo e dall'altro l'invio di eventuali pacchetti aggiuntivi (dati da M_n) oltre a quelli già contati da A_n.

In generale, un'ipotesi di approccio per riorganizzare il modello classico potrebbe essere quella di utilizzare le Catene di Markov, modellando il sistema come un insieme di stati (e.g. stato di trasmissioni senza perdite, stato di trasmissioni con perdite combinate a ritrasmissioni anticipate, stato di recovery, etc) che esso attraversa secondo determinate probabilità (stabilite da matrici di transizione) finché non giunge nello stato finale di terminazione del ciclo dovuto a FR o TO per poi riniziare (rinnovamento).

Scendendo più nello specifico e valutando brevemente le singole quantità necessarie per il calcolo del throughput, si nota che l'intervallo di timeout, $E[Z_\infty^{TO}]$, non avrebbe bisogno di essere ricalcolato, considerando già una nuova ridefinizione di p, poiché tale quantità dipenderebbe ancora dal tempo trascorso dal sistema in attesa della scadenza del TO e dalla probabilità che nei cicli successivi al primo TO tutti i pacchetti vengano nuovamente persi conducendo a ulteriori TO (i.e. calcolo backoff multiplo). D'altro canto, come si può intuire, definizione e derivazione di W_n andrebbero riviste poiché l'aggiunta di M_n influirebbe direttamente sul numero di pacchetti inviati in ogni ciclo, quindi la dimensione della finestra alla fine di un ciclo non dipenderebbe più soltanto da A_n e dall'incremento accumulato in ogni round, ma anche da M_n, che potrebbe introdurre pacchetti aggiuntivi in previsione del fatto che la dimensione della finestra di congestione potrebbe non essere necessariamente ridotta dopo l'evento di prima perdita del ciclo. Inoltre, si dovrebbe considerare che in caso di perdite multiple recuperate dalle ritrasmissioni anticipate l'incremento lineare di $1/b$ della finestra potrebbe subire un rallentamento, che eviterebbe il regolare incremento di pacchetti della finestra durante i b round e di conseguenza la dimensione della finestra al termine del ciclo. Tutte le quantità direttamente legate a $E[W_\infty]$ sono perciò condizionate da questo nuovo aspetto: ad esempio, le distribuzioni binomiale e uniforme relative al calcolo del numero di pacchetti trasmessi nel round della prima perdita diverrebbero quasi certamente non più esaurientemente

descrittive se si considerano casi con perdite multiple combinate a ritrasmissioni anticipate. In merito alle restanti quantità si può dire che la lunghezza dell'intervallo di FR, $E[Z_\infty^{FR}]$, potrebbe essere ridotta di un round considerando una ritrasmissione anticipata che consegna un pacchetto nel round precedente, ma sarebbe possibile derivarla nello stesso modo utilizzato nel modello classico se si suppone che le ritrasmissioni anticipate non abbiano effetto per i pacchetti ritrasmessi durante il FR. Infine, non risulta del tutto chiaro se anche per Q il procedimento di derivazione potrebbe essere analogo: introducendo infatti M_n la valutazione delle quantità (X_n, D_n, C_n) relative al round in cui si verifica l'effettiva perdita dovrebbe cambiare; tuttavia senza conoscere anticiptamente l'approccio usato e senza considerare gli sviluppi pratici per tale calcolo sembrerebbe che sostanzialmente le considerazioni sulla distribuzione utilizzata (Binomiale) sul procedimento di calcolo e sulla distinzione dei due casi RED e DropTail potrebbero essere riutilizzate.

Pertanto, si può concludere dicendo che al fine di prendere tali decisioni e concretizzare un modello per il sistema anticipato si dovrebbe tener conto in primis di come il modello venga ristrutturato per descrivere i vari stati del sistema, ed in secondo luogo delle nuove variabili eventualmente introdotte per supportare la nuova strategia di modellazione.

