STUDIO SIMULATIVO DI TRAFFICO VIDEO STREAMING SCALABILE IN MODALITÀ BROADCAST

RELATORE: Leonardo Badia

CORRELATORI: Daniele Munaretto

LAUREANDO: Luca Callegher

A.A. 2011-2012

UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
TESI DI LAUREA TRIENNALE IN INGEGNERIA DELL'INFORMAZIONE

STUDIO SIMULATIVO DI TRAFFICO VIDEO STREAMING SCALABILE IN MODALITÀ BROADCAST

RELATORE: Leonardo Badia

CORRELATORI: Daniele Munaretto

LAUREANDO: *Luca Callegher*

Padova, 24 luglio 2012

Indice

Sommario

Le reti cellulari di ultima generazione (3.9-4G) daranno la possibilità di accedere a contenuti video streaming in modalità broadcast. Parlando di broadcast si ha sempre a che fare con utenti in situazioni eterogenee: alcuni godranno di buon SNR, altri meno, principalmente in dipendenza dalla loro distanza rispetto alla stazione base. Obiettivo di questa tesi è quello di analizzare un possibile approccio allo streaming broadcast (diverso da quello standard) che garantisca la miglior qualità video consentita dalle condizioni del canale. Tale approccio si basa su:
(i) proprietà di scalabilità del traffico video;
(ii) scelta opportuna della modulazione e della codifica a livello MAC.
Daremo particolare risalto all'aspetto simulativo del nostro lavoro. Lo scenario sul quale andremo ad effettuare le simulazioni è quello di un'unica stazione base in grado di servire N utenti posti a distanze via via crescenti.

Capitolo 1

Introduzione

Il traffico video sul web sta da tempo ricoprendo, in termini di mole di dati trasmessi, una posizione di primo piano (51% del traffico totale nel 2012 [2], *peer-to-peer file sharing* escluso) e, secondo le stime più recenti, il suo peso è destinato ad aumentare (54% del totale nel 2016, [2]). Sempre secondo le stesse previsioni inoltre, il numero di utenti che accederanno a risorse internet tramite dispositivi mobili è destinato ad crescere di 18 volte nello stesso arco di tempo. Le reti cellulari di ultima generazione (LTE) daranno la possibilità di accedere a contenuti video streaming in modalità broadcast [3].
L'acronimo LTE (Long Term Evolution) fa riferimento a quell'insieme di standard che rappresentano la naturale evoluzione di tecnologie cellulari come GSM e UMTS: non si tratta di una vera e propria tecnologia 4G, piuttosto si colloca in una posizione intermedia fra la terza e la quarta generazione, [11]. Fra le varie possibilità messe a disposizione da LTE [10] figura appunto quella di poter fornire efficacemente traffico in modalità broadcast. Il tutto rientra in un progetto più ampio conosciuto come MBMS (Multimedia Broadcast Multicast Service) [3]. In modalità broadcast un pacchetto, spedito una volta per tutte, può essere ricevuto da un numero potenzialmente molto grande di utenti, con conseguente risparmio di risorse radio.
Il codec SVC (Scalable Video Codec) ha la capacità di codificare flussi video scalabili [6], ossia composti da differenti livelli di qualità. SVC introduce la nozione di layer all'interno di uno stream, concetto che si rivelerà particolarmente utile ai fini della nostra trattazione. Vedremo quindi come sia possibile sfruttare le caratteristiche della tecnologia LTE e le proprietà del traffico video per realizzare

un approccio innovativo [1] allo streaming broadcast. Tale approccio si basa sulla combinazione di due principi:
(i) il flusso video, a differenza di altri tipi di traffico dati, presenta caratteristiche peculiari che gli consentono di essere "scalato" [6] a livello temporale, spaziale e in termini di qualità percepita;
(ii) un opportuno scheduling unito a un'opportuna scelta della modulazione e della codifica a livello MAC danno la possibilità di selezionare il sub-stream più adatto a ciascun utente finale ([1]).
Il nostro studio sarà supportato dall'utilizzo del simulatore LENA (LTE-EPC network simulator)[21], basato su ns-3 [18] e sviluppato dal CTTC (Centre Tecnologic de Telecomunicacions de Catalunya) [17] in collaborazione con Ubiquisys (sviluppatore di intelligent cells) [16]. Nello specifico le metriche di valutazione che andremo a prendere in esame saranno *throughput* e *delay* al variare di distanza degli utenti dalla stazione base, modulazione usata e dimensione dei pacchetti inviati.

Il seguito di questo lavoro è organizzato in questo modo: nel **Capitolo 2** definiremo le proprietà di scalabilità del traffico video, anche in riferimento a SVC. Sempre nel **Capitolo 2** forniremo una panoramica delle caratteristiche base di LTE, tecnologia sulla quale supporremo di effettuare le simulazioni. Il **Capitolo 3** è invece dedicato al lavoro simulativo. Nelle sue sottosezioni prenderemo in esame, nell'ordine, il simulatore open source ns-3, la sua versione orientata alle reti LTE (LENA) e le simulazioni *ad hoc* da noi realizzate. Nel **Capitolo 4** forniremo nello specifico i parametri con cui le simulazioni sono state impostate. Particolare risalto sarà dato ai risultati numerici riportati nei grafici. Giungeremo infine alla conclusioni mettendo in evidenza quelli che possono essere aspetti e tematiche di un'eventuale indagine futura nel **Capitolo 5**.

Capitolo 2

Il contesto

2.1 Il traffico video streaming

Com'è noto uno streaming video è un flusso di dati audio/video trasmessi da un provider e riprodotti dall'utente finale solitamente in tempo reale (o quasi), ossia man mano che questi giungono a destinazione [5]. Spesso tali dati sono salvati e spediti in forma compressa, per risparmiare spazio su server e banda in trasmissione. Per la loro visualizzazione non è richiesto il download dell'intero file [5], ma al contempo ciò introduce requisiti stringenti per quanto che concerne il ritardo (*delay*) nella ricezione dei pacchetti (application time-sensitive), anche nel caso di modalità broadcast. Ricordiamo che si parla di broadcast quando una sorgente, la nostra stazione base ad esempio, trasmette simultaneamente a tutti i potenziali destinatari che si trovano nella zona. Protocolli basati, a livello del trasporto, su TCP (Transmission Control Protocol) dominano rispetto a quelli basati su UDP (User Datagram Protocol), essendo usati per circa il 70% dei byte trasferiti [8]. Nel nostro caso (modalità broadcast) tuttavia riteniamo opportuno utilizzare UDP, che meglio si adatta alle nostre necessità: si preoccupa della consegna dei pacchetti nella maniera più celere possibile, senza tuttavia gestire riordinamento e ritrasmissione di quelli persi [9]. Sarà dunque il protocollo che andremo ad utilizzare nelle simulazioni.

2.1.1 Scalabilità

Per scalabilità in riferimento a un flusso video intendiamo la possibilità di rimuovere in maniera opportuna parte dell'informazione contenuta nello stream

originale, ottenendo così un sub-stream di dimensioni inferiori ma pur sempre riproducibile [6]. Nello specifico, sempre secondo [6] si parla di scalabilità:

- A livello **temporale** quando dalla sequenza complete sono rimosse alcune frame (l'equivalente dei fotogrammi). Il sub-stream così ottenuto presenta quindi un frame rate inferiore.
- A livello **spaziale** quando viene ridotta la dimensione delle immagini. Si può passare ad esempio da 1600x1200 a 640x480 pixel.
- A livello di **qualità percepita** quando il video viene codificato con fedeltà inferiore.

È inoltre possibile una combinazione di tutte e tre queste alternative.

