Università degli Studi di Padova

Analisi e simulazione di reti subacquee multimodali adattive

Laurea Magistrale in Ingegneria delle Telecomunicazioni

Laureando:
Riccardo Casagrande

Relatore:
Prof. Michele Zorzi

Signet
Dipartimento di Ingegneria dell'Informazione

Anno Accademico 2017/2018

UNIVERSITÀ DEGLI STUDI DI PADOVA

Abstract

Scuola di Ingegneria
Dipartimento di Ingegneria dell'Informazione

Laurea Magistrale in Ingegneria delle Telecomunicazioni

Analisi e simulazione di reti subacquee multimodali adattive

Riccardo Casagrande

Negli ultimi anni l'interesse per le comunicazioni subacquee è aumentato notevolmente sia per fini militari come il controllo marittimo e la difesa, che per scopi civili come la ricerca scientifica, l'estrazione di combustibili fossili e numerosi altri. In questa tesi viene esteso il simulatore di reti sottomarine DESERT, basato su NS-MIRACLE, introducendo la possibilità di fare simulazioni con nodi multimodali e adattivi, inoltre è stato sviluppato un nuovo modulo che permette di utilizzare un jammer in DESERT. Nella seconda parte della tesi vengono analizzate e confrontate diverse reti, mettendo in risalto soprattutto la differenza in termini di Throughput e Packet Delivery Ratio tra i nodi multimodali adattivi rispetto ai classici nodi statici. Multimodalità e adattività rappresentano un passo importante per l'ottimizzazione delle comunicazioni subacquee sia in termini di velocità che di robustezza nelle trasmissioni.

Riconoscimenti

Ringrazio infinitamente i miei genitori per aver sempre supportato le mie scelte.

Ringrazio Sara per avermi aiutato, sostenuto e sopportato anche nei momenti più difficili.

I miei ringraziamenti vanno anche a tutti i membri del laboratorio Signet (parte DEI/G), in particolare Filippo e Federico che mi hanno seguito per tutta la tesi.

Indice

Elenco delle figure

Elenco delle tabelle

Abbreviazioni

8PSK	**8** **P**hase-**S**hift **K**eying
BER	**B**it **E**rror **Rate**
BPSK	**B**inary **P**hase **S**hift **K**eying
CBR	**C**onstant **B**it **R**ate
ChSAP	**Ch**annel **S**ervice **A**ccess **P**oint
ClSAP	**C**ross-**l**ayer **S**ervice **A**ccess **P**oint
CMRE	**C**entre for **M**aritime **R**esearch and **E**xperimentation
CSMA	**C**arrier **S**ense **M**ultiple **A**ccess
DESERT	**DE**sign, **S**imulate, **E**mulate and **R**ealize **T**est-beds for underwater network protocols
HF	**H**igh **F**requency
LF	**L**ow **F**requency
LUT	**L**ook **U**p **T**able
MAC	**M**edia **A**ccess **C**ontrol
MFSK	**M**ultiple **F**requency-**S**hift **K**eying
NATO	**N**orth **A**tlantic **T**reaty **O**rganization
NS	**N**etwork **S**imulator
NS-MIRACLE	**M**ulti- **I**nter**R**f**A**ce **C**ross-**L**ayer **E**xtension library for the **N**etwork **S**imulator
OSI	**O**pen **S**ystem **I**nterconnection model
PER	**P**acket **E**rror **Rate**
PSD	**P**ower **S**pectral **D**ensity
QoS	**Q**uality **o**f **S**ervice
QPSK	**Q**uadrature **P**hase-**S**hift **K**eying

RMS	**R**oot **M**ean **S**quare
ROV	**R**emotely **O**perated underwater **V**ehicle
SAP	**S**ervice **A**ccess **P**oint
SIR	**S**ignal to **I**interference **R**atio
SNR	**S**ignal to **N**oise **R**atio
SWiG	**S**ubsea **Wi**reless **G**roup
TCP	**T**ransmission **C**ontrol **P**rotocol
WOS	**W**orld **O**cean **S**imulation **S**ystem

Capitolo 1

Introduzione

Negli ultimi anni le comunicazioni subacquee sono divenute oggetto di crescente interesse, sia nella comunità scientifica che tra aziende di diversi settori. Le trasmissioni acustiche sottomarine si sono diffuse in ambito civile creando o migliorando strumenti utili in vari campi quali la navigazione, la ricerca di particolari specie marine, le immersioni, la telemetria e l'estrazione di combustibili fossili come il petrolio. Il primo ad osservare che il suono si propaghi in modo differente sott'acqua e a provare ad utilizzare le caratteristiche delle onde sonore subacquee fu Leonardo Da Vinci nel XV secolo che utilizzando un lungo tubo creò il primo rudimentale esempio di *sonar passivo*[1]. Con il passare del tempo varie tecnologie sono state inventate e, ad oggi, è possibile trasmettere informazione sott'acqua utilizzando onde sonore, segnali magneto-induttivi, onde ottiche e frequenze radio. L'obiettivo dei ricercatori in questo ambito è migliorare le tecnologie elencate incrementando la robustezza delle comunicazioni, la bitrate e le distanze raggiungibili. Uno degli step fondamentali per raggiungere questi traguardi consiste nel simulare reti subacquee per confrontare e testare nuovi dispositivi e protocolli prima di passare ad una valutazione reale sul campo. In quest'ottica si pone il progetto DESERT Underwater (*DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols*) sviluppato all'Università degli Studi di Padova e rilasciato per la prima volta nel 2012 [2]. DESERT è un insieme di librerie C++ pubbliche che estendono il simulatore NS-MIRACLE [3] per supportare il design e l'implementazione di protocolli di rete per trasmissioni subacquee. Il primo scopo di questa tesi è quello di estendere le funzionalità di DESERT introducendo la

possibilità di effettuare simulazioni che sfruttino l'adattività e la multimodalità. L'adattività è la possibilità di cambiare i parametri di ricezione e trasmissione dei modem in base alle condizioni del canale o alla QoS (*Quality of Service*) richiesta, la multimodalità è l'utilizzo di diverse tecnologie a livello fisico all'interno di un unico dispositivo. Nel capitolo 5 vengono analizzati e illustrati i miglioramenti ottenibili con queste tecniche mettendo a confronto in diversi scenari reti adattive e multimodali con altre che non dispongono di queste tecnologie. La seconda parte della tesi consiste nell'implementazione di un jammer (o disturbatore di frequenze) in DESERT e nell'utilizzo di questo dispositivo in simulazioni subacquee come fonte di rumore. Questa tesi è così strutturata: nelle sezioni 1.1, 1.2 e 1.3 vengono definiti i concetti di multimodalità, adattività e jammer rispettivamente, sottolineando il motivo della loro importanza in ambiente subacqueo, il capitolo 2 fornisce una panoramica della fisica delle trasmissioni subacquee, il capitolo 3 descrive i simulatori NS, NS-MIRACLE e DESERT, le sezioni 4.1, 4.2 e 4.3 spiegano come sono stati implementati adattività, multimodalità e jammer in DESERT rispettivamente. Il capitolo 5 fornisce i risultati di quattro diverse simulazioni, analizzate e confrontate tra loro. Infine il capitolo 6 contiene le conclusioni della tesi.

1.1 Multimodalità

La recente diffusione di diversi sistemi di trasmissione subacquea (onde ottiche, segnali elettromagnetici, onde acustiche) ha portato alla creazione di dispositivi con varie interfacce a livello fisico e quindi con la possibilità di trasmettere in parallelo contemporaneamente diversi flussi di dati con differenti tecnologie. Apparecchiature di questo tipo sono dette multimodali. In [6] sono illustrate e comparate le caratteristiche di molti dispositivi che utilizzano diversi sistemi di trasmissione. La multimodalità riguarda anche la banda utilizzata, ovvero un dispositivo può utilizzare un'unica tecnologia ma inviare dati su canali paralleli semplicemente utilizzando bande diverse che non interferiscono. In questa tesi abbiamo implementato e analizzato dispositivi che possono utilizzare due livelli fisici diversi denominati HF (*High Frequency*) e LF (*Low Frequency*), entrambi basati su onde sonore ma su bande differenti. La mulitmodalità, in aggiunta all'adattività, è una tecnica fondamentale per migliorare le comunicazioni sottomarine e renderle maggiormente flessibili in base alle necessità dell'utente e alle condizioni del

canale. Per sottolineare questo aspetto è interessante osservare la simulazione effettuata in [6] e riportata in figura 4.1.

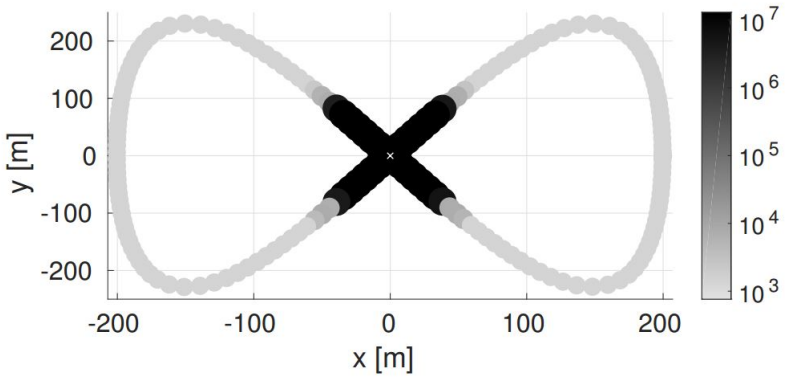


FIGURA 1.1: Throughput del ROV in un sistema multimodale rispetto alla posizione [6].

Nello scenario analizzato nell'articolo si fa muovere un *Remotely Operated Vehicle* (ROV), cioè un veicolo mobile comandato a distanza, su una traiettoria chiusa a forma di 8 e si utilizza il modem (scelto tra due di tipo acustico e uno ottico) che consente la bitrate migliore in base alla distanza dalla base di controllo, posizionata al centro della figura e identificata da una x bianca. Il colore dei cerchi indica il livello del throughput in base alla scala posta sulla destra. Come si può notare dalla figura, la multimodalità consente di utilizzare trasmissioni più robuste o più veloci in base alla distanza da coprire. L'implementazione della multimodalità in DESERT verrà descritta nella sezione 4.1.