Capitolo 7

SVILUPPI FUTURI

Un chiaro punto di partenza per un prossimo lavoro che si avvalga ed estenda quanto già fatto è la realizzazione tecnica di un modello per il sistema anticipato, costruito sulla base di quello classico sfruttando le casistiche esaminate, l'analisi teorica affrontata e le proposte considerate. Risulta evidente che nonostante ciò, realizzare un modello che descriva correttamente e completamente il sistema anticipato, o una sua variante, non è cosa banale e richiederebbe tempistiche elevate, un ulteriore lavoro di analisi minuziosa ed una padronanza di strumenti matematici e statistici che probabilmente esula da quella posseduta da uno studente di informatica magistrale.

In secondo luogo, un'aspetto importante da considerare sarebbe l'esame di uno scenario che introduca diverse sorgenti eterogenee a flusso multiplo, interessante ed utile caratteristica da analizzare poiché situazione molto comune nelle attuali reti esistenti (e.g. Internet), che tuttavia comporterebbe presumibilmente una profonda ristrutturazione delle assunzioni concettuali e matematiche dei sistemi e relativi modelli.

Un'altra importante funzionalità da considerare sarebbe una seconda modifica del sistema con ritrasmissione anticipata che includa la ritrasmissione a livello access point nella parte inversa della comunicazione (i.e. da AP a nodo mobile); nel caso la trasmissione di un pacchetto non vada a buon fine, quindi anche in questo caso in seguito alle segnalazioni di un TED,

l'AP ritrasmetterebbe i pacchetti al MN. Tale aggiunta nel sistema è discussa ed implementata in [3] e [5] e prevede l'utilizzo di Netfilter che intercetta i pacchetti in uscita dal sistema operativo dell'AP e li accoda, leggendoli in un secondo momento da un'applicazione dedicata che li bufferizzerà in opportune strutture dati prima di trasmetterli all'host destinatario. Tale sistema dovrebbe favorire i meccanismi di controllo di congestione del TCP introducendo un ulteriore potenziamento del sistema anticipato in termini di affidabilità ed interattività, sebbene condurrebbe inevitabilmente ad un notevole stravolgimento dei problemi affrontati in questo lavoro.

Un aspetto del tutto nuovo riguarda invece un suggerimento di modifica del sistema anticipato (a livello TCP) che potrebbe rafforzare un suo punto debole probabilmente sottovalutato: aggiungere un meccanismo a livello TCP che permetta alle ritrasmissioni anticipate che hanno successo di alzare la soglia di ricezione dei DUPACK (e.g. da 3 a 4) o meglio di evitare il recovery nonstante la ricezione dei 3 DUPACK; ciò potrebbe concretizzarsi fodamentalmente grazie all'interattività, cioé alla ricezione dei pacchetti persi (tramite ritrasmissione anticipata) entro un'intervallo di tempo ragionevole prima delle ritrasmissioni ordinarie causate dal FR.

Nello specifico di sviluppi e cambiamenti strettamente legati al modello, si potrebbero ad esempio riconsiderare i parametri utilizzati come costanti nel modello (e.g. p e RTT) e tentare di integrarli e renderli parte attiva dell'analisi, offrendo un metodo di analisi più dinamico e generale che descriva scenari più complessi ed eterogenei. Inoltre, gli stessi, ed altri, obiettivi potrebbero essere raggiunti ristrutturando il modello sulla base di un diverso metodo di analisi che non si avvalga della Renewal Theory (e.g. Fixed Point Method), introducendo quindi una complessità ed un'espressività certamente maggiori e forse permetterebbero di unificare più modelli in un unico modello generico per entrambi i sistemi descritti.

Infine, parallelamente alla tipologia di lavoro intrapreso, si potrebbe condurre uno studio in cui si affrontano casi specifici di problemi legati ai nodi

mobili, per esempio periodi di disconnessione, spostamenti tipici, situazioni conflittuali, interferenze ed ostacoli, sovraccarichi improvvisi, etc, al fine di comprendere approfonditamente ed in specifico la reazione caso per caso del TCP, cogliendo gli aspetti che in un modello generico potrebbero risultare come anomalie o eccezioni di analisi.