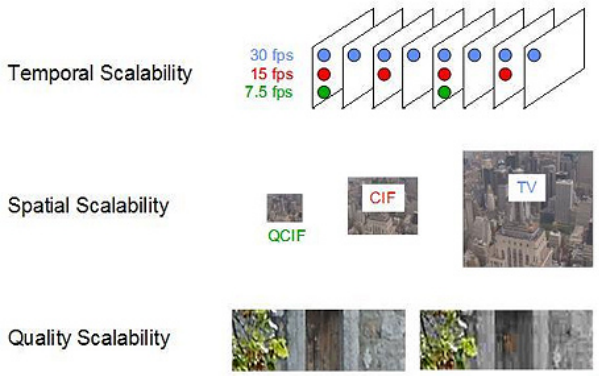

Figura 2.1: Esempi di scalabilità

A questo punto risulterebbero particolarmente interessanti le seguenti proprietà:

- avere a disposizione uno stream di alta qualità (HQ), codificato una volta per tutte, contenente al suo interno uno o più stream scalati in uno dei sensi appena citati.
- La possibilità di passare da una risoluzione ad un'altra semplicemente rimuovendo alcuni pacchetti piuttosto che altri dalla sequenza, senza alcun processo addizionale.

Sono esattamnete questi ultimi gli obiettivi raggiunti dal codec SVC.

2.1.2 Il codec H.264 SVC

SVC (nome completo H.264 SVC - scalable video codec), sviluppato dal Joint Video Team(JVT), è in grado di codificare video stream di alta qualità contenenti uno o più substream [7]. Un sub-stream è ricavato rimuovendo pacchetti dalla sequenza originale e occupa pertanto minor banda in trasmissine. Per ciò che concerne la codifica, SVC sfrutta ampiamente gli strumenti del preesistente AVC(Advanced Video Codec) [6], la sua versione non scalabile. Introduce tuttavia la nozione di layer, non presente in AVC. Un base layer, com'è intuibile, codifica il sub-stream con risoluzione/fedeltà minore. Uno o più enhancement layers (e-layers) possono contenere informazione aggiuntiva e, cosa veramnte importante, sia in fase di codifica che di decodifica fanno sempre riferimento a layer che li precedono nella gerarchia. In altri termini un e-layer non può essere decodificato in maniera a se stante, ma sempre in relazione ad esempio al layer base, garantendo così dimensioni relativamente ridotte per l'e-layer stesso. Il flusso video può essere in qualsiasi momento troncato, ossia privato di un certo numero di layers scartando, come già si accennava, i pacchetti corrispondenti. Quest'ultima operazione non necessariamente è effettuata a livello applicazione, ma, come vedremo, può avvenire anche a "livello MAC". In altre parole anche la "rete" può opportunamente selezionare a chi inviare quali pacchetti.

2.2 La tecnologia LTE

A cavallo fra 2004 e 2005 il Third Generation Partnership Project (3GPP), consorzio che raggruppa svariati enti impegnati nell'ambito delle telecomunicazioni, inizia gli studi riguardanti la definizione di LTE, standard per sistemi di accesso mobile a banda larga [12]. Fra gli obiettivi principali del progetto figura quello di incrementare i rate di downlink e uplink portandoli rispettivamente a 150 Mbps e 50 Mbps [11], con riserva di raggiungere velocità ancora maggiori sfruttando modulazioni più efficienti e/o configurazioni MIMO (Multiple Input Multiple Output) per le antenne. LTE, la cui documentazione ufficiale si trova nella Relaese 8 e in parte nella Release9 delle 3GPP specificaitions, da un punto di vista concettuale si colloca fra le tecnlogie di terza(3G) e quarta (4G) generazione [11]. La sua standardizzazione è terminata nel 2008 e già a partire dall'anno suc-

cessivo era commercialmente disponibile a Oslo e Stoccolma [4]. La *Figura 2.2* mostra gli stati che hanno adottato/stanno adottando LTE ed evidenzia come tale tecnologia si stia affermando globalmente.

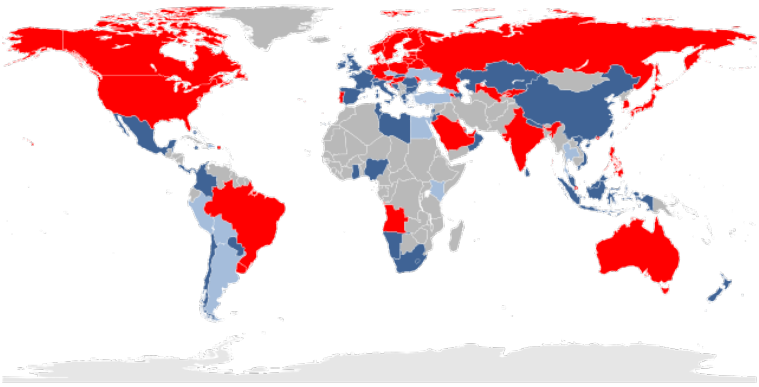

Figura 2.2: Adoption of LTE technology as of May 8, 2012. Red: Countries with commercial LTE service Blue: Countries with commercial LTE network deployment on- going or planned Light-blue: Countries with LTE trial systems (pre-commitment). From `http://en.wikipedia.org/wiki/LTE_(telecommunication)`

Una descrizione dettagliata di tutti i concetti relativi a LTE esula dagli scopi di questa tesi. Per questo rimandiamo ad esempio a [10, 12]. Ci limitiamo invece a fornire una panoramica generale focalizzata sulle caratteristiche importanti dal punto di vista della nostra trattazione.

2.2.1 Ue, EnB, stack protocollare

Ue (*User equipment*) identifica un qualsiasi dispositivo mobile in grado di connettersi alla stazione base, detta anche EnB (*Evolved node B*). Lo stack protocollare, oltre ai classici PHY e MAC comprende i seguenti *sub-layers* [11]:

- **RLC:** *Radio Link Control* - opera segmentazione e riassemblaggio in relazione alla dimensione del *Transport Block* (TB). Gestisce inoltre la consegna ordinata e il controllo di doppioni. Può operare in tre differenti modalità: *transparent mode* (TM), *Acknowledge mode* (AC) e *unacknowledge mode* (UM).

- **PDCP**: *Packet Data Convergence Protocol* - svolge funzioni come compressione dell'*header*, numerazione della sequenza e rimozione di doppioni.
- **RRC:** *Radio Resource Control* - gestisce prevalentemente inizializzazione e rilascio del collegamento radio.

2.2.2 Interfaccia aerea e modulazioni

I canali di *uplink* e *downlink* dell'interfaccia aerea sono organizzati nei seguenti elementi [13]:

- una **frame** della durata di 10 ms è ripartita nel dominio del tempo in 10 *subframe*;
- una **subframe** della durata di 1 ms, divisa, sempre nel dominio del tempo, in 2 *slots*, è uno degli l'intervalli di tempo fondamentali di trasmissione;
- una **slot** (0.5 ms) è divisa nel dominio della frequenza in un numero variabile di RBs (*Resource Blocks*), dipendentemente dall'ampiezza di banda del canale;
- un **RB** (0.5 ms), composto da 6 o 7 simboli (numero dipendente dal *prefix cycle* usato. Un *prefix cycle* non è alto che un intervallo di tempo di scurezza fra un simbolo e il successivo, introdotto per evitare ISI) nel dominio del tempo e da 12 *sub-carriers* in quello della frequenza, rappresenta l'unità fondamentale per ciò che riguarda l'allocazione delle risorse.

Questo schema, sfruttato per gestire l'accesso al mezzo da parte di numerosi Ue, si adatta facilmente al caso broadcast: una certa stazione base può dedicare interamente le sue risorse alla trasmissione broadcast oppure, identificando opportunamente i relativi RBs, servizi *unicast* e broadcast possono coesistere [3]. Definiamo *Transport Block* (TB) un gruppo di RBs ai quali sono applicate codifica e modulazione comuni. Le modulazioni disponibili alla EnB sono 64-QAM, 16-QAM e QPSK [10], caratterizzate nell'ordine da crescente affidabilità ma decrescente *symbol rate*. In modalità *unicast* gli utenti inviano un indicatore della qualità del canale (*cqi*) a EnB [12]: questo meccanismo di *feedback* consente alla stazione base di scegliere la modulazione più adatta a ciascun Ue. Tale approccio non è chiaramente applicabile in broadcast classico, tutti i pacchetti sono quindi inviati con la modulazione più bassa.