1.2 Adattività

I motivi che rendono fondamentale l'adattività nelle reti subacquee sono principalmente due:

- Miglioramento delle performance della rete.
- Possibilità di connettere dispositivi eterogenei tra loro.

In primo luogo la possibilità di poter ridefinire in ogni istante i parametri della rete aumenta notevolemente le prestazioni raggiungibili, infatti si provi a pensare a reti con nodi che necessitino di utilizzare modulazioni, potenze e bitrate diverse in base alla distanza che si vuole

raggiungere ma anche rispetto alle condizioni del mare (temperatura, salinità, velocità del vento e delle onde). Un altro aspetto rilevante è la robustezza delle comunicazioni, in certi scenari è preferibile abbassare la bitrate per ottenere trasmissioni più affidabili, utilizzando l'adattività è possibile cambiare la potenza di trasmissione e la modulazione per raggiungere questo scopo. Un esempio che verrà analizzato nei capitoli seguenti è l'utilizzo di un preambolo che contenga le informazioni necessarie a connettere due nodi. Questa componente del messaggio è molto piccola (2 Byte nelle simulazioni) ma di notevole importanza, quindi è preferibile utilizzare una modulazione più resistente agli errori (es: BPSK). Il secondo motivo per cui l'adattività ricopre un ruolo fondamentale nelle trasmissioni sottomarine è la necessità di connettere dispositivi diversi, creati da aziende differenti con interfacce e protocolli che variano da azienda ad azienda. Inizialmente le reti subacquee erano gestite da un'unica compagnia che quindi non aveva problemi nel collegare i propri dispositivi, ad oggi la diversità delle apparecchiature utilizzate e il diffondersi di nuove tecnologie nel mercato ha reso necessaria la definizione di uno standard comune. Di questa problematica si è occupato il gruppo di ricerca SWiG (*Subsea Wireless Group*) fondato nel 2011 e formato da studiosi provenienti dalla rete internazionale dell'industria petrolifera e del gas comprendente operatori, installatori e aziende tecnologiche [4]. Dallo studio effettuato è emerso che il protocollo migliore da inserire nello standard delle comunicazioni subacquee fosse JANUS, un protocollo precedentemente creato dal CMRE (*Centre for Maritime Research and Experimentation*) della NATO [5] e diventato standard nel 2017. JANUS è stato pensato come un'interfaccia che permetta di connettere dispositivi diversi ma che non li vincoli a comunicare utilizzando esclusivamente quel protocollo, al contrario JANUS consente di effettuare l'*handshake* tra i due modem che in seguito hanno la possibilità di accordarsi per utilizzare protocolli di comunicazione più adatti alle caratteristiche dei dispotivi e alle condizioni del canale. In quest'ottica l'adattività è fondamentale per avere la possibilità di cambiare metodi di trasmissione. In figura 1.2 è illustrato il diagramma a blocchi della codifica di JANUS. L'implementazione dell'adattività in DESERT sarà descritta nella sezione 4.2.

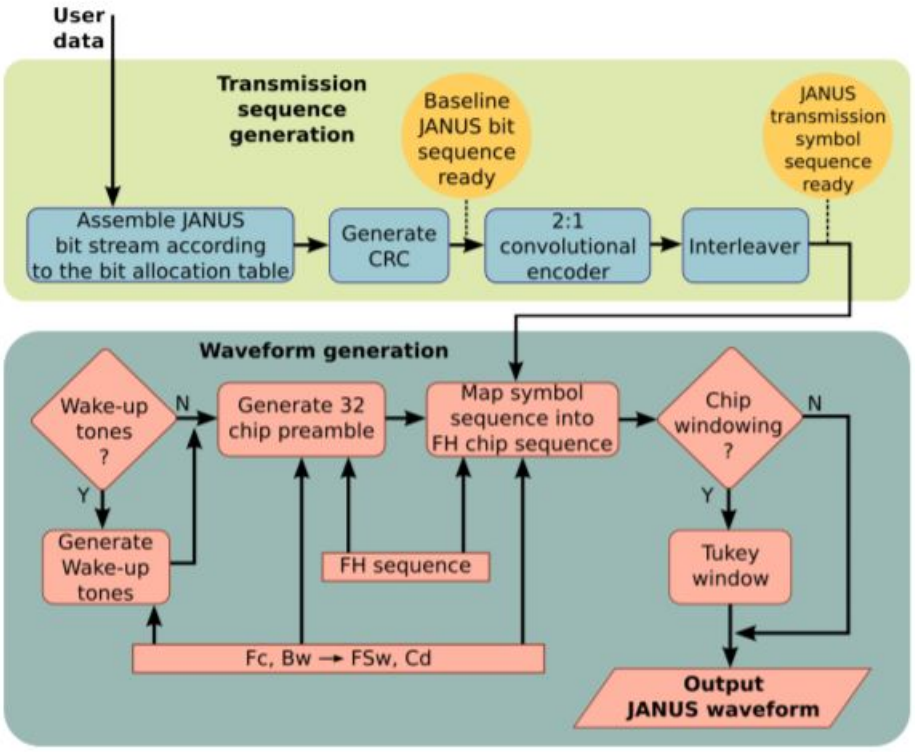


FIGURA 1.2: Diagramma a blocchi della codifica di JANUS [5].

1.3 Jammer

La seconda parte della tesi consiste nell'implementazione di un jammer utilizzato poi all'interno di alcune simulazioni. L'obiettivo di questo dispositivo è creare interferenza nel canale impedendo ad altri nodi di comunicare tra loro. La logica alla base è molto semplice, il jammer invia continuamente in broadcast pacchetti privi di informazione sulla banda selezionata, impedendo la corretta ricezione dei messaggi da parte degli altri dispositivi. Nelle simulazioni di reti subacquee, un apparecchio di questo tipo ha una doppia funzionalità, in primo luogo può essere utilizzato per simulare l'effettiva presenza di un disturbatore di frequenze, in secondo luogo può essere impiegato come fonte di rumore. In particolare è stato mostrato in [7] che l'interferenza generata da una nave sopra ad un'area coperta da sonar, agisce come un disturbatore di frequenze e quindi si può simulare accuratamente parametrizzando correttamente un jammer in DESERT. Nella sezione 4.3 viene descritta l'implementazione del jammer.

Capitolo 2

Canale acustico sottomarino

In questo capitolo viene data una panoramica generale sulle trasmissioni acustiche subacquee, per una trattazione più completa si rimanda a [1] e [8]. La propagazione delle onde sonore in acqua è molto diversa rispetto a quella delle onde radio, la prima differenza consiste nella velocità di propagazione del suono che in acqua è di circa 1500 m/s. In realtà 1500 m/s è una stima approssimata del valore medio, la reale velocità dell'onda dipende da diversi fattori quali pressione, temperatura e livello di salinità, inoltre varia al cambiare della profondità nella colonna d'acqua. Una caratteristica fondamentale delle onde sonore è che l'attenuazione è fortemente legata alla frequenza infatti a basse frequenze i segnali subiscono attenuazioni minori e possono raggiungere distanze più elevate [1]. Nelle simulazioni effettuate verrà analizzato questo fenomeno che è uno dei principali motivi per il quale la multimodalità è fondamentale in una rete subacquea. Un fattore importante è anche la profondità del mare infatti, in acque basse, i fronti d'onda si propagano in forma cilindrica e non sferica come invece accade in acque più profonde. Oltre all'attenuazione, anche il rumore dipende dalla frequenza ed è quindi colorato, altri effetti che incidono sulle trasmissioni subacquee sono le correnti che creano sfasamenti in frequenza dovuti all'effetto Doppler e la forma del fondale che crea complicati fenomeni di multi-path.

2.1 Modello di canale

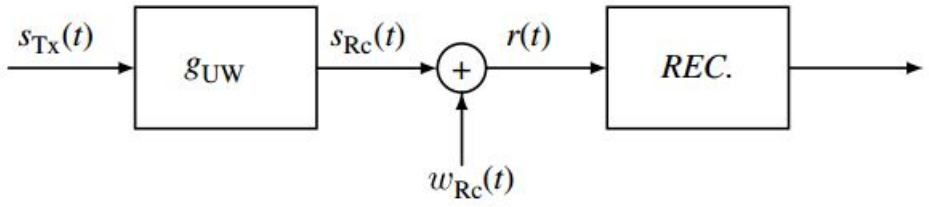


FIGURA 2.1: Modello di canale subacqueo [1].

Il modello di canale considerato è mostrato in figura 2.1. Nelle trasmissioni subacquee l'intensità di un segnale trasmesso viene definita da una pressione misurata in μPa. La potenza considerata non è quindi quella elettrica, l'equazione 2.1 presa da [7] mette in relazione la potenza elettrica con la pressione. L'unità di misura della potenza di un segnale subacqueo è dB re μPa @ 1 m cioè dB riferiti a 1 μPa misurati ad 1 m dalla sorgente.

$$10 \log \frac{P}{1W} = 20 \log \frac{P_{RMS}}{1\mu Pa} - 170.8 \tag{2.1}$$

Dove P_{RMS} è il *Root Mean Square* del segnale trasmesso. Nel simulatore DESERT il parametro da inserire per definire l'intensità del segnale è quello della pressione, per questo motivo l'equazione 2.1 è importante per mettere in relazione le potenze elettriche nominali date dai costruttori dei modem con l'effettivo parametro da utilizzare nelle simulazioni.

2.1.1 Attenuazione e frequenza

L'equazione 2.2 mostra la relazione tra attenuazione del canale a_{Ch} e la frequenza del segnale trasmesso f_0, il fattore d^k è detto *spreading loss* e dipende dalla geometria della propagazione mentre il termine $[\alpha(f_0)]^d$ è detto *absorption loss* e modella la perdita di energia dovuta alla trasformazione della pressione in calore. Il valore d rappresenta la distanza tra trasmettitore e ricevitore mentre il coefficiente k definisce la geometria della propagazione (cilindrica, sferica o mista) [1] [9].

$$a_{Ch}(d, f_0) = d^k \cdot [\alpha(f_0)]^d \tag{2.2}$$

In figura 2.2 è mostrato l'andamento dell'*absorption loss* al variare della frequenza, come si può notare la dipendenza è molto forte, per questo è importante utilizzare modem multimodali che permettano di scegliere la frequenza migliore in base alla distanza che il segnale dovrà percorrere.