CONCLUSIONI

Durante il corso di questo lavoro di tesi si è mostrata un'analisi matematica e teorica del funzionamento del protocollo TCP in contesti di mobilità. In particolare, lo studio si è incentrato su due versioni specifiche di TCP, il Tahoe e, soprattutto, il NewReno, poiché considerate le più conosciute e innovative sotto diversi aspetti.

Tali versioni di TCP sono state contestualizzate in uno scenario d'uso, che rappresenta un sistema informatico di comunicazione, in cui i due endpoint della comunicazione, un nodo mobile ed un host fisso, si scambiano dati attraverso un tratto di rete wireless (nodo mobile-access point) ed un tratto di rete wired (access point-routers-host fisso).

Per l'analisi delle prestazioni del sistema si sono valutati diversi metodi analitici per la modellazione del TCP, tra cui uno dei più noti chiamato Fixed Point Method, ma la scelta finale è ricaduta sul metodo basato sulla Teoria del Rinnovamento (in cui l'analisi del sistema si fonda su cicli legati alla riduzione della finestra di congestione in situazione di stazionarietà del sistema) poiché relativamente più semplice ed appropriato allo scopo prefissato.

Tra i vari modelli considerati, è stato scelto quello in [16] che più sembrava soddisfare i requisiti richiesti utilizzando la Teoria del Rinnovamento. Questo modello non solo tratta l'analisi per entrambe le versioni di TCP suddette, ma considera due diverse politiche di AQM, DropTail e RED, fondamentali per descrivere le dinamiche della finestra di congestione in situazioni in cui la mobilità su reti wireless conduce a diverse cause di perdita dei pacchetti.

In primo luogo tale modello è stato spiegato ed utilizzato per modella-

re il sistema classico con lo scopo di ricavare un metodo analitico corretto e completo che sia generico e misuri le prestazioni di un sistema basato su TCP tramite la stima del throughput, cioè la quantità di pacchetti consegnati nell'unità di tempo. Un simile approccio ha evidenziato diverse difficoltà derivanti dall'incapacità di riuscire a cogliere perfettamente tutti gli aspetti del sistema reale considerato, rendendo necessaria l'introduzione di un numero significativo di assunzioni di semplificazione e artifici matematici che permettessero di rendere il modello appropriato nei limiti di validità.

In seguito il risultato ottenuto da questa prima parte del lavoro è stato utilizzato come base per l'analisi teorica di un secondo sistema: il sistema anticipato. Lo scenario considerato è lo stesso ma il sistema è stato sottoposto ad alcune modifiche tecniche per permettere la ritrasmissione anticipata dei pacchetti persi, inviati tra il nodo mobile e l'access point, al fine di migliorare l'interattività della comunicazione.

La comprensione approfondita del funzionamento di un tale sistema modificato, la valutazione dei parametri e delle variabili utilizzate nel modello classico, la discussione di casi tipici di comportamento del TCP, l'analisi dei principali aspetti matematici, sono serviti per comprendere come in linea di massima un tale sistema funzionerebbe in varie situazioni e condizioni, se influirebbe positivamente o negativamente sulle prestazioni ed in che modo un modello matematico per questo nuovo sistema anticipato potrebbe essere sviluppato a partire da quello per un sistema classico.

In particolare si è notato che molto probabilmente un tale sistema aiuterebbe ad ottimizzare le prestazioni in diverse condizioni operative, soprattutto migliorando l'interattività con cui le comunicazioni hanno luogo senza interferire eccessivamente con l'utilizzo di banda del canale di comunicazione.