Capitolo 3

Il lavoro simulativo

3.1 Contributo della tesi

La possibilità, offerta dalla tecnologia LTE, di ricevere contenuti in modalità broadcast si colloca all'interno di un progetto più ampio conosciuto col nome di MBMS (*Multimedia Broadcast Multicast Service*) [3] secondo quanto specificato dal 3GPP. Vantaggio chiave di questo tipo di servizi è chiaramente quello di fornire ad un potenzialmente elevato numero di Ue contenuti trasmessi in *downlink* una volta per tutte, con notevole risparmio di risorse radio. La più grande limitazione è invece la seguente: in un approccio broadcast classico tutti i pacchetti sono trasmessi con una modulazione bassa, che sappiamo essere resistente ai disturbi introdotti dal canale, ma meno efficiente relativamente alla mole di dati trasmessi. Questo, se da un lato consente di servire anche utenti che non godono di SNR particolarmente buono (tipicamente utenti molto lontani dall'EnB), dall'altro fornisce contenuti video che di certo non raggiungono la massima QoS (*Quality of Service*) possibile. L'approccio alternativo, che trae spunto dal lavoro [1], analizzato in questa tesi mira a garantire a ciascun utente una QoS proporzionale alle condizioni del suo canale. Tale obiettivo può essere raggiunto combinando le caratteristiche di scalabilità proprie del traffico video con un'opportuna scelta delle modulazioni da associare a ciascun layer. L'idea di base è molto semplice. Supponiamo ad esempio che il nostro stream scalabile sia composto da tre tipi di pacchetto:

- *Pacchetti A:* sono i più importanti e devono essere ricevuti da tutti per una corretta visualizzazione. La modulazione a loro più adatta è quindi una

QPSK. Riprendendo il linguaggio di SVC diremo che essi codificano il layer base.

- *Pacchetti B:* la loro importanza è intermedia. Nel linguaggio di SVC corrispondono ad un primo enhancement-layer. Andremo ad assocare loro una 16-QAM.

- *Pacchetti C:* supporremo che la loro corretta ricezione, assieme a quella degli altri due tipi, consenta la visualizzazione di un video di alta qualità. Corrispondono infatti a un secondo e-layer. Andremo ad inviarli con una modulazione alta (64-QAM).

In questo modo gli utenti più lontani riceveranno solo i pacchetti A e godranno di una QoS base. Quelli che invece si trovano nel raggio di copertura della 64-QAM potranno usufruire della miglior QoS disponibile, senza tuttavia violare i requisiti sul *delay*. Le modulazioni alte infatti, in virtù del loro maggior symbol rate, garantiranno la consegna dei pacchetti in un tempo accettabile. La scelta di avere tre tipi di pacchetto è stata puramente dettata dal fatto che tre sono le modulazioni a disposizione. Nulla vieta tuttavia, magari in studi futuri, di sperimentare soluzioni diverse. Precisiamo inoltre che la tecnologia LTE fornisce qualcosa di più raffinato: si parla propriamente infatti di *Adaptive Modulation and Coding scheme* (*amc*), tecnica che consente di adattare non solo la modulazione, ma anche la codifica alle condizioni del canale. In totale gli schemi *amc* disponibili sono 28. Noi abbiamo scelto tre di questi schemi (6,12,26) e li abbiamo mantenuti fissi nel corso delle simulazioni.

3.2 Il simulatore ns-3

ns-3 (in accordo con quanto riportato nella home page del sito ufficiale, [18]) è un simulatore di reti a eventi discreti pensato principalmente per la ricerca e la didattica, liberamente disponibile disponibile sotto licenza GNU GPLv2. ns-3 è organizzato sotto forma di libreria software, scritta in C++, alla quale può essere collegato il programma contenete il main. Tale programma a sua volta serve a definire topologia, parametri e a dare inizio alla simulazione. La *Figura 3.1* mostra la struttura gerarchica del simulatore. Nel seguito andremo a definire alcuni dei più importanti concetti, fra i quali quelli di *Tracing, Logging,*

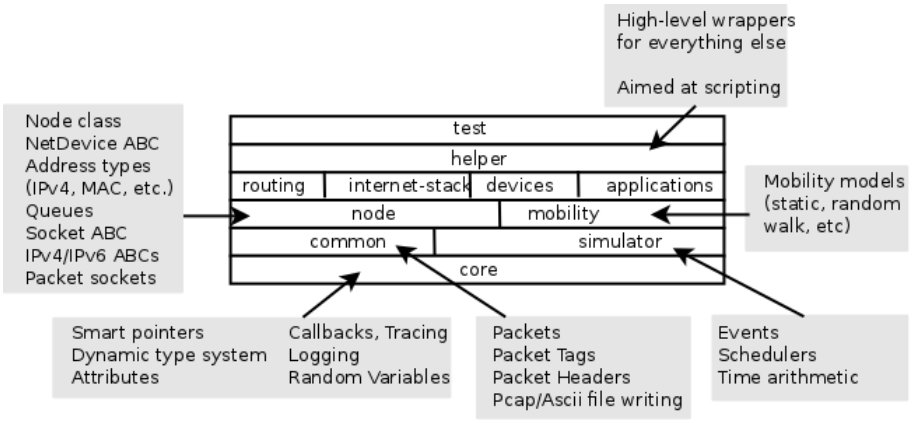

Figura 3.1: Struttura a moduli di ns-3, [20]

Attributes e *Callbacks*. Prima però prendiamo in esame alcune delle astrazioni chiave. Si tratta di nozioni comuni in telecomunicazioni ma in ns-3 assumono il loro particolare significato ([19]).

3.2.1 Nodi

Il nodo, rappresentato in C++ dalla classe *Node*, è l'astrazione base per il generico dispositivo computazionale. Ad un nodo possono essere aggiunte funzionalità quali applicazioni, stack protocollari e dispositivi per la connessione con altri nodi.

3.2.2 Applicazioni

La classe C++ *Applications* fornisce i metodi base per quelle che possono essere pensate come applicazioni del livello più alto dello stack ISO-OSI. Si tratta cioè di software in grado di generare una qualche attività che andrà poi simulata. Si suppone che gli sviluppatori definiscano, a livello di programmazione orientata agli oggetti (OOP), le loro specifiche applicazioni, cosa che faremo anche noi per generare un pattern prestabilito di pacchetti che rispettino determinate specifiche. Per funzionare correttamente le applicazioni devono essere installate sui nodi.

3.2.3 Canali di comunicazione

La classe *Channel* è il punto di partenza per modellare un generico link fra due o più nodi. Questo link può essere un semplice cavo come anche un ambiente ricco di ostacoli nel quale si propaga un segnale in comunicazioni *wireless*. Il simulatore mette a disposizione numerose versioni specializzate della classe *Channel*. Andremo a sfruttare tali versioni, senza doverci preoccupare della loro effettiva implementazione.

3.2.4 Net Devices

Una *Net Device* è l'astrazione che consente di interfacciare l'un l'altro i nodi della rete, attraverso i canali. Come ormai il lettore avrà intuito, è rappresentata in C++ dalla classe *NetDevice* che, anche in questo caso, può essere specializzata per ottenere ad esempio una *PointToPoitNetDevice*, pensata per lavorare con un *PointToPointChannel*, o una *WifiNetDevice* associata chiaramente a un *WifiChannel*.

Un simulatore di modeste dimensioni deve poter fornire sistemi di accesso alle informazioni, ai messaggi di errore/debug, e alla modifica dei parametri che non generino interdipendenze fra moduli distinti, in modo da garantire non solo la riusabilità del codice, ma anche flessibilità e vincoli non troppo stretti in fase di compilazione. Sono questi i criteri coi quali sono progettati i meccanismi di *Tracing, Logging* e *l'Attribute System* ([20]).