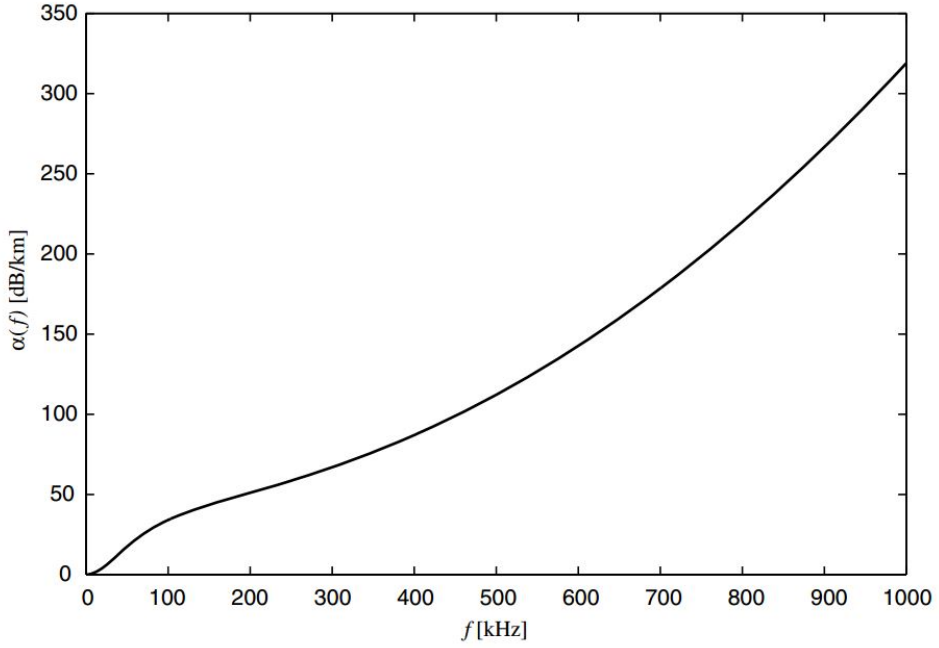


FIGURA 2.2: Absorption loss $\alpha(f)$.

2.1.2 Rumore

Il rumore w_{Rc} può essere considerato come sovrapposizione di quattro effetti:

- **Turbolenza**
- **Interferenza dovuta al passaggio delle navi**
- **Moto delle ondemarine**
- **Rumore termico**

Ogni componente può essere modellata come una variabile aleatoria Gaussiana con PSD (*Power Spectral Density*) continua calcolata sperimentalmente. Le formule che descrivono le PSD si possono trovare in [10]. L'SNR (*Signal to Noise Ratio*) Γ può quindi essere calcolato come segue:

$$\Gamma(d, f_0) = \frac{M_{s_{Tx}}}{P_{w_{Rc}}(f_0) \cdot 2B \cdot a_{Ch}(d, f_0)} \tag{2.3}$$

dove B è la banda del segnale trasmesso e $P_{w_{Rc}}$ è la PSD del rumore assunta costante nella banda attorno ad f_0. Dall'analisi dell'SNR si ottiene che per trasmettere ad una distanza fissata d esiste una frequenza ottimale che massimizza Γ, in figura 2.3 è illustrato questo valore al variare della distanza. Ancora una volta si nota come poter utilizzare frequenze differenti per trasmettere incida fortemente sulle prestazioni di una rete subacquea.

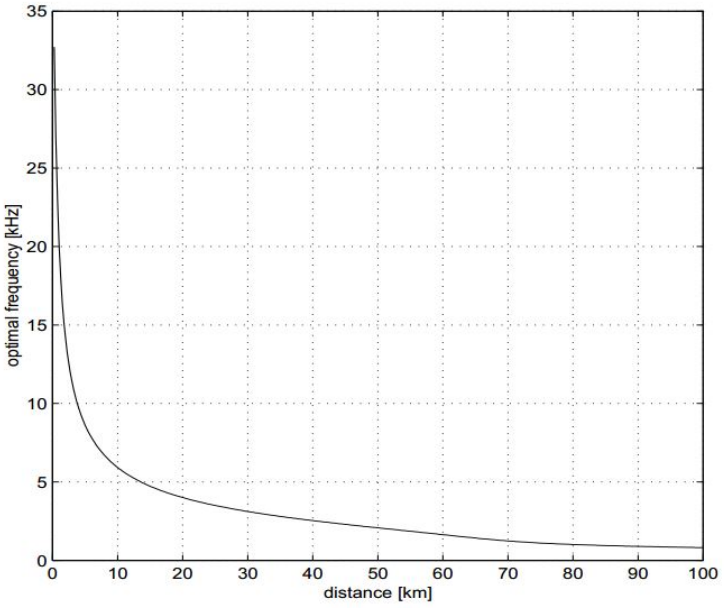


FIGURA 2.3: Frequenza ottimale.

Capitolo 3

DESERT Underwater

DESERT Underwater [2] è un framework per valutare le prestazioni di reti sottomarine che punta ad andare oltre le simulazioni, rendendo possibile la progettazione, l'implementazione e il testing di reti eterogenee con dispositivi reali. L'obiettivo iniziale era di creare un insieme di librerie C++ per supportare il design di protocolli di rete subacquee, ma nel tempo DESERT si è evoluto connettendosi maggiormente ad apparecchiature e scenari reali [11]. Attualmente è collegato con il *World Ocean Simulation System* (WOSS) [12] che permette di generare modelli di canale realistici recuperando ed elaborando automaticamente dati ambientali reali. DESERT è basato sui simulatori NS e NS-MIRACLE, descritti rispettivamente nelle sezioni 3.1 e 3.2.

3.1 Network Simulator 2

NS2 è un simulatore ad eventi discreti creato per supportare la ricerca nel campo delle reti di telecomunicazioni. Fornisce un importante strumento per effettuare simulazioni di *Transmission Control Protocol* (TCP), routing e protocolli multicast su reti cablate o wireless [13]. NS2 nasce nel 1989 come variazione del *REAL network simulation*, è scritto in C++ e le simulazioni sono fatte con una versione orientata agli Oggetti del linguaggio Tcl chiamata oTcl [14]

3.2 NS-MIRACLE

NS-MIRACLE è una libreria che estende il simulatore NS2, sviluppata nel laboratorio SIGNET dell'Università di Padova [15]. La principale funzionalità di questa libreria è quella di consentire comunicazioni *cross-layer* e permettere di simulare dispositivi con design *multi-layer* come ad esempio trasmettitori dotati di diverse interfacce a livello fisico. Per rendere scalabile la comunicazione cross-layer, è stato definito uno standard per questo tipo di messaggi. La figura 3.1 illustra schematicamente la struttura di NS-MIRACLE.

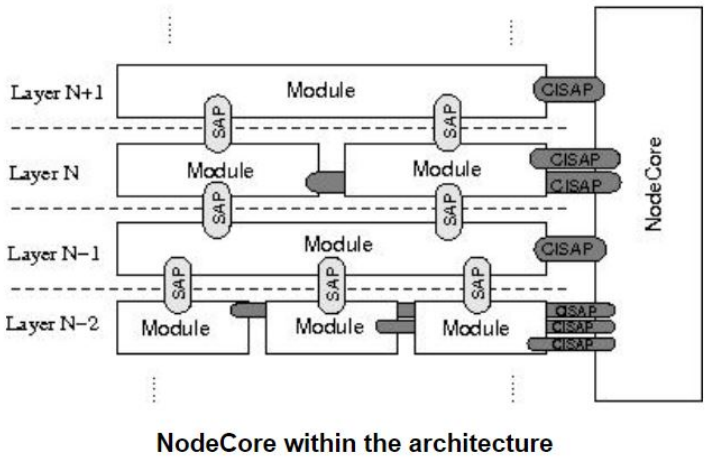


FIGURA 3.1: Struttura del simulatore NS-MIRACLE [16].

I componenti di base del simulatore sono cinque:

- **Module**: è la classe principale che simula un livello del modello *Open Systems Interconnection* (OSI), può contenere qualunque tipo di protocollo o modulo.
- **SAP**: *Service Access Point*, è il canale di comunicazione che collega due moduli adiacenti nel modello OSI. Può essere utilizzato anche per ricavare metriche e salvare tracce delle simulazioni eseguite, in quanto i pacchetti trasmessi passano obbligatoriamente da questi canali.
- **ChSAP**: *Channel Service Access Point*, è un modulo particolare che simula il canale connettendo tutti i nodi della rete, questo modulo deve essere unico.

- **Node Core**: è un modulo speciale che permette le comunicazioni cross-layer, può essere considerato come il *bus* di questi messaggi che ha anche il compito di gestirli e indirizzarli ai moduli corretti.

- **ClSAP**: *Cross-layer Service Access Point*, sono i canali che consentono di inviare *ClMessage* (messaggi cross-layer) al Node Core per poi giungere a moduli non adiacenti a quello che li trasmette. Ci sono due tipi di ClSAP: sincroni e asincroni. I primi richiedono una risposta immediata, nella stessa istanza del ClSAP ricevuto, mentre i secondi non richiedono alcuna risposta. I ClSAP sincroni permettono anche la condivisione di alcune variabili tra moduli diversi. La classe dei ClMessage può essere estesa per generare ogni tipo di ClMessage possibile, contenente dati strutturati e non.

NS-MIRACLE ha una struttura molto flessibile, è infatti possibile definire qualunque numero di moduli ad ogni livello connettendoli con canali SAP.