Infine, si nota inoltre che la realizzazione di un modello ad hoc per il sistema anticipato risulterebbe cosa non banale; infatti si può concludere affermando che il comportamento del TCP è in generale estremamente variabile dipendendo da numerosi fattori esterni, come la capacità del canale e dei buffer, le interferenze, i delay di trasmissione, etc, e fattori interni, come

l'impostazione dei parametri per il controllo di congestione e la lunghezza dei timer. Ciò implica che ogni minima modifica introdotta al meccanismo di funzionamento del TCP, come la ritrasmissione anticipata, comporta una consistente ristrutturazione delle ipotesi e dei requisiti chiave su cui un modello si fonda.

Appendice A

Approfondimenti paper

A.1 Paper [16] (*I. Kaj & J. Olsén*)

A.1.1 Calcoli componenti throughput generico

Definito il numero ordinato di cicli nell'intervallo in cui il j-esimo timeout occorre, $L_j = inf\{k \geq L_{j-1} + 1 : J_k = 1\}$, con $j \geq 1$ e $L_0 = 0$, K_j può essere espresso come $K_j = L_j - L_{j-1}$, con $j \geq 1$, e le varie parti dell'equazione di throughput generico sono calcolate come segue:

$$\sum_{k=L_{j-1}+1}^{L_j} Z_k = \sum_{k=L_{j-1}+1}^{L_j} (Y_k + Z_k^{FR}) + Z_{L_j}^{TO}$$

che indica il numero di round tra due timeout consecutivi;

$$\sum_{k=L_{j-1}+1}^{L_j} (A_k + B_k), \quad j \geq 1$$

che indica il numero di pacchetti inviati con successo tra due timeout consecutivi;

$$E[\sum_{k=L_{j-1}+1}^{L_j} A_k] = E[K_j] \cdot E[A_1]^1$$

che indica il numero medio di pacchetti inviati con successo precedentemente alla prima perdita, tra due timeout consecutivi.

[1] Per il lemma di Wald.

A.1.2 Calcoli derivazioni variabili throughput

W_n (calcoli)
Dall'equazione $A_n = a \cdot W_{n-1} \cdot J_{n-1} \cdot Y_n + \frac{Y_n^2}{2b}$, si ha:

$$Y_n^2 + (2 \cdot b) \cdot a \cdot W_{n-1} \cdot J_{n-1} \cdot Y_n - (2 \cdot b) \cdot A_n = 0$$

da cui considerando lo sviluppo del quadrato $(Y_n + b \cdot a \cdot W_{n-1} \cdot J_{n-1})^2$ si ricava:

$$\begin{aligned} Y_n^2 + 2 \cdot b \cdot a \cdot W_{n-1} \cdot J_{n-1} \cdot Y_n \pm (b \cdot a \cdot W_{n-1} \cdot J_{n-1})^2 - 2 \cdot b \cdot A_n &= 0 \\ (Y_n + b \cdot a \cdot W_{n-1} \cdot J_{n-1})^2 - b^2 \cdot a^2 \cdot W_{n-1}^2 \cdot J_{n-1}^2 - 2 \cdot b \cdot A_n &= 0 \\ Y_n + b \cdot a \cdot W_{n-1} \cdot J_{n-1} \pm \sqrt{b^2 \cdot a^2 \cdot W_{n-1}^2 \cdot J_{n-1}^2 + 2 \cdot b \cdot A_n} &= 0 \end{aligned}$$

ed infine, scartando il risultato negativo poiché non significativo, si ottiene $\frac{Y_n}{b}$:

$$\frac{Y_n}{b} = \sqrt{a^2 \cdot W_{n-1}^2 \cdot J_{n-1} + \frac{2A_n}{b}} - a \cdot W_{n-1} \cdot J_{n-1}$$

in cui J_{n-1}^2 coincide con J_{n-1} poiché può assumere unicamente i valori 0 o 1. Sostituendo all'equazione (6.5), W_n diventa

$$W_n = \sqrt{a^2 \cdot W_{n-1}^2 \cdot J_{n-1} + \frac{2A_n}{b}}$$

Iterando si ha

$$\begin{aligned} W_n^2 &= a^2 \cdot W_{n-1}^2 \cdot J_{n-1} + \frac{2A_n}{b} \\ &= a^4 \cdot W_{n-2}^2 \cdot J_{n-1} \cdot J_{n-2} + \frac{2(a^2 \cdot J_{n-1} \cdot A_{n-1} + A_n)}{b} \end{aligned}$$

Proseguendo con le iterazioni si giunge all'equazione (6.6).