3.2.5 Tracing sub-system

Il motivo principale per cui si esegue un simulazione è chiaramente quello di ricavarne dati e statistiche (solitamente per avvalorare o meno una data ipotesi, teoria o modello). Il meccanismo che consente tutto ciò in ns-3 è appunto il *Tracing sub-system*, costruito sui concetti di *"tracing sources", "tracing sinks"* e *"callbacks"*.

- Una *trace source* è un'entità che può segnalare l'avvenimento di una particolare azione e consentire l'accesso ai dati relativi. Può ad esempio segna-

lare l'arrivo di un pacchetto ad un qualche livello dello stack protocollare e fornire informazioni su dimensioni e ritardo del pacchetto stesso.

- Un *trace sink* è quel meccanismo che, una volta connesso ad una trace source, da la possibilità di estrarre le informazioni disponibili, elaborarle e volendo stamparle su file.

- Una *callback* è il meccanismo generico per connettere una source al suo sink. Senza scendere eccessivamente nei dettagli realizzativi, possiamo dire che è basata sul concetto di chiamata ad una funzione per mezzo del su puntatore. È quindi legata più alla OOP che a concetti di reti.

Il fondamento logico che motiva questo apparentemente macchinoso congegno è il seguente: consentire agli utenti di connettere i propri sinks a preesistenti sources senza dover rieditare o ricompilare il core del simulatore. Inoltre è un modo per ricavare informazione in maniera selettiva: se tutti gli oggetti della simulazione generassero indiscriminatamente i loro output, il caos risultante sarebbe ingestibile.

3.2.6 Logging module

è lo strumento per gestire, a vari livelli di dettaglio, i messaggi provenienti dal simulatore. Le sue funzioni si sovrappongono in parte con quelle del *Tracing subsystem*. Gli sviluppatori di ns-3 tuttavia consigliano [19] di usarlo per informazioni di debug, messaggi di errore e warnings, nonchè ogni volta si desideri ottenere rapidamente messaggi mirati dai nostri script. Attualmente sono disponibili sette livelli di accesso alle informazioni, abilitabili indipendentemente l'uno dall'altro:

```
NS_LOG_ERROR - Log error messages;
NS_LOG_WARN - Log warning messages;
NS_LOG_DEBUG - Log relatively rare, ad-hoc debugging messages;
NS_LOG_INFO - Log informational messages about program progress;
NS_LOG_FUNCTION - Log a message describing each function called;
NS_LOG_LOGIC - Log messages describing logical flow within a function;
NS_LOG_ALL - Log everything.
```

3.2.7 Attribute system

L' *Attribute System* si pone l'obiettivo di organizzare l'accesso a variabili private (nel senso informatico del termine), non altrimenti raggiungibili. Viene usato principalmente per mdificare i parametri della simulazione evitando di dover inserire puntatori ad oggetti nel codice sorgente, per il solito discorso di riusabilità e flessibilità. Un esempio tipico (di riconfigurazione dei parametri di default) è il seguente:

```
Config::SetDefault ("ns3::PointToPointNetDevice::DataRate", StringValue ("5Mbps"));
```

Il codice si spiega da solo.

3.3 Il simulatore LENA

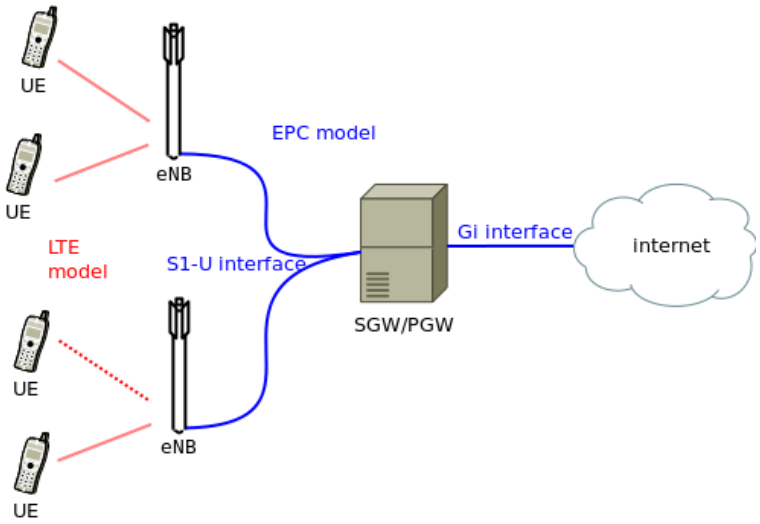

Figura 3.2: Architettura del simulatore LENA, [15]

LENA (LTE-EPC Network Simulator) è un simulatore di reti LTE (primo al mondo ad essere open [17]) sviluppato a partire dal dicembre 2010 dal CTTC (*Centre Tecnologic de Telecomunicacions de Catalunya*) [17] in collaborazione con Ubiquisys (*small-cells and femto-cells technology company*) [16] . Sebbene si presenti come un progetto a se stante, eredita la stragrande maggioranza dei concetti (e del codice) da ns-3. Ciò che lo differenzia da ns-3 è appunto la sezione

riguardante LTE-EPC appositamente sviluppata con un grado piuttosto elevato di fedeltà [21]. L'architettura generale del simulatore è illustrata in *Figura 3.2*. Si distinguono chiaramente il modulo LTE, che include l'LTE *radio protocol* stack (*Figura 3.3*) e il modello EPC. Per inciso, EPC (*Evolved Packet Core*) è l'interfaccia fra una rete LTE una generica rete IP-*based*. Lo stack protocollare radio LTE comprende i classici layer e sub-layer PHY, MAC, RLC, PDCP, RRC. La *Figura 3.4* offre una visione più dettagliata del lower LTE protocol stack così com'è implementato nel simulatore. Va detto che, per ciò che concerne l'*upper* stack, i sub-layer RLC e PDCP sono accuratamente modellati nel rispetto delle specifiche del 3GPP. RRC è invece sviluppato in una maniera molto semplificata. Per una descrizione maggiormente dettagliata dei componenti e per una visione più generale di quelli che sono i criteri con cui i vari moduli sono stati concepiti rimandiamo a [14] o a [15].

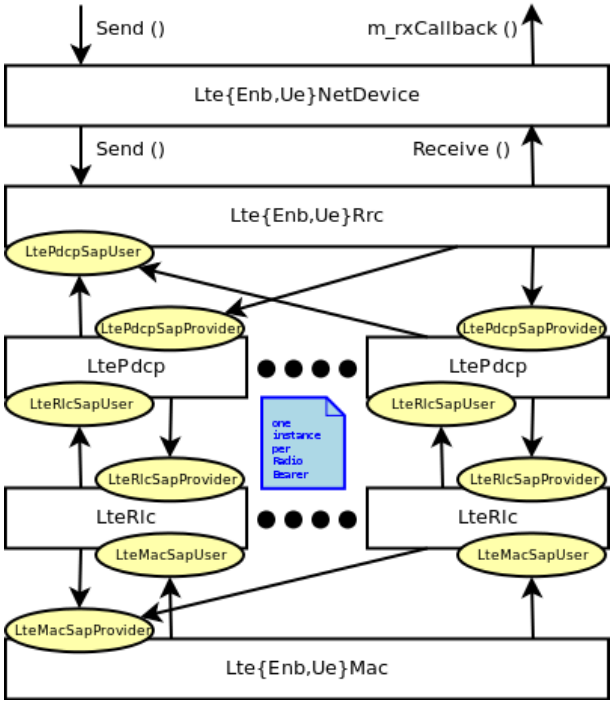

Figura 3.3: Stack protocollare LTE così com'è implementato nel simulatore, [15]