3.3 DESERT

Come anticipato nell'introduzione di questo capitolo, DESERT è un insieme di librerie C++ che estende NS-MIRACLE per supportare il design e l'implementazione di protocolli di reti subacquee [17]. Come NS-MIRACLE anche questo progetto è sviluppato nel laboratorio SIGNET dell'Università di Padova [15]. La prima versione del simulatore è stata presentata nel 2012 [2] ma è in costante aggiornamento. Nel 2016 è stata rilasciata la versione 2 del progetto [11] che permette di effettuare simulazioni con dati reali, fornisce interfacce a programmi esterni come Matlab e inserisce una predisposizione del sistema alla multimodalità. Negli ultimi anni le comunicazioni sottomarine hanno attirato l'attenzione di molti studi in diversi campi come la difesa militare, la sicurezza, lo studio di alcune specie animali, l'estrazione di idrocarburi e la navigazione, DESERT nasce come strumento per permettere la progettazione e la simulazione di protocolli in reti subacquee eterogeneee, inoltre consente ai ricercatori di poter condividere i risultati delle simulazioni e ripetere facilmente gli esperimenti. In [2] viene fornita una descrizione dei principali moduli inizialmente implementati in DESERT. Nel capitolo 4

verranno presentate delle nuove librerie che permettono di simulare reti adattive multimodali e di utilizzare un nuovo modulo che simula un Jammer di frequenze subacquee.

Capitolo 4

Nuove librerie

In questo capitolo vengono descritte le librerie implementate in questa tesi per il framework DESERT. Nello specifico, queste librerie aggiungono le nuove funzionalità di multimodalità e adattività, inoltre viene data la possibilità di inserire nelle simulazioni l'uso di un jammer. Il codice sviluppato si può trovare su GitHub, infatti tutte le librerie create sono open source ed estendibili per eventuali progetti futuri [18]. Su GitHub è possibile trovare anche degli esempi di simulazione per ogni libreria sviluppata.

4.1 Multimodalità

In questa tesi è stata implementata la multimodalità intesa come possibilità di simulare dispositivi che operano a due differenti bande di trasmissione acustica. Le due interfacce agiscono in parallelo, senza creare mutua interferenza e sono denominate HF (*High Frequency*) e LF (*Low Frequency*). DESERT consente senza particolari sforzi di definire ulteriori interfacce fisiche, sia acustiche (quindi con diverse bande) che ottiche. Per implementare la multimodalità è stato esteso un controllore definito MULTI-TRAFFIC-CONTROLLER posto tra il livello MAC (*Media Access Control*) e il livello rete che gestisce il traffico ricevuto utilizzando l'interfaccia fisica e il MAC corretti per ogni tipologia di messaggio. Come si può vedere dalla figura 4.1, che mostra lo schema a blocchi del simulatore DESERT, il controllore si prende carico di

gestire sia il traffico in entrata che quello in uscita, indirizzando correttamente ogni pacchetto. La struttura è facilmente scalabile con qualsiasi numero di livelli fisici o di tipologie di traffico. Le policies che determinano come il controllore gestisce i pacchetti vengono definite insieme all'intera simulazione in un file tcl, un esempio di utilizzo della libreria è presente in GitHub nella cartella *desert_samples*. Ogni interfaccia fisica è completamente indipendente dalle altre, può quindi utilizzare potenze e modulazioni differenti atte a soddisfare una particolare *Quality of Service* (QoS) richiesta dal tipo di traffico gestito. La multimodalità rende sicuramente le reti più flessibili e, insieme all'adattività, è uno strumento di fondamentale importanza per reagire attivamente a variazioni delle condizioni del canale di trasmissione.

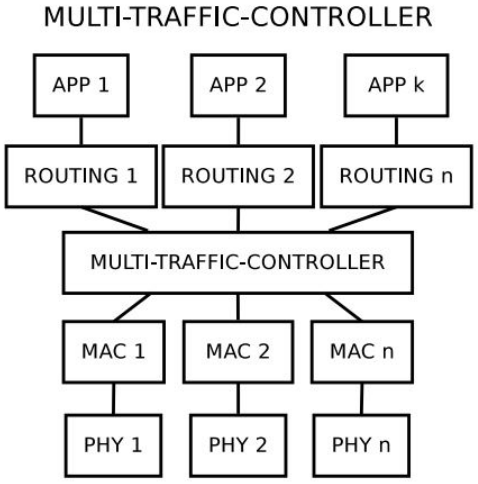


FIGURA 4.1: Schema a blocchi del simulatore DESERT [11].

Un'altra funzione fondamentale del controllore è quella di evitare le collisioni tra pacchetti provenienti da fisici diversi. La libreria creata si chiama *uwmulti-band* ed estende dalla classe *uwmultitrafficcontrol* inserita nella versione 2 di DESERT. *Uwmulti-band* è un addon che permette di creare due interfacce fisiche diverse utilizzando i comandi *addHFLowLayer* e *addLFLowLayer* seguiti da un numero intero che identifica la tipologia di traffico gestita e dal nome del livello fisico. Il controllore si prende carico di verificare che lo stack HF o LF non sia stato precedentemente definito per evitare errori nella simulazione, in caso contrario l'esecuzione viene interrotta ed è mostrato un messaggio di errore. Oltre a queste funzionalità, *uwmulti-band* ha anche due interfacce denominate *sendDownHF* e *sendDownLF* che sono state inserite per progetti futuri, in particolare per rendere possibile l'utilizzo della multimodalità

con altre applicazioni che sfruttano il protocollo di rete GUWMANET [19].

4.2 Adattività

Un dispositivo adattivo è in grado di modificare i parametri di trasmissione e ricezione durante il suo normale funzionamento, adattandosi così alle condizioni del canale e/o alle richieste in termini di QoS del traffico gestito. Alcuni esempi di parametri modificabili sono la modulazione, la potenza e la soglia minima per la quale un dispositivo riceve un pacchetto. In DESERT è stata implementata l'adattività intesa come possibilità di cambiare parametri durante la simulazione utilizzando appositi comandi nel file tcl che la definisce. Inoltre con l'utilizzo di un preambolo che precede il pacchetto si dà la possibilità ai nodi della rete di trasmettere con qualunque modulazione disponibile senza accordarsi precedentemente con gli altri dispositivi. Questa nuova funzionalità permette di creare scenari di simulazione molto più ampi, concentrati sull'adattamento e sull'ottimizzazione delle trasmissioni in base alle condizioni di rumore e interferenza. Nel capitolo 5 verranno mostrate alcune simulazioni esemplificative di questo concetto. L'adattività è gestita dallo stesso controllore definito per la multimodalità (uwmulti-band) che si prende carico di inviare ClMsg (classe di messaggi cross-layer che sfrutta i ClSAP definiti nel paragrafo 3.2) al livello fisico per riparametrizzare le trasmissioni future. Per facilitare l'adattamento al canale di comunicazione, sono stati implementati anche dei ClMsg che compiono il percorso inverso, cioè dal fisico al controllore. Questi messaggi contengono informazioni riguardanti lo stato del canale durante la ricezione dell'ultimo pacchetto, come il valore assoluto massimo di ampiezza del segnale, l'RMS (*Root Mean Square*) dei vari simboli, l'azimuth e altri. Questa tipologia di ClMsg è stata predisposta ed implementata per applicazioni future ma non è ancora stata utilizzata in DESERT. La tabella 4.1 fornisce una descrizione dei comandi utilizzabili nel simulatore, ogni volta che una di queste istruzione viene invocata da tcl, il controllore crea uno specifico ClMsg sincrono e lo invia al fisico corretto. I ClMsg implementati sono un'estensione della classe ClMsgUwPhy, per ogni parametro è definito uno specifico messaggio contenente solo quel valore. I comandi vengono invocati dal file tcl con la seguente sintassi:

$ns at time "$ctr($i) comando valore nome_fisico

dove *time* è l'istante in cui si vuole cambiare il parametro, *ctr* è il nome del controllore, *i* è il numero che identifica il nodo, $nome\_fisico$ è il nome dato al livello fisico che si vuole riparametrizzare. La tebella 4.1 elenca i comandi disponibili per le simulazioni e i rispettivi valori che si possono impostare da tcl. Se il valore è *double* significa che può essere utilizzato qualunque numero reale, se è un elenco significa che il valore è di tipo enum e solo quei parametri sono accettati.

Comando	**Valori**	**Descrizione**
setThreshold	double	Ridefinisce il parametro AcquisitionThreshold_dB_ cioè la soglia minima per ricevere il pacchetto in dB
setSignalType	signal_ifs signal_juwel signal_ucac signal_ofdm signal_msml signal_scte signal_dsss signal_frss signal_mfsk	Tipologie di segnali in trasmissione
setModulationType	modulation_qpsk modulation_bpsk modulation_mfsk modulation_8psk	Modulazione utilizzata per trasmettere
setCodingType	coding_alpha coding_beta coding_gamma coding_delta coding_eta	Codifica utilizzata per trasmettere
setSourceLevel	double	Ridefinisce il parametro MaxTxSPL_dB_ cioè la potenza utilizzata per trasmettere in dB

Tabella 4.1: Comandi per creare i ClMsg che ridefiniscono i parametri in trasmissione.

Nella tabella 4.2 vengono elencati altri tipi di ClMsg implementati ma non ancora utilizzabili nel simulatore DESERT, questi comandi sono stati inseriti per essere utilizzati in progetti futuri basati sull'adattività.

La tabella 4.3 illustra i comandi che possono essere invocati da tcl per fare in modo che il fisico restituisca i valori interessati quando riceve un pacchetto, questi dati vengono inviati

Comando	Valori
setDecodingSteps	double
setRandomSeed	double
setQpskIterationDecodingSteps	double
setGain	double
setDetectionLimit	double

Tabella 4.2: ClMsg usati dal trasmettitore, definiti per progetti futuri.

al controllore che poi li può inoltrare ai livelli superiori per permettere a protocolli come GUWMANET di scegliere i parametri migliori per trasmettere i nuovi pacchetti.

Comando	Valori
getMaxAmp	double
getRms1	double
getRms2	double
getRms3	double
getRms4	double
getDoppler	double
getTimeSpread	double
getPeakDistance	double
getElevation	double
getAzimuth	double

Tabella 4.3: ClMsg usati dal ricevitore, definiti per progetti futuri.