***E*[*W*_∞] (Tahoe)**
Per Tahoe $a = 0$ e $K^0 = 0$, quindi

$$W_\infty = \sqrt{\frac{2V_0}{b \cdot p}} \quad \Rightarrow \quad V_0 = \frac{W_\infty^2 \cdot b \cdot p}{2}$$

da cui derivando e sostituendo si sottiene la funzione di densità di W

$$f_W(v) = (b \cdot p \cdot v) \cdot e^{-\frac{b \cdot p \cdot v^2}{2}}, \quad v \geq 0$$

con

$$E[W_\infty] = \sqrt{\frac{\pi}{2 \cdot b \cdot p}}$$

dove π è ottenuto tramite la funzione Gamma (Γ) di Eulero.

***Q* (Tahoe)**

Evitare un TO in Tahoe implica ricevere almeno k DUPACK nel ciclo n:

$$\begin{aligned} 1 - Q_n &= P(B_n \geq k) \\ &= P(X_n \geq k, B_n \geq k) \end{aligned}$$

Ponendo $X_n = x$, ricordando (6.8) e applicando la probabilità condizionata, si ha

$$\begin{aligned} P(B_n \geq k \mid X_n = x) &= \sum_{i=k}^{x} \binom{x}{i} \cdot (1-p)^i \cdot p^{x-i} \\ &\approx 1 - \binom{x}{k-1} \cdot p^{x-k+1} \end{aligned} \tag{A.1}$$

Approssimando:

$$Q_n = 1 - E\left[\left(1 - \binom{X_n}{k-1} \cdot p^{X_n - k + 1}\right); X_n \geq k\right] \tag{A.2}$$

in cui si ricorda che $E[(Z;A)] = E[Z \cdot 1_A]$ denota il valore atteso di Z sull'insieme A, con $1_A : Y \to \{0,1\}$ funzione indicatrice di A. Ponendo $k = 3$ ed approssimando adeguatamente gli integrali si ricava Q per RED

$$Q = 1 - \int_3^\infty \left(1 - \frac{1}{2} \cdot (v-1) \cdot (v-2) \cdot p^{v-3}\right) \cdot f_W(v)\, dv \tag{A.3}$$

e DropTail

$$Q = 1 - \int_3^\infty \left(1 - \frac{1}{2} \cdot x \cdot (x-1) \cdot p^{x-2}\right) \cdot f_C(x)\, dx \tag{A.4}$$

dove f_W e f_C sono le funzioni di densità di probabilità di, rispettivamente, W e C.

A.1.3 Compendio risultati

Tahoe

$$
\begin{aligned}
E[W_\infty] &= \sqrt{\frac{\pi}{2bp}} \\
E[Y_\infty] &= (1-p)\cdot\sqrt{\frac{\pi\cdot b}{2p}} \\
E[Z_\infty^{FR}] &= 1 \\
E[B_\infty] &= (1-p)\cdot\frac{E[W_\infty]}{2} && \text{(DropTail)} \\
E[B_\infty] &= (1-p)\cdot(E[W_\infty]-1) && \text{(RED)}
\end{aligned}
$$

N.B.: la probabilità di timeout Q, non esplicitata, è differente per i due ambienti AQM e viene calcolate in base alla funzione di densità di probabilità delle variabili aleatorie che considerano il numero di pacchetti inviati con successo precedentemente e successivamente alla prima perdita.