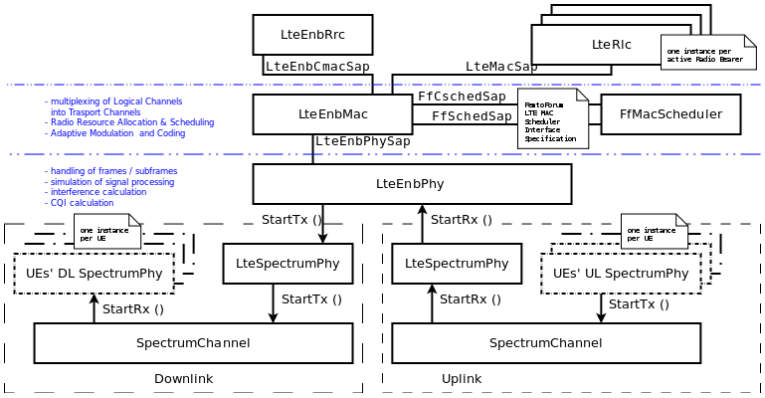

Figura 3.4: Particolare dello stack protocollare LTE, [21]

3.4 Set-up della simulazione

3.4.1 Definizione dello scenario

Si tratta di uno scenario estremamente semplice in cui un unico EbB trasmette ad Ue posizionati a distanze via via crescenti. Gli utenti non sono disposti in maniera uniforme su tutto lo scenario ma si è scelto di addensarne la presenza in due zone critiche. Si tratta infatti di zone di transizione che delimitano regioni coperte da modulazioni diverse (come sarà chiarito in 4.2.2). Tutto questo per ottenere grafici qualitativamente più precisi. L'EnB è inoltre collegato ad un *remote host*, sul quale supponiamo sia immagazzinato il video, tramite un *PointToPointChannel*. Tale canale ha un data rate piuttosto elevato (500 Mbps) e un tempo di propagazione che è idealmente stato posto a zero. È infatti nostra intenzione valutare non tanto le prestazioni di un'eventuale *core network* quanto quelle della rete LTE. Il *remote host* ha dunque il solo scopo di fare da server.

3.4.2 Livello applicazione

Il traffico video formato dai tre tipi di pacchetto A,B,C è fornito da un'applicazione costruita (modificando opportunamente quella presente nel file *fifth.cc* del tutorial di ns-3 [19]) *ad hoc* per i nostri scopi. Tale applicazione è in grado di generare il seguente pattern di pacchetti: un numero arbitrario x di pacchetti A, seguito da un numero arbitrario y di pacchetti B, seguito da un numero z di pacchetti C e ripetere a piacere questa sequenza. Nelle simulazioni abbiamo scelto

di usare il pattern più semplice con $x = y = z = 1$. I pacchetti sono generati a intervalli di 50ms l'uno dall'altro e le loro rispettive dimensioni sono impostabili manualmente.

Capitolo 4

Risultati numerici

4.1 Premessa

Nel seguito prenderemo in considerazione due tipi di simulazione in parte differenti. Il primo, in un'ottica generale, esamina la QoS totale del sistema scegliendo come metriche di valutazione *throughput* e *delay*. Il secondo si focalizza invece su Ue dotati di buon SNR e va a tener conto di variazioni del *delay* in funzione delle dimensioni dei pacchetti inviati. Entrambi i tipi hanno tuttavia i seguenti punti in comune:

- tutti i parametri del livello fisico (come ad esempio *noise figure* di Ue e EnB o modello di propagazione del segnale) sono quelli di default del simulatore;
- lo scenario è quello del punto 3.4.1 (con numero di utenti N che però può variare) e viene impiegata l'applicazione descritta al punto 3.4.2;
- le *amc* con cui sono inviati i pacchetti restano sempre fisse (ricordiamo che sono: (A, 6, QPSK), (B, 12, 16-QAM), (C, 26, 64-QAM) dove i valori di ciascuna tripla indicano nell'ordine tipo di pacchetto, identificativo dell'*amc*, tipo di modulazione);
- i dati numerici riportati nei grafici sono sempre reperiti a livello PDCP con granularità estremamente sottile (cioè sono misurati sul singolo pacchetto). LENA mette a disposizione delle tracce preimpostate (per questo livello) di cui riportiamo un esempio e la relativa descrizione.

```
PDCP Downlink
1.25 1.3 1 1 1 1 1 830 1 830 0.00498 0 0.00498 0.00498 830 0 830 830
1.3 1.35 1 1 1 1 1 430 1 430 0.00299 0 0.00299 0.00299 430 0 430 430
1.35 1.4 1 1 1 1 1 330 1 330 0.00299 0 0.00299 0.00299 330 0 330 330
```

Dove ciascuna colonna rappresenta rispettivamente [21]:

1. Istante iniziale dell'intervallo di tempo in secondi dall'inizio della simulazione in cui è effettuata la misura
2. Istante finale dell'intervallo di tempo in secondi dall'inizio della simulazione in cui è effettuata la misura
3. ID della cella
4. ID unico dell'Ue
5. ID dell'Ue rispetto alla cella
6. ID del canale logico
7. Numero di PDU trasmessi dal livello PDCP
8. Byte totali trasmessi
9. Numero di PDU ricevuti dal livello PDCP
10. Byte totali ricevuti
11. Delay medio in secondi
12. Deviazione standard del delay
13. Delay massimo
14. Delay minimo
15. Dimensione media del PDU in byte
16. Deviazione standard della dimansione del PDU
17. Dimensione massima
18. Dimensione minima

4.2 Simulazione 1: QoS totale del sistema

4.2.1 Configurazione e parametri

Questa prima simulazione (*sim1*) si propone di andare a valutare *throughput* e *delay* al crescere della distanza degli Ue dalla stazione base. Al contempo verificheremo sia la fattibilità del nostro particolare approccio sia saremo in grado di determinarne le prestazioni. Il *throughput*, valutato in questo caso come rapporto fra byte ricevuti su byte trasmessi (normalizzato quindi), è un buon indicatore della QoS del sistema. Sarà importante inoltre tenere sotto controllo il *delay*, qui misurato come media dei *delay* dei singoli pacchetti (*mean delay*), per verificare che non raggiunga valori eccessivi. *Sim1* comprende tre varianti (nel seguito indicate con *var1, var2, var3*) caratterizzate da:

- numero di UE fisso e pari a 23;
- pesi in byte attribuiti ai singoli pacchetti diverso da una variante all'altra (ma fisso nel corso della singola simulazione);

Il senso di questa variazione nella dimensioni risulta chiaro in riferimento ai layer di SVC. Non tutti i pacchetti hanno pari importanza ma ad esempio gli A sono prioritari, i B intermedi, i C son un "optional" che garantisce HQ. Ha senso allora cercar di capire come varia la QoS globale attribuendo loro pesi differenti.

- *Riassunto parametri sim1:*
  - *metriche: thrughput e* delay *al crescere della distanza*
  - *numero Ue: N=23 fisso*
  - *distanza da EnB: 23 misure nell'intervallo [0 , 10400]* metri
  - *dimensioni pacchetti:*
    var1: dimA=dimB=dimC=500 byte
    var2: dimA=800, dimB=400, dimC=300 byte
    var3: dimA=300, dimB=500, dimC=700 byte
  - *tot pacchetti inviati: 3x2000*

Osserviamo infine che il numero totale di byte trasmessi nel corso di ogni variante non cambia. Cambia solo la loro ripartizione.