Per permettere la completa adattività dei nodi (cioè poter cambiare i parametri in ogni istante senza rispettare schedule fissati) è necessario che i dispositivi abbiano a disposizione un modo per avvisare gli altri nodi riguardo la modulazione con cui viene trasmesso un pacchetto. Per questo motivo è stato introdotto un nuovo header che ha la funzione di preambolo. L'header in questione è denominato UWPRE ed è configurabile da tcl, in particolare si può modificarne la grandezza in byte cambiando il parametro pre_size_ che di default è fissato a 2 byte. Il preambolo è inviato sempre con una modulazione prefissata e contiene le informazioni necessarie a ricevere correttamente il resto del pacchetto, in particolare contiene la modulazione utilizzata, il tipo di codifica e il tipo di segnale come definito nella tabella 4.1. Per rendere più flessibile il simulatore è stato definito un flag denominato *preamble_setting* che permette di disabilitare l'uso del preambolo (di default è attivo), in questo caso ogni dispositivo sarà informato a priori e senza possibilità di errore sui parametri altrimenti contenuti nel preambolo. Il modulo che gestisce l'invio e la ricezione del preambolo è a livello fisico e si chiama *uwadaptivephysicaldb*,

estende da *uwphysicaldb* ed è anche incaricato di gestire l'invio e la ricezione dei ClMsg al controllore. In DESERT il preambolo è stato implementato come parte integrante del pacchetto e quindi non viene trasmesso separatamente ma all'interno del pacchetto stesso, il dispositivo che riceve il messaggio prova prima a leggere il preambolo, se riesce a leggerlo correttamente, procede leggendo il contenuto del pacchetto vero e proprio. In particolare quando il fisico di un nodo riceve un messaggio, effettua 4 controlli prima di inoltrarlo ai livelli superiori:

- **SNR del preambolo:** il fisico calcola l'SNR come potenza del segnale ricevuto su potenza del rumore e in base al valore ottenuto consulta una *Look Up Table* (LUT) che fornisce la *Packet Error Rate* (PER) rispettiva. Una volta ottenuta la PER, il modulo fisico utilizza una variabile aleatoria uniforme distribuita in [0,1] per valutare se il preambolo è ricevuto correttamente, nello specifico se il valore della variabile è inferiore alla PER il pacchetto viene scartato e si incrementa il contatore ErrorPreNoise in caso contrario si procede al secondo controllo.

- **SIR del preambolo:** il SIR (*Signal to Interference Ratio*) è calcolato come potenza del segnale ricevuto su potenza dei segnali che creano interferenza, oltre al SIR viene calcolato anche l'overlap cioè la percentuale di tempo in cui si verifica l'interferenza rispetto al totale del tempo impiegato per ricevere il pacchetto. In base all'overlap viene consultata una specifica LUT che fornisce la PER rispetto al valore di SIR ottenuto, la definitiva ricezione del preambolo è poi valutata con una variabile aleatoria in modo completamente analogo all'SNR.

- **SNR del pacchetto:** se il preambolo è correttamente ricevuto, il nodo è stato informato circa la modulazione, il tipo di segnale e la codifica utilizzata per inviare il messaggio, in questo modo è possibile calcolare l'SNR del pacchetto e consultare la corretta LUT (dipende dalla frequenza e dalla modulazione) per ottenere la PER e verificare la corretta ricezione del pacchetto.

- **SIR del pacchetto:** se anche il controllo sull'SNR dà esito positivo, l'ultimo test da effettuare riguarda il SIR del pacchetto, in modo analogo al preambolo si calcola

l'overlap e si ottiene la PER corretta, poi si utilizza una variabile aleatoria per verificare la definitiva ricezione del pacchetto.

In caso di fallimento in uno dei quattro controlli sopracitati vengono incrementati rispettivamente i seguenti contatori: *ErrorPreNoise*, *ErrorPreInterf*, *ErrorPktNoise*, *ErrorPktInterf*. I valori di questi contatori possono essere ottenuti da tcl in qualsiasi istante della simulazione. Le LUT contengono valori di SNR discreti quindi si valuta l'SNR (o il SIR) più vicino a quello considerato. I valori di overlap devono essere predefiniti nel file tcl utilizzando il seguente comando dal modulo fisico:

nome_fisico addOverlap valore

Ad ogni overlap deve corrispondere una LUT che contiene il SIR e la PER riguardanti quella percentuale di overlap. Le LUT sono file di testo contenenti due colonne di valori double separate da uno spazio (nelle impostazioni di DESERT è possibile cambiare il separatore se si preferisce usare \t), la prima colonna rappresenta l'SNR o il SIR a seconda del tipo di LUT, la seconda è la PER corrispondente. I nomi dei file contenenti le LUT devono seguire la seguente sintassi:

- **LUT SNR preambolo:** preamble_frequenza
- **LUT SIR preambolo:** preamble_frequenza_overlap_SIR
- **LUT SNR pacchetto:** modulazione_frequenza
- **LUT SIR pacchetto:** modulazione_frequenza_overlap_SIR

Dove $frequenza$ indica l'interfaccia multimodale utilizzata (HF o LF), questa parte può anche essere omessa se non si utilizzano dispositivi multimodali. Alcuni esempi di nomi corretti dei file sono: preamble_HF, preamble_LF_50_SIR, mfsk_LF, qpsk_HF_100_SIR. Se si usa la multimodalità è anche necessario utilizzare il comando setFrequency nel file tcl per informare il modulo fisico sul tipo di frequenza utilizzata (HF o LF), la sintassi è la seguente:

nome_fisico setFrequency valore

in cui *valore* è appunto HF o LF. La tabella 4.4 illustra i valori di default che il modulo uwadaptivephysicaldb utilizza per trasmettere i pacchetti se non vengono cambiati con i comandi della tabella 4.1.

Parametro	**Valore**
signal_type	signal_ifs
modulation_type	modulation_bpsk
coding_type	coding_alpha

Tabella 4.4: Valori di default inseriti nel preambolo.

4.3 Jammer

Il jammer è un disturbatore di frequenze, questo dispositivo è utilizzato in vari campi sia militari che civili. Nelle comunicazioni subacquee può essere anche utilizzato come fonte di rumore per simularne alcune componenti come descritto nel capitolo 1.1.3. Esempi di simulazioni con jammer sono forniti nel capitolo 5. In DESERT il jammer è stato implementato come modulo a livello data-link e si chiama uwjammer, questo modulo invia tutti i pacchetti che riceve dai livelli superiori senza ascoltare il canale o aspettare tempi di backoff come solitamente i dispositivi fanno utilizzando protocolli CSMA-ALOHA o altri protocolli di accesso al mezzo. L'obiettivo del jammer è quello di creare più interferenza possibile nel canale di trasmissione impedendo agli altri nodi di comunicare correttamente. Nel capitolo 5 sono illustrate le prestazioni del jammer che può raggiungere bitrate molto elevate grazie al fatto che non deve rispettare nessuno scheduling per trasmettere. In DESERT il modulo contiene una coda chiamata Q_data che mantiene in memoria tutti i pacchetti ancora da trasmettere, la grandezza della coda è regolabile da tcl ridefinendo il parametro buffer_data_pkts_ che di default vale 50. La capacità massima della coda non deve comunque superare il valore MAX_BUFFER_SIZE fissato a 100. Il jammer è regolato da una semplice macchina a stati finiti formata da due stati: UWJAMMER_STATUS_IDLE e UWJAMMER_STATUS_BUSY. Il primo stato indica che il jammer ha inviato i pacchetti nella coda e sta attendendo l'arrivo di un nuovo messaggio, il secondo stato indica che la trasmissione è in esecuzione. Quando il jammer riceve un pacchetto dal livello superiore lo inserisce nella coda, se questa non è piena, altrimenti il pacchetto viene

scartato e si incrementa il contatore n_jam_discarded. Quando il jammer è in stato IDLE controlla se la coda è vuota, altrimenti procede all'invio del primo messaggio estratto dalla coda e incrementa il contatore n_jam_sent. Come gli altri moduli precedentemente illustrati, anche uwjammer dispone di alcuni comandi che possono essere invocati da tcl per gestire la simulazione:

- **initialize:** inizializza il modulo e la macchina a stati che lo descrive.
- **getDataQueueSize:** restituisce il numero di pacchetti presenti nella coda nell'istante in cui viene invocato.
- **getJamSent:** restituisce il numero di pacchetti inviati dal jammer.
- **getJamDiscarded:** restituisce il numero di pacchetti scartati perchè quando arrivano al jammer la coda è piena.
- **getDataDiscarded:** restituisce il numero di pacchetti ricevuti dal jammer e scartati.

I pacchetti trasmessi dal jammer non contengono alcun tipo di informazione utile perchè l'obiettivo del dispositivo è esclusivamente quello di disturbare le comunicazioni tra gli altri nodi. Quando il jammer riceve un messaggio, questo viene scartato a priori (non c'è alcuna distinzione tra diverse tipologie di pacchetto) e viene incrementato il contatore n_data_discarded.

Capitolo 5

Simulazioni

In questo capitolo vengono presentate e confrontate quattro simulazioni differenti create per mostrare alcuni possibili scenari in cui multimodalità e adattività incrementano sensibilmente le prestazioni di una rete. Nelle simulazioni 2-3-4 viene anche utilizzato il jammer come fonte di disturbo per gli altri nodi. I file tcl che definiscono tutte le simulazioni possono essere trovati su GitHub nella cartella *desert_samples* insieme alle LUT utilizzate e ad alcuni script MATLAB che permettono di riordinare e analizzare i dati della rete. In particolare un file chiamato *data_analysis* permette di riorganizzare i dati in matrici più semplici da analizzare, mentre *metriche* crea i grafici utilizzando i dati generati da *data_analysis*. Le metriche prese in considerazione sono due:

- **Throughput:** è fornito in bps e corrisponde al numero di bit ricevuti da un nodo rispetto al tempo. Nei grafici seguenti il throughput considerato è "istantaneo" cioè viene misurato in finestre temporali di 30 s (questo parametro è facilmente modificabile nei file tcl) e inserito nei grafici al variare del tempo.
- **Packet Delivery Ratio (PDR):** è il numero di pacchetti ricevuti rispetto a quelli inviati su un determinato link della rete. Questa metrica si può visualizzare nelle Tabelle poste dopo i grafici di ogni simulazione.