NewReno

$$
\begin{aligned}
E[W_\infty] &= \sqrt{\frac{2}{bp}}\cdot\frac{2}{\sqrt{3+Q}} \\
E[Y_\infty] &= b\cdot\left(1-\sqrt{\frac{2}{bp}}\cdot\frac{1-Q}{\sqrt{3+Q}}\right) \\
E[Z_\infty^{FR}] &= 1+\frac{1}{4}\cdot E[W_\infty] && \text{(DropTail)} \\
E[Z_\infty^{FR}] &= 1+\frac{5}{24}\cdot E[W_\infty]\cdot p && \text{(RED)} \\
E[B_\infty] &= (1-p)\cdot\frac{E[W_\infty]}{2} && \text{(DropTail)} \\
E[B_\infty] &= (1-p)\cdot(E[W_\infty]-1) && \text{(RED)} \\
Q &= 1-(1-p)\cdot\int_3^\infty g(v)f_W(v)\,dv
\end{aligned}
$$

Anche in questo caso la probabilità di timeout Q si differenzia tra i due

ambienti di AQM, mostrando valori diversi per la funzione $g(v)$:

$$g(v)_{DropTail} = \frac{1-(1-p)^{v-3}}{-v\log(1-p)}$$
$$g(v)_{RED} = (1-\frac{1}{2}\cdot(v-1)\cdot(v-2)\cdot p^{v-3})\cdot\frac{1-(1-p^2)^v}{-v\log(1-p^2)}$$

Tale funzione rappresenta la densità di probabilità per la variabile aleatoria B_n.

A.2 Paper [35] (*D. J. Leith*)

L'equazione (4.22) descrive le dinamiche di una collezione di flussi della stazione wireless s, in cui

$$W_s(k) = \begin{bmatrix}\gamma_{s,1}\cdot E[w_{s,1}(k)] & \cdots & \gamma_{s,1}\cdot E[w_{s,n_s}(k)]\end{bmatrix}^T$$

mentre

$$A_s = \begin{bmatrix} E[b_{s,1}] & 0 & \cdots & 0 \\ 0 & E[b_{s,2}] & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & E[b_{s,n_s}] \end{bmatrix} + \frac{1}{\sum_{j=1}^{n_s}\gamma_{s,j}E[a_{s,j}]}\cdot\begin{bmatrix}\gamma_{s,1}\cdot E[a_{s,1}]\\ \vdots \\ \gamma_{s,n_s}\cdot E[a_{s,n_s}]\end{bmatrix}\cdot\begin{bmatrix}1-E[b_{s,1}], & \cdots & ,1-E[b_{s,n_s}]\end{bmatrix}$$

con condizione iniziale che soddisfa $\sum_{i=1}^{n_s}\gamma_{s,i}E[w_{s,i}(0)] = E[B_s]$. La quantità $E[a_{s,i}]$ viene calcolata approssimandola con $\gamma_{s,i}\cdot\alpha_{s,i}$. Per calcolare $E[b_{s,i}]$ occorre invece considerare la probabilità che un flusso i sia affetto da backoff in caso di congestione, $\lambda_{s,i}$ (nota a priori); segue quindi che $E[b_{s,i}] = \lambda_{s,i}\cdot(1-\beta_{s,i})$.

Bibliografia

[1] RFC 793 - Transmission Control Protocol

[2] RFC 5681 - TCP Congestion Control (PRP STD)

[3] Mirco Pedrini (2011) - TCP a Ritrasmissione Asimmetrica Anticipata su WiFi

[4] Paolo Alberti (2012) - Analisi per Via Simulativa del Protocollo TCP a Ritrasmissione Asimmetrica Anticipata su Wi-Fi

[5] Claudia Minardi (2012) - TCP a Ritrasmissione Anticipata Tramite Access Point Wi-Fi