4.2.2 Analisi/interpretazione dei risultati

Il grafico di *Figura 4.1* illustra l'andamento del *throughput* normalizzato in funzione della distanza dell'utente dalla stazione base. Ogni punto evidenziato nel grafico corrisponde infatti ad un Ue nella simulazione. Poiché il numero totale di byte trasmessi in ogni variante di sim1 non cambia, se avessimo rappresentato il numero totale di byte ricevuti da ciascun Ue (senza normalizzarli) l'andamento qualitativo sarebbe rimasto lo stesso. Sono evidenti tre zone distinte, delimitate da strette regioni di transizione in cui le varie curve compio salti piuttosto bruschi. Chiamiamole rispettivamente zonaC (da 0 a 2000 metri circa) , zonaB(0, 6000 m .ca), zonaA(0,10000 m .ca). La prima è chiaramente quella coperta dalla 64-QAM, la seconda dalla 16-QAM, la terza dalla QPSK. Se immaginiamo la QoS del sistema come area sottesa dalla curva del *throughput*, risulta chiaro che attribuire maggior peso ai pacchetti A come in *var2* porta ad un aumento della qualità totale. emphVar1 è invece una situazione intermedia , con pesi bilanciati, infine *var3* da questo punto di vista è la peggiore fra quelle considerate. Oltre i 10000 metri infine le curve calano bruscamente: il canale in è più in condizioni idonee alla trasmissione. Distinguiamo le stesse tre regioni anche per quanto concerne il *delay*(*Figura 4.2* - in cui l'andamento di var3 non è stato riportato poiché andava a sovrapporsi quasi totalmante agli altri due). Le differenze però non sono così marcate fra una variante e l'altra. Questo ci garantisce, fra le tre, di poter scegliere la migliore dal punto di vista della QoS. *Figura 4.3* infine è semplicemente un particolare della precedente. Abbiamo notato, nel corso delle simulazioni, che, mentre il *delay* per Ue vicini a EnB dipende principalmente dalle dimensioni dei pacchetti (e quindi dal tempo di trasmissione), per utenti lontani il contributo principale è quello di coda/eventuale riordinamento. Nella simulazione 2 andremo a valutare l'entità del primo di questi due fenomeni.

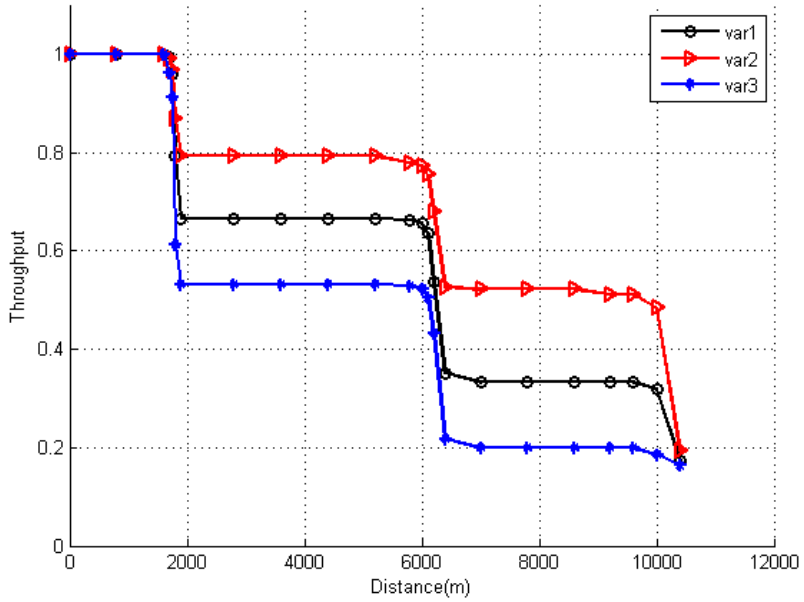

Figura 4.1: Throughput

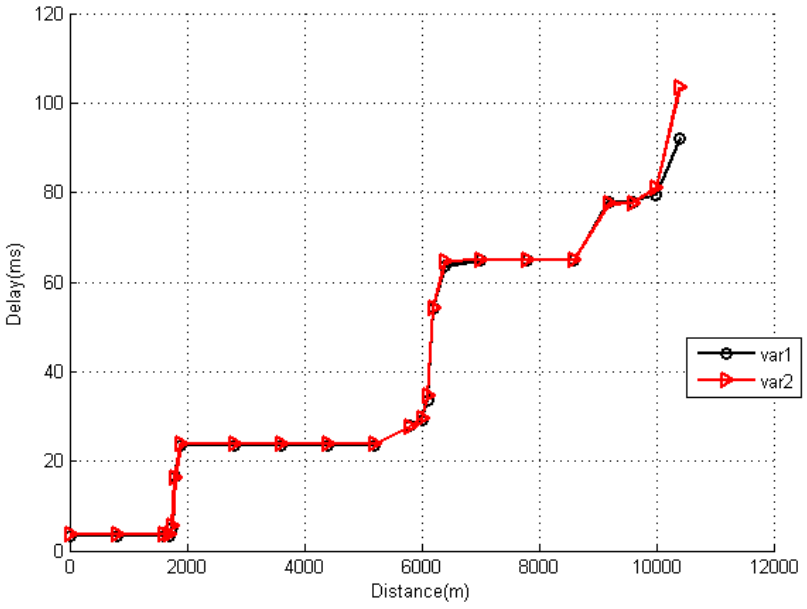

Figura 4.2: Delay

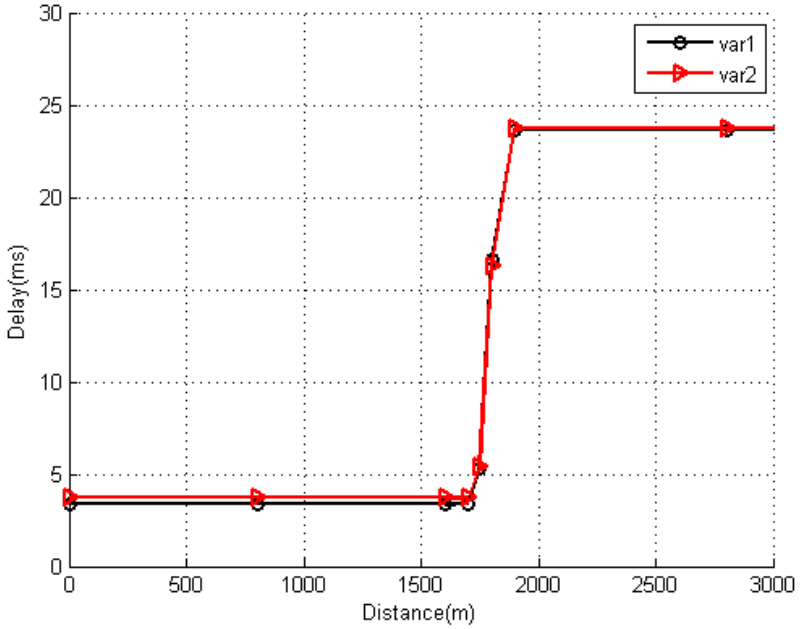

Figura 4.3: Particolare di Figura 4.2

4.3 Simulazione 2: Ue che godono di buon SNR

4.3.1 Configurazione e parametri

Nella simulazione 1 abbiamo preso in esame la QoS totale del sistema, individuando gli accorgimenti che consentono di incrementarla globalmente. In questa seconda simulazione (d'ora in poi *sim2*) scegliamo invece un approccio complementare, andando ad analizzare prestazioni di Ue che godono di buon SNR. Consideriamo un intervallo di distanze $D = [0, d]$, sia inoltre x_i la distanza del generico Ue da EnB. Notiamo da sim1 che, se $d < 2000$ m, qualsiasi Ue posto a distanza $x_i \in D$ godrà di *throughput* circa unitario (ossia esperirà la massima QoS disponibile). Scegliamo allora, per fissare le idee, $x_i = 800$ m. Come abbiamo già avuto modo di osservare, per utenti relativamente vicini a EnB il *delay* non è particolarmente influenzato da eventuali tempi di coda/riordinamento fra RLC e PDCP, ma dipende invece prevalentemente dalle dimensioni dei pacchetti. In *sim2* andremo allora a valutare il *delay* relativo ad un unico utente, posto appunto a distanza $x_i = 800$ m, al variare delle dimensioni dei pacchetti inviati, cercando di fornire una mappatura esaustiva. Per farlo riprogrammiamo la nostra applicazione in

modo che si comporti nel seguente modo: generi il solito pattern A,B,C con periodo $T = 150$ ms ma, mantenendo fissa la dimensione dei pacchetti A (cioè fissando la qualità video base), incrementiamo ad ogni periodo la dimensione dei pacchetti B e C (cioè supponendo di incrementare la qualità degli altri due sub-stream). Le regole di incremento sono:

$$dimB_{j*T+1} = (dimB_{j*T} + 100)\text{byte} \tag{4.1}$$

$$dimC_{j*T+1} = (dimC_{j*T} + 200)\text{byte} \tag{4.2}$$

$$j = 1, 2, ..., 19 \tag{4.3}$$

- *Riassunto parametri sim2:*
  - *metrica: delay al variare delle dimensioni dei pacchetti*
  - *numero Ue: N=1*
  - *distanza da EnB: 800* m
  - *dimensione pacchetti A: fissa a 500* byte
  - *dimensione pacchetti B: da 200 a 2000* byte*, incremento 100* byte/periodo
  - *dimensione pacchetti C: da 400 a 4000* byte*, incremento 200* byte/periodo
  - *tot pacchetti inviati: 3x19*