Nelle sezioni che descrivono le varie simulazioni vengono elencati i parametri utilizzati, tutti quelli non citati sono stati settati con i valori di default definiti dal simulatore DESERT. Nei grafici presenti in questo capitolo le linee orizzontali più spesse indicano il throughput medio calcolato sul range di tempo occupato dalla linea. In tutte le simulazioni i parametri utilizzati per i fisici HF e LF rispecchiano i dispositivi reali EvoLogics S2CR 48/78 Acoustic Modem [20] e S2CR 18/34 Acoustic Modem [21] rispettivamente. In Figura 5.1 vengono mostrati i due modem impiegati nelle simulazioni.

(A) HF (B) LF

FIGURA 5.1: Modem HF e LF utilizzati.

5.1 SNR e SIR

Le LUT contenenti i valori di SNR sono state create a partire dalle classiche formule di *Bit Error Rate* (BER) definite per le modulazioni BPSK, QPSK, MFSK, 8PSK [1] e mostrate nelle equazioni 5.1-5.4 in cui γ è l'SNR per bit. Nelle simulazioni effettuate la modulazione MFSK è stata utilizzata con $M = 8$.

$$\textbf{BER BPSK} = Q(\sqrt{2\gamma}) \tag{5.1}$$

$$\textbf{BER QPSK} = Q(\sqrt{\gamma}) \tag{5.2}$$

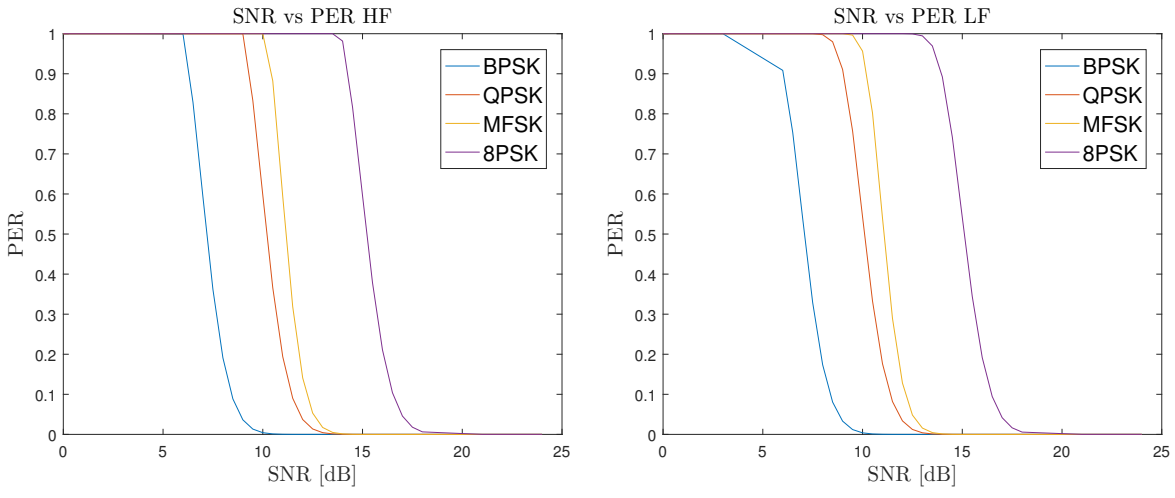
$$\textbf{BER MFSK} = \frac{M}{2}Q(\sqrt{\gamma}) \tag{5.3}$$

$$\textbf{BER 8PSK} = \frac{2}{3}Q(\sqrt{2\gamma}\sin\frac{\pi}{8}) \tag{5.4}$$

Nelle LUT è salvata la PER, calcolata a partire dalla BER con la seguente formula:

$$\textbf{PER} = 1 - (1 - BER)^n \tag{5.5}$$

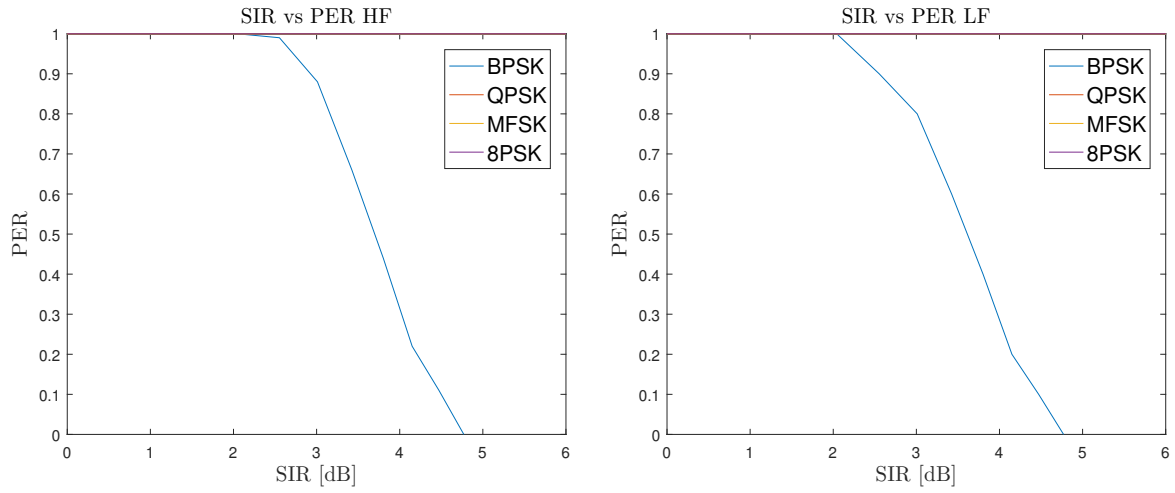
Dove n è il numero di bit contenuti in un pacchetto. L'SNR così ottenuto è stato inserito nelle LUT di tipo LF, per calcolare le LUT di tipo HF è stato utilizzato un parametro $\epsilon \geq 1$ che rappresenta una penalità rispetto alle LUT di tipo LF. In particolare tutti i valori di PER ottenuti per LF vengono moltiplicati per ϵ e posti a 1 se la PER diventa maggiore di 1. Nelle simulazioni è stato utilizzato $\epsilon = 1.1$ quindi i valori di PER delle LUT HF sono il 10% più elevati rispetto a quelle di tipo LF. Su GitHub è possibile trovare uno script MATLAB che crea automaticamente tutte le LUT e permette di modificare i parametri n, M e ϵ. In Figura 5.2a e 5.2b vengono illustrati i risulati ottenuti per le PER HF e LF rispettivamente. I valori di SNR utilizzati sono $[0, 3, 6 : 0.5 : 18, 21, 24]$ dB.



(A) (B)

FIGURA 5.2: LUT HF e LF rispetto i valori di SNR.

I valori di PER rispetto al SIR delle modulazioni QPSK e 8PSK sono stati presi da dati sperimentali ottenuti nel progetto RACUN [22], per cui la PER in presenza di interferenza è



(A) (B)

FIGURA 5.3: LUT HF e LF rispetto i valori di SIR.

fissa a 1. La PER della modulazione MFSK è stata assunta costante uguale ad 1 in quanto è meno robusta della modulazione QPSK e quindi non può avere probabilità di errore inferiori. Al contrario, la modulazione BPSK è la più resistente agli errori, per questo motivo la PER è stata costruita partendo dal fatto che con potenza di interferenza inferiore al 33% rispetto alla potenza del segnale ricevuto, è possibile ricevere informazione con una certa probabilità. Sopra al 33% la probabilità di errore resta costante ad 1. Le Figure 5.3a e 5.3b illustrano l'andamento della PER rispetto al SIR per le quattro modulazioni considerate, per QPSK, MFSK e 8PSK la probabilità si sovrappone e quindi non è visibile. La costruzione delle LUT di tipo HF è analoga a quelle dell'SNR, utilizzando il parametro $\epsilon = 1.1$. Nelle simulazioni effettuate è stato considerato solo un overlap del 100% quindi in presenza di qualsiasi tipo di interferenza vengono utilizzate le LUT con overlap massimo perchè i dati sperimentali presi dal progetto RACUN mostrano che per le modulazioni QPSK e 8PSK la PER è sempre 1 indipendentemente dalla percentuale di overlap.

5.2 Scenario base e multimodalità

La prima simulazione fatta mette in risalto le differenze in termini di throughput e PDR in trasmissioni LF e HF, serve inoltre come confronto per le successive simulazioni. La topologia

della rete utilizzata è mostrata in Figura 5.4, 4 nodi esterni sono collegati ad un nodo centrale con una doppia interfaccia (HF e LF) mentre ogni dispositivo è legato a quello a fianco con un'unica interfaccia di tipo LF. Nell'analisi il nodo centrale è chiamato C mentre quelli attorno sono definiti da un numero da 0 a 3. Questa topologia verrà riproposta anche nelle simulazioni successive ma saranno variate le condizioni del canale e altri parametri determinanti. Come si può notare dalla Figura 5.4 i due nodi in basso distano 3 km da quello centrale mentre i due in alto 1 km.