[6] S. Ferretti, V. Ghini, M. Marzolla, F. Panzieri (2012) - Modeling the Always Best Packet Switching Mechanism

[7] S. Ferretti, V. Ghini, M. Marzolla, F. Panzieri (2009) - Always Best Packet Switching: the Mobile VoIP Case Study

[8] J. Olsén (2003) - Stochastic Modeling and Simulation of the TCP Protocol

[9] Eli Brosh (2007) - Stochastic Modeling of the TCP Protocol

[10] M. Mathis, J. Semke, J. Mahadvi (1997) - The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm

[11] J. Padhye, V. Firoiu, D. Towsley, J. Kurose (1998) - Modeling TCP Throughput: A Simple Model and its Empirical Validation

[12] J. Padhye, V. Firoiu, D. Towsley, J. Kurose (2000) - Modeling TCP Reno performance: A Simple Model and its Empirical Validation

[13] Anurag Kumar (1998), Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link

[14] N. Cardwell, S. Savage, T. Anderson (2000) - Modeling TCP Latency.

[15] I. Kaj, J. Olsén (2001) - Throughput Modeling and Simulation for Single Connection TCP-Tahoe

[16] I. Kaj, J. Olsén (2001) - Stochastic Equilibrium Modeling of the TCP Dynamics in Various AQM Environments

[17] I. Yeom, A. L. N. Reddy (2001) - Modeling TCP Behavior in a Differentiated Services Network

[18] N. Malouch, Z. Liu (2001) - On Steady State Analysis of TCP in Networks with Differentiated Services

[19] A. Misra, J S. Baras, T. Ott (1999) - Window Distribution of Multiple TCPs with Random Loss Queues

[20] V. Firoiu, M. Borden (2000) - A Study of Active Queue Management for Congestion Control

[21] T. Bu, D. Towsley (2001) - Fixed Point Approximations for TCP Behaviour in an AQM network

[22] V. Firoiu, I. Yeom, X. Zhang (2001) - A Framework for Pratical Performance Evaluation and Traffic Engineering in IP Networks

[23] C. Casetti, M. Meo (2000) - A New Approach to Model the Stationary Behavior of TCP Connections

[24] C. Casetti, M. Meo (2001) - An Analytical Framework for the Performance Evaluation of TCP Reno Connections

[25] V. Misra, W.-B. Gong, D. F. Towsley (2000) - Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED

[26] E. Altman, K. Avrachenkov, C. Barakat (2000) - A Stochastic Model of TCP/IP with Stationary Random Losses

[27] E. Altman, T. Jiménez, R. Nunez-Queija (2002) - Analysis of Two Competing TCP/IP Connections

[28] A. Riedl, T. Bauschert, M. Perske, A. Probst (2000) - Investigation of the M/G/R Processor Sharing Model for Dimensioning of IP Access Networks with Elastic Traffic

[29] S. B. Fredj, T. Bonald, A. Proutiere, G. Régnié, J. W. Roberts (2001) - Statistical Bandwidth Sharing: A Study of Congestion at Flow Level

[30] R. Vranken, R. D. van der Mei, R. E. Kooij, J. L. van der Berg (2002) - Performance of TCP with Multiple Priority Classes

[31] S. H. Low, D. E. Lapsley (1999) - Optimization Flow Control, I: Basic Algorithm and Convergence

[32] S. H. Low, S. Athuraliya (2000) - Optimization Flow Control, II: Implementation

[33] S. H. Low (2000) - A Duality Model of TCP and Queue Management Algorithms

[34] C. V. Hollot, V. Misra, D. F. Towsley, W. Gong (2001) - A Control Theoretic Analysis of RED

[35] D. J. Leith (2005) - Modelling TCP Dynamics in Wireless Networks

[36] F. Biagini, M. Campanino (2006) - Elementi di Probabilità e Statistica

[37] SLIDING WINDOW DEMO - `http://www2.rad.com/networks/2004/sliding_window/`