Precisiamo che solitamente vari layer dello stack ISO-OSI tendono a partizionare pacchetti di dimensioni troppo elevate (dimensioni tipiche sono dell'ordine dei 1500 byte). Noto tutto ciò, in *sim2* forziamo i pacchetti C a raggiungere dimensioni fino a 4000 byte. Tutto questo poiché l'andamento del *delay* in funzione delle dimensioni risulti il più chiaro possibile. Vedremo come tale andamento resti valido, in generale, qualsiasi sia la modulazione usata e, in particolare, con pacchetti di dimensioni minori associati a modulazioni più basse.

4.3.2 Analisi/interpretazione dei risultati

il grafico di *Figura 4.4* mostra l'andamento dei ritardi con l'avanzare del tempo di *sim2*. Più precisamente siamo andati a "campionare" i ritardi di ciascun tipo di pacchetto in ogni periodo T. Sappiamo che tale periodo ha durata di 150 ms. In figura per semplicità si è supposto $T = 1$ s. Il *delay* dei pacchetti a, dopo un primo istante di assestamento, si stabilizza ad un valore costante di 4 ms .ca (com'era logico aspettarsi visto che dimA non varia). Gli altri due ritardi mostrano un andamento a "gradini". La linea tratteggiata è invece la media fra i tre.

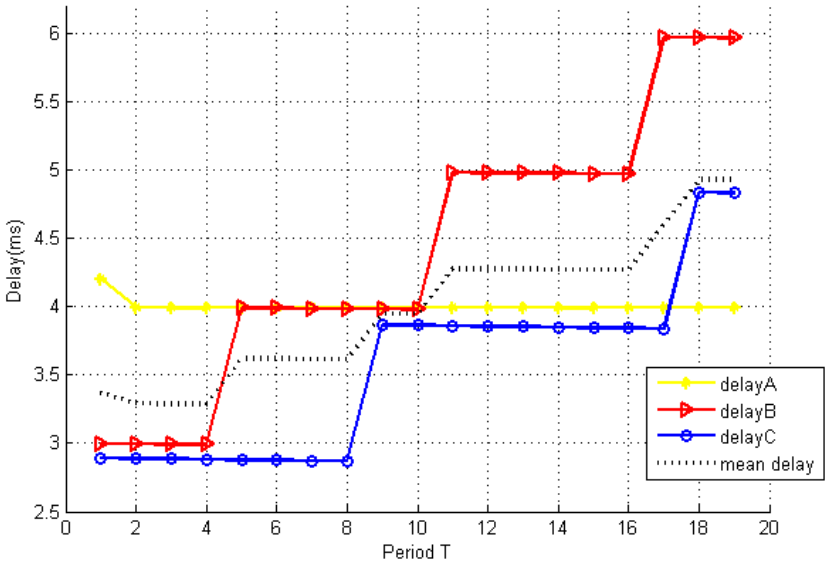

Figura 4.4: Andamento del *delay* per i vari tipi di pacchetto in sim1

L'andamento a gradini per per i ritardi di B e C, ancor più evidente nella *Figura 4.5*, è facilmente interpretabile. Notiamo infatti che, a meno di una piccola discrepanza dovuta ai byte di *header* aggiunti dai livelli RLC e MAC, il "passo del gradino" corrisponde con le dimensioni del TB associato a ciascuna modulazione (*cfr.* estratto seguente, dove la modulazione è indicata in colonna 7, le dimensioni (in byte) del corrispondente TB in colonna 8 [21]).

```
MAC Downlink
1.052 1 1 106 3 1 6 309 0 0
1.101 1 1 111 2 1 12 597 0 0
1.154 1 1 116 5 1 26 1836  0 0
```

In altre parole, ogni qual volta sia necessario un ulteriore TB, il *delay* aumenta bruscamente per poi mantenersi costante lungo un determinato intervallo di byte. Tutto ciò fornisce allora un criterio per la scelta ottimale delle dimensioni dei pacchetti, senza sforare requisiti sul tempo di trasmissione.

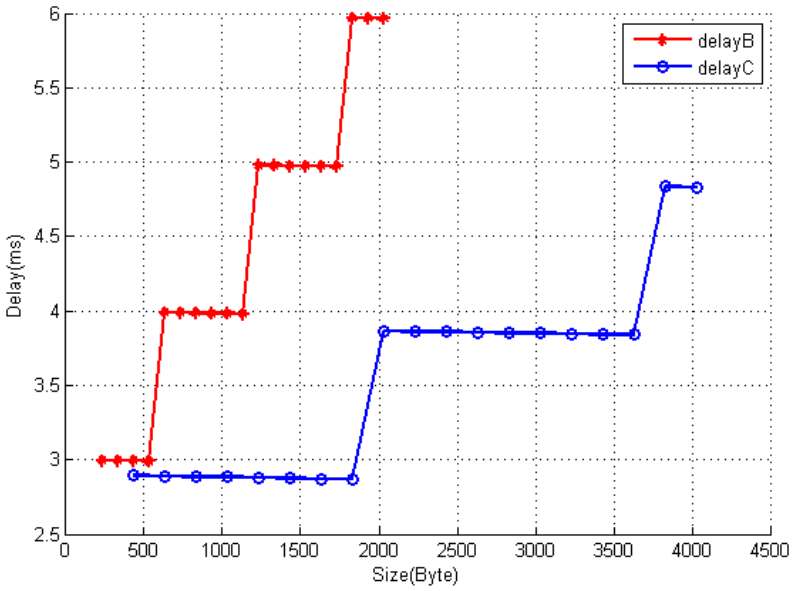

Figura 4.5: Andamento del *delay* al crescere delle dimensioni pacchetti di B e C

Capitolo 5

Conclusioni ed eventuali sviluppi futuri

5.1 Conseguenze pratiche

In questa tesi siamo andati a valutare le prestazioni di un approccio allo streaming di traffico video broadcast alternativo a quello classico. Tale approccio, reso possibile dalle proprietà di scalabilità del traffico video stesso e dalle potenzialità messe a disposizione dalla tecnologia LTE, si è rivelato valido sia nell'incrementare la QoS totale del sistema, sia nel caso in cui si scelga di privilegiare utenti con buone condizioni del canale. I parametri che possono andare ad influenzare le prestazioni di tale approccio sono: (i)scelta delle modulazioni da associare a ciascun tipo di pacchetto, in funzione della sua importanza; (ii) dimensioni (sia relative sia assolute) dei pacchetti; (iii) distanza degli Ue dalla stazione base (associata alla qualità del canale). Un risultato che tuttavia ci ha lasciati un po' sorpresi è quello relativo ai *delay* di Figura 4.2. in quelle che abbiamo definito regioni A e B infatti tali ritardi tendono a raggiungere valori molto superiori rispetto a quelli di zona C e risultano inoltre dipendenti non tanto dal tempo effettivo di propagazione attraverso il canale radio, quanto da quello di coda/riordinamento.