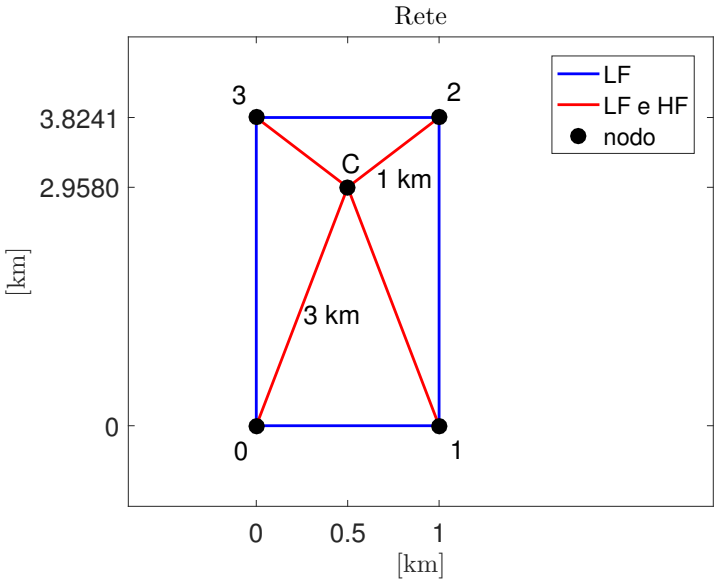


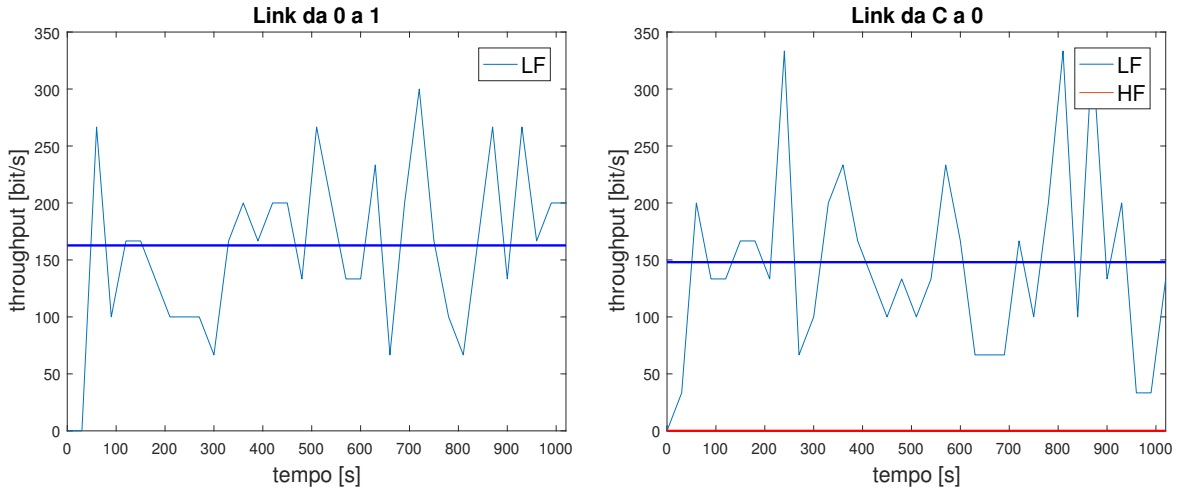
FIGURA 5.4: Scenario base: topologia della rete.

Nella Tabella 5.1 vengono mostrati tutti i parametri utilizzati nella prima simulazione, sia per il modem HF che per quello LF. Il periodo *Constant Bit Rate* (CBR) è la distanza di tempo che intercorre tra la generazione di due pacchetti a livello applicazione, la soglia di acquisizione è il valore minimo di SNR in dB per cui il dispositivo prova a ricevere un messaggio. Tutti i nodi della rete (anche nelle simulazioni successive) utilizzano un protocollo di accesso al mezzo di tipo CSMA-ALOHA.

Parametro	**HF**	**LF**
portante	63 kHz	26 kHz
banda	30 kHz	16 kHz
bitrate	15.6 kbps	6.95 kbps
potenza	180 dB re μPa @ 1 m	180 dB re μPa @ 1 m
dimensioni pacchetto	125 byte	125 byte
periodo CBR	2s	3.5s
soglia di acquisizione	1 dB	1 dB
preamble setting	1	1
preamble size	2 byte	2 byte
profondità	100 m	100 m

Tabella 5.1: Scenario base: parametri di simulazione della rete.

I grafici in Figura 5.5 indicano l'evolversi del throughput al variare del tempo in alcuni link significativi. Come era prevedibile l'interfaccia HF che collega i nodi 0 e 1 a C non riesce a ricevere alcun messaggio perchè la distanza è troppo elevata, in link di questo tipo è necessario utilizzare una portante sensibilmente più bassa. Ovviamente i link di tipo HF tra C e 3 o tra 2 e C hanno prestazioni migliori rispetto a quelli LF. Interessante è anche notare il diverso throughput medio che c'è nel collegamento tra C e 0 rispetto a quello tra 1 e C, le distanze sono identiche ma l'interferenza presente attorno al nodo C è molto più alta rispetto a quella vicino ad 1 e quindi anche il throughput ne risente. Discorso analogo può essere fatto mettendo a confronto il collegamento tra 0 e 1 con quello tra 3 e 2: il throughput tra 0 e 1 è più alto perchè meno contaminato dall'interferenza delle altre trasmissioni.



(A) (B)

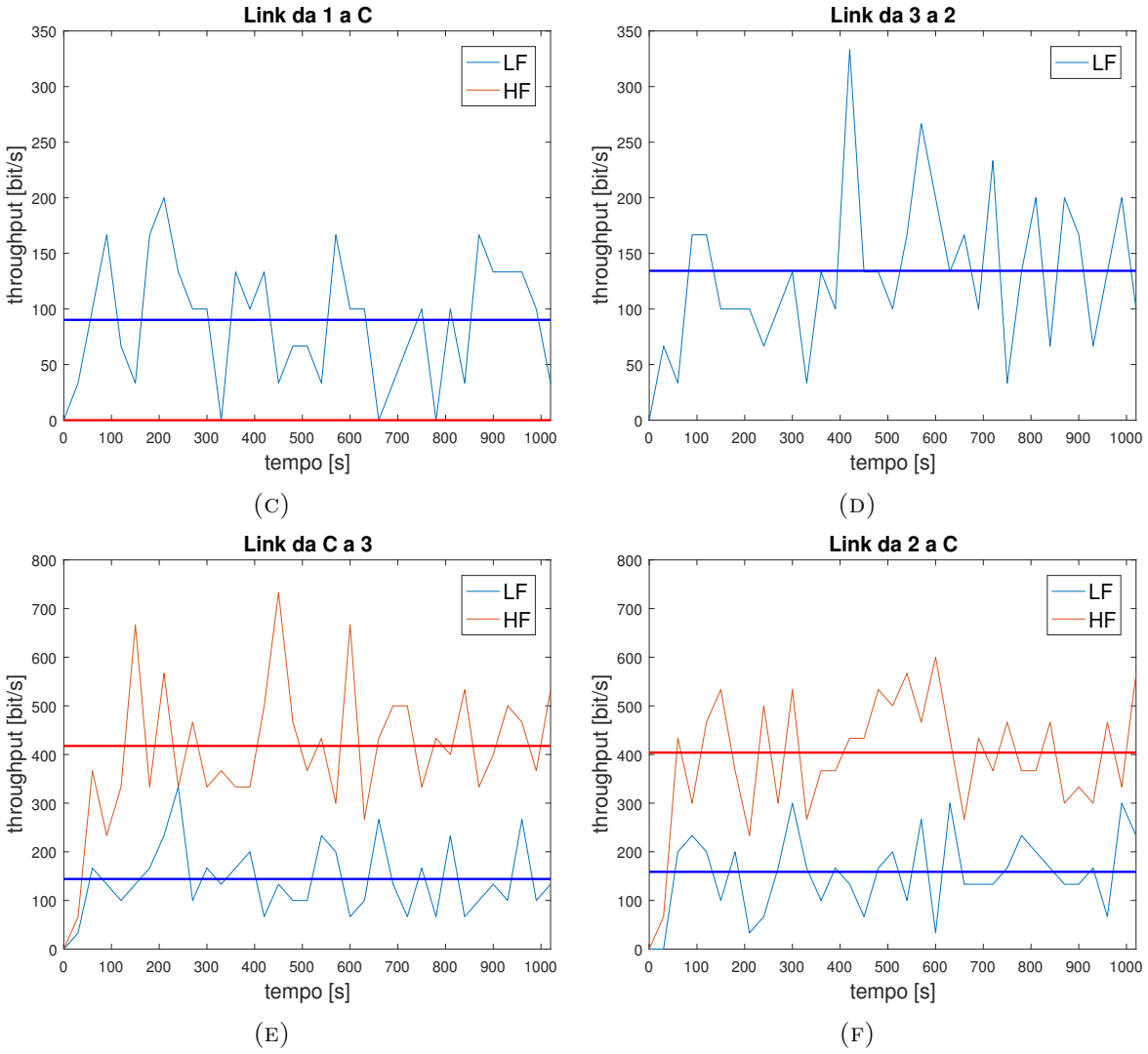


FIGURA 5.5: Scenario base: prestazioni della rete

La Tabella 5.2 mostra il PDR di tutti i link della rete, i risultati ottenuti ricalcano quanto osservato dallo studio del throughput: le trasmissioni verso C hanno PDR inferiori perchè l'interferenza è maggiore, come osservato nei grafici precedenti. Il simbolo "-" indica che quel collegamento non è attivo nella rete. Con questa prima simulazione si è voluto mettere in risalto la differenza in termini di throughput e PDR in base all'utilizzo di bande e portanti differenti, questo sottolinea l'importanza della multimodalità nelle trasmissioni subacquee: condizioni differenti necessitano di parametri e tecnologie diverse per ottenere link stabili ed efficienti.

Link	HF	LF	Link inverso	HF	LF
0 → C	0%	33.55%	C → 0	0%	51.01%
1 → C	0%	34.07%	C → 1	0%	50.08%
2 → C	82.14%	51.59%	C → 2	90.58%	52.30%
3 → C	79.68%	44.44%	C → 3	91.43%	55.35%
0 → 1	-	57.84%	1 → 0	-	52.40%
1 → 2	-	28.82%	2 → 1	-	34.88%
2 → 3	-	48.82%	3 → 2	-	46.62%
3 → 0	-	32.22%	0 → 3	-	30.54%

TABELLA 5.2: Scenario base: Packet Delivery Ratio di tutti i link della rete.

5.3 Jammer e adattività

In questa simulazione viene utilizzato un jammer come fonte di disturbo per i dispositivi di tipo HF, di conseguenza alcuni nodi sfrutteranno l'adattività per cercare di ottimizzare le trasmissioni in base alle condizioni del canale, altri manterranno gli stessi settaggi e verrà fatto un confronto. La topologia è analoga a quella della prima simulazione, l'unica differenza è che al posto del nodo 2 ora è presente un jammer, un'illustrazione della rete è fornita in Figura 5.6.