5.2 Integrazioni e sviluppi

5.2.1 Fading traces

Il *fading model* è uno strumento messo a disposizione da LENA [21] per modellare nella maniera più accurata possibile il canale di trasmissione. Si parla di *fading* in riferimento allo scostamento nell'attenuazione del segnale radio durante la sua propagazione in un mezzo (`http://en.wikipedia.org/wiki/Fading`). Tale fenomeno può essere ad esempio dovuto a percorsi multipli (*multipath*) o a quello che vine definito *shadowing*, a sua volta dovuto ad ostacoli lungo il percorso di propagazione. Il *fading model* implementato nel simulatore é basato su tracce precalcolate ([21]). Purtroppo richiede grandi risorse computazionali e nelle nostre simulazioni non è stato usato. Sarebbe tuttavia interessante, al fine di ottenere risultati numerici più realistici, sfruttare questa opzione, magari in future simulazioni.

5.2.2 Celle multiple

Un possibile sviluppo futuro riguarda lo studio di scenari in cui coesistano più EnB (vedi anche [1]), detti anche scenari a celle multiple. Secondo quanto specificato nel progetto MBMS del 3GPP [3] un singolo utente dovrebbe poter essere in grado di ricevere lo stesso pacchetto broadcast inviato da più celle fra loro opportunamente sincronizzate. Si parla a tal proposito di MBSFN, *Multicast/Broadcast over Single Frequency Network.* Un semplice accorgimento è quello di dilatare a sufficienza il *prefix cycle* per garantire ricezione di RB provenienti da celle distanti senza interferenze. Anche in caso di ricezione di pacchetti parzialmente corrotti, l'utente risulterebbe in grado di correggere un maggior numero di errori sfruttando la combinazione di informazioni provenienti da celle diverse. Si tratta com'è intuibile di una configurazione non semplice da modellare in un simulatore. Qualora venisse effettivamente implementato in LENA, sarebbe interessante valutare nuovamente le prestazioni del nostro sistema per verificare l'effettiva efficacia di MBSFN.

Appendice A

Parte del codice delle simulazioni

Riportiamo di seguito le più importanti modifiche apportate alla classe *MyApp* dell'esempio fifth.cc del tutorial di ns-3 [19], usata come base per realizzare la nostra applicazione.

```
void MyApp::SendPacket_A (void)
{
  Ptr<Packet> packet = Create<Packet> (m_sizeA);
  for(int i = 1; i<=1;i++)
    m_socket->Send (packet);
  m_cqi = 7;
  m_sizeLittle = m_sizeA;
  m_sendEvent = Simulator::Schedule (Seconds(0.05), &MyApp::SendLittle, this);
  m_sendEvent = Simulator::Schedule (Seconds(0.1), &MyApp::SendPacket_B, this)
    if (++m_packetsSent < m_nPackets)
    {
      ScheduleTx ();
    }
}
void MyApp::SendPacket_B(void)
{
    Ptr<Packet> packet = Create<Packet>(m_sizeB);
    for(int i = 1; i<=1;i++)
        m_socket->Send (packet);
    m_cqi=14;
```

```
    m_sizeLittle = m_sizeB;
    m_sendEvent = Simulator::Schedule (Seconds(0.05), &MyApp::SendLittle, this);
    m_sendEvent = Simulator::Schedule (Seconds(0.1), &MyApp::SendPacket_C, this);
}
void MyApp::SendPacket_C(void)
{
    Ptr<Packet> packet = Create<Packet>(m_sizeC);
    for(int i = 1; i<=1;i++)
        m_socket->Send (packet);
    m_cqi=4;
    m_sizeLittle = m_sizeC;
    m_sendEvent = Simulator::Schedule (Seconds(0.05), &MyApp::SendLittle, this);
}
void MyApp::SendLittle (void)
{
  LteAmc::DYNAMIC_CQI = m_cqi;
  Ptr<Packet> packet = Create<Packet> (m_sizeLittle);
  m_socket->Send (packet);
}
void MyApp::ScheduleTx (void)
{
  if (m_running)
    {
      Time tNext (Seconds (0.3));
      m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket_A, this);
    }
}
```

Riportiamo anche le modifiche apportate al file .src\lte\model\lte-amc.cc per consentire la scelta dinamica delle modulazioni (tale file fa parte del codice sorgente del simulatore così com'è scaricabile da [17]):

```
...
294   //variabile statica non presente nel file sorgente originale.
295    int LteAmc::DYNAMIC_CQI = 4;
...
395       for (uint8_t j = 0; j < rbgSize; j++)
396              {
397                 //originariamente: cqi.push_back (rbgCqi);
398                 cqi.push_back (DYNAMIC_CQI);
399              }
...
```

Bibliografia

[1] D. Munaretto, D. Jurca, J. Widmer, *"Scalable Video Broadcast in Cellular Networks: Impact on QoS and Network Resources"*,
DOCOMO Communications Laboratories GmbH Munich, Germany

[2] "*Cisco Visual Networking Index: Forecast and Methodology, 2011-2016*",
`http://www.cisco.com/`

[3] A. Alexiou, C. Bouras, V. Kokkinos, A. Papazois, G. Tsichritzis, *"Multimedia Broadcasting in LTE Networks"*, Research Academic Computer Technology Institute, Computer Engineering and Informatics Department University of Patras, Greece

[4] M. Kottkamp, A. Rössler, J. Schlienz, J. Schütz , *"LTE Release 9 Technology Introduction"*, White Paper, December 2011

[5] A. Rao, Y. Lim, C.i Barakat, A. Legout, D. Towsley, W. Dabbous, *"Network Characteristics of Video Streaming Traffic"*, Author Manuscript, published in ACM CoNEXT (2011)

[6] H. Schwarz, D. Marpe, T. Wiegand, *"Overview of the Scalable Video Coding Extension of the H.264/AVC Standard"*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, NO. 9, September 2007

[7] T. Schierl, T. Stockhammer, T. Wiegand, *"Mobile Video Transmission Using Scalable Video Coding"*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, NO. 9, September 2007

[8] J. v.d. Merwe, S. Sen, C. Kalmanek,
"Streaming Video Traffic: Characterization and Network Impact",
AT&T Labs Research

[9] N. Benvenuto, M. Zorzi,
"Principles of Communications Networks and Systems", (book)
Wiley 2011

[10] D. Astèly, E. Dahlman, A. Furuskär, Y. Jading, M. Lindström, S. Parkvall, *"LTE: The Evolution of Mobile Broadband"*, IEEE Communications Magazine, April 2009

[11] Freescale Semiconductor, *"Long Term Evolution Protocol Overview"*, White Paper, Rev 0 10/2008

[12] Motorola, *"Long Term Evolution (LTE): Overview of LTE Air-Interface",* Technical White Paper, Motorola, Inc. 2007

[13] P. Fuller , *"LTE eNodeB MAC Scheduler Introduction"*, Roke

[14] N. Baldo, M. Miozzo, M. Requena-Esteso, J. Nin-Guerrero, *"An Open Source Product-Oriented LTE Network Simulator based on ns-3"*, Centre Tecnolgic de Telecomunicacions de Catalunya (CTTC)

[15] ____ , *"A new model for the simulation of the LTE-EPC data plane"*,Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)

[16] "*Ubiquisys and CTTC Develop World's First Open Source Product-Oriented LTE Network Simulator"*,
`http://www.ubiquisys.com/femtocell-media-press-releases-id-203.htm`,
femtocell-media-press-releases-id-203.htm, Feb. 2011.

[17] "*The LTE-EPC Network Simulator (LENA) project"*,
`http://iptechwiki.cttc.es/LTE-EPC_Network_Simulator_(LENA)`,
LTE-EPC Network Simulator (LENA).

[18] "*ns-3 official site"*,
`http://www.nsnam.org/`,

[19] "*ns-3 tutorial"*,
`http://www.nsnam.org/docs/release/3.13/tutorial/html/index.html`

[20] "*ns-3 manual*",
`http://www.nsnam.org/docs/manual/html/index.html`

[21] "*ns-3 LTE module documentation*",
`http://lena.cttc.es/manual/`,