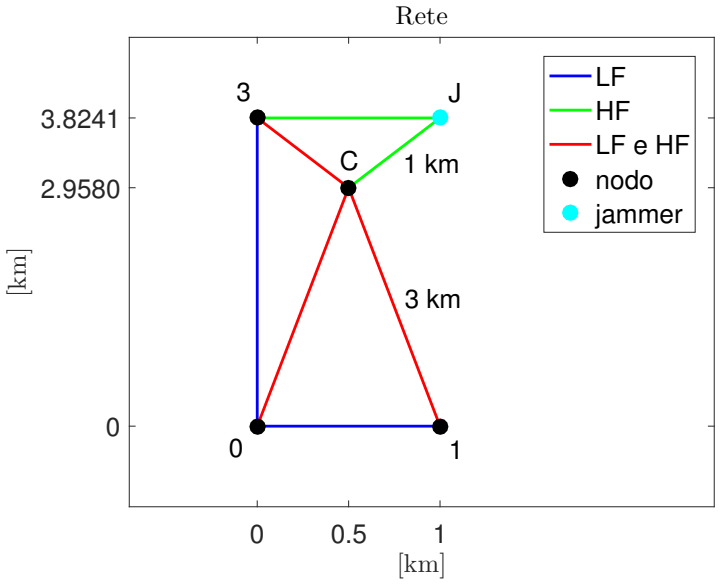


FIGURA 5.6: Jammer e adattività: topologia della rete.

La simulazione dura 1000 s, inizialmente il jammer (J) è spento e quindi non modifica in alcun modo le trasmissioni tra gli altri nodi della rete, dopo 500 s viene attivato J sulla banda HF.

I nodi C e 3 analizzati in questa simulazione risentono notevolemente del cambiamento e le prestazioni decadono velocemente. Il jammer è settato con periodo CBR di 0.5 s, potenza di trasmissione di 165 dB re μPa @ 1 m e *buffer_data_pkts_* di 50, gli altri parametri sono analoghi ai dispositivi HF elencati in Tabella 5.3.

Parametro	**HF**	**LF**
portante	63 kHz	26 kHz
banda	30 kHz	16 kHz
bitrate	15.6 kbps	6.95 kbps
potenza	165 dB re μPa @ 1 m	165 dB re μPa @ 1 m
dimensioni pacchetto	125 byte	125 byte
periodo CBR	2s	3.5s
soglia di acquisizione	1 dB	1 dB
preamble setting	1	1
preamble size	2 byte	2 byte
profondità	100 m	100 m

Tabella 5.3: Jammer e adattività: parametri iniziali di simulazione della rete.

La Tabella 5.3 mostra tutti i parametri iniziali della rete, i dispositivi utilizzano la modulazione 8PSK. A seguito dell'attivazione del jammer il nodo C si adatta modificando la potenza di trasmissione e la modulazione che diventano rispettivamente 180 dB re μPa @ 1 m e BPSK per cercare di sopperire all'interferenza generata dal jammer. Al contrario il nodo 3 continua a trasmettere con gli stessi settaggi iniziali. I comandi utilizzati nel file tcl per utilizzare l'adattività sono i seguenti:

$ns at 500 "$ctr($nodeC) setModulationType modulation_bpsk PHY_HF"

$ns at 500 "$ctr($nodeC) setSourceLevel 180 PHY_HF"

I grafici 5.7a e 5.7b mostrano le differenze dei due approcci (adattivo e statico) per quanto riguarda il throughput, la linea tratteggiata mostra il throughput medio calcolato dopo l'attivazione del jammer quindi da 500 s a 1000 s. In questi grafici si nota chiaramente come l'adattività aiuti i dispositivi a massimizzare le prestazioni anche in presenza di condizioni pessime come quelle considerate nella simulazione. In questo caso sono state modificate solo la potenza di trasmissione e la modulazione ma in modem reali si potrebbero creare sistemi adattivi in grado di variare tutti i parametri di trasmissione e ricezione in base ai cambiamenti delle condizioni del canale.

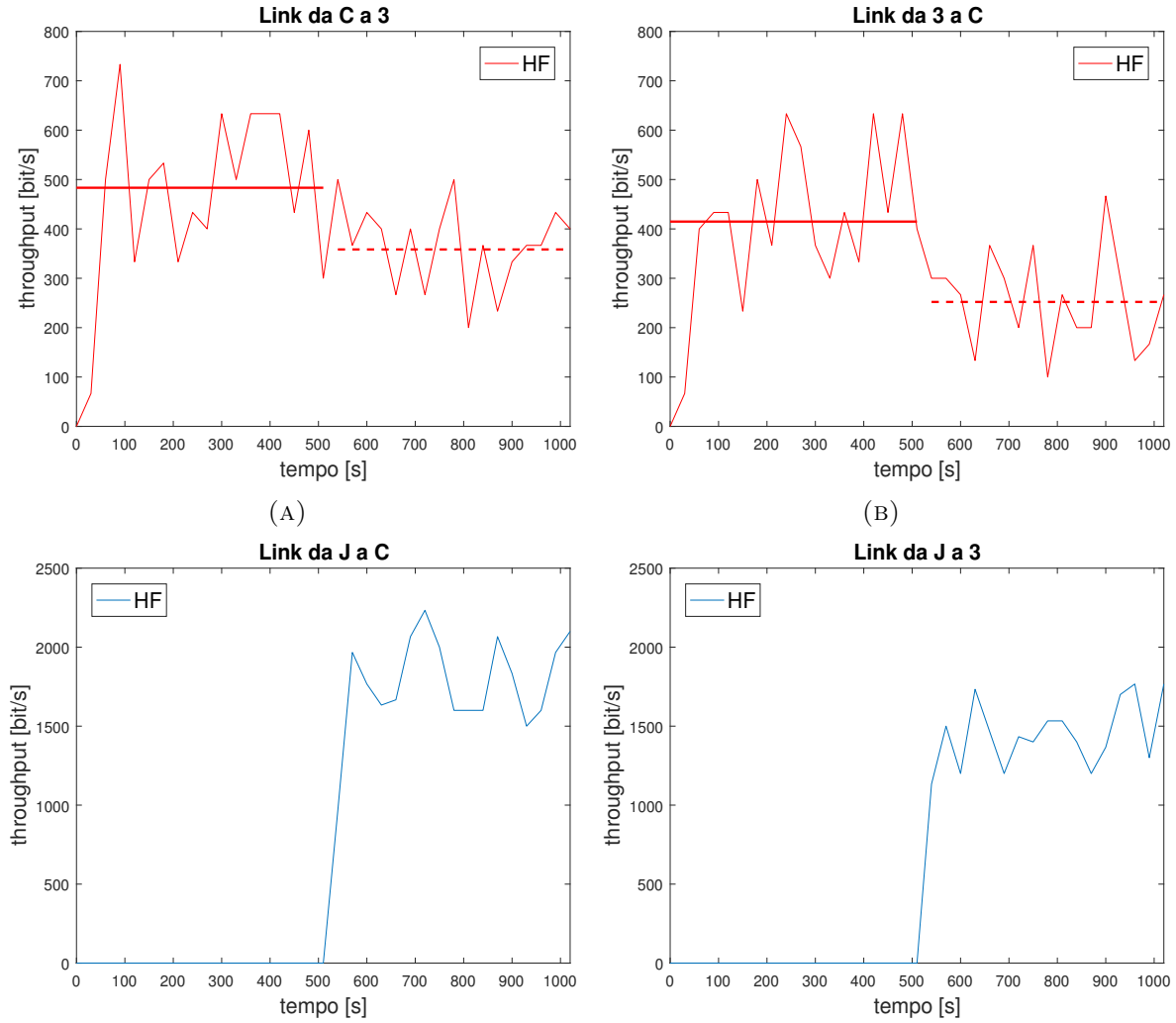


FIGURA 5.7: Jammer e adattività: prestazioni della rete

Le Figure 5.7c e 5.7d mostrano invece il throughput del jammer nei confronti del nodo C e del 3 rispettivamente, in questo caso i valori sono molto elevati se comparati agli altri dispositivi perchè il jammer non utilizza un protocollo di accesso al mezzo di tipo CSMA ma semplicemente invia tutti i pacchetti che riesce ad inviare. La Tabella 5.4 elenca i valori di PDR ottenuti nella simulazione dei link che risentono dell'attività del jammer. Il PDR del link da C a 3 è del 15% maggiore rispetto a quello del collegamento tra 3 e C, quindi il miglioramento introdotto dall'adattività è rilevante.

Link	HF	Link inverso	HF
3 → C	69.70%	C → 3	84.77%
J → C	86.27%	C → J	-
J → 3	79.47%	3 → J	-

Tabella 5.4: Jammer e adattività: Packet Delivery Ratio della rete.

5.4 Jammer e multimodalità

La terza simulazione ricalca la topologia e lo scenario della seconda ma in questo caso viene sottolineato l'effetto della multimodalità come metodo di reazione a disposizione dei nodi nel caso in cui un canale di trasmissione venga disturbato a tal punto che non sia più usufruibile. Il jammer utilizza una potenza di trasmissione di 180 dB re μPa @ 1 m, un periodo CBR di 0.05 s e un *buffer_data_pkts_* di 50, quindi in questo caso è molto più aggressivo rispetto alla simulazione sull'adattività. La Tabella 5.5 mostra i parametri utilizzati nell'esperimento.

Parametro	HF	LF
portante	63 kHz	26 kHz
banda	30 kHz	16 kHz
bitrate	15.6 kbps	6.95 kbps
potenza	180 ddB re μPa @ 1 m	180 dB re μPa @ 1 m
dimensioni pacchetto	125 byte	125 byte
periodo CBR	2s	3.5s
soglia di acquisizione	1 dB	1 dB
preamble setting	1	1
preamble size	2 byte	2 byte
profondità	100 m	100 m

Tabella 5.5: Jammer e multimodalità: parametri di simulazione della rete.

Quando i nodi C e 3 si accorgono della presenza del jammer sulle frequenza HF, reagiscono in modo differente: il nodo C utilizza la multimodalità per interrompere la trasmissione sul canale HF e passare alle frequenze LF mentre il nodo 3 continua a provare ad utilizzare il canale HF senza successo. Le prestazioni dei due nodi sono illustrate nelle Figure 5.8a e 5.8b, qui la differenza di throughput è enorme, dall'attivazione del jammer il nodo 3 non riesce più a trasmettere nulla mentre il nodo C mantiene un throughput rilevante anche se inevitabilmente peggiore rispetto alla prima parte della simulazione. Analogamente alla seconda simulazione, le Figure 5.8c e 5.8d mostrano il throughput ottenuto dal jammer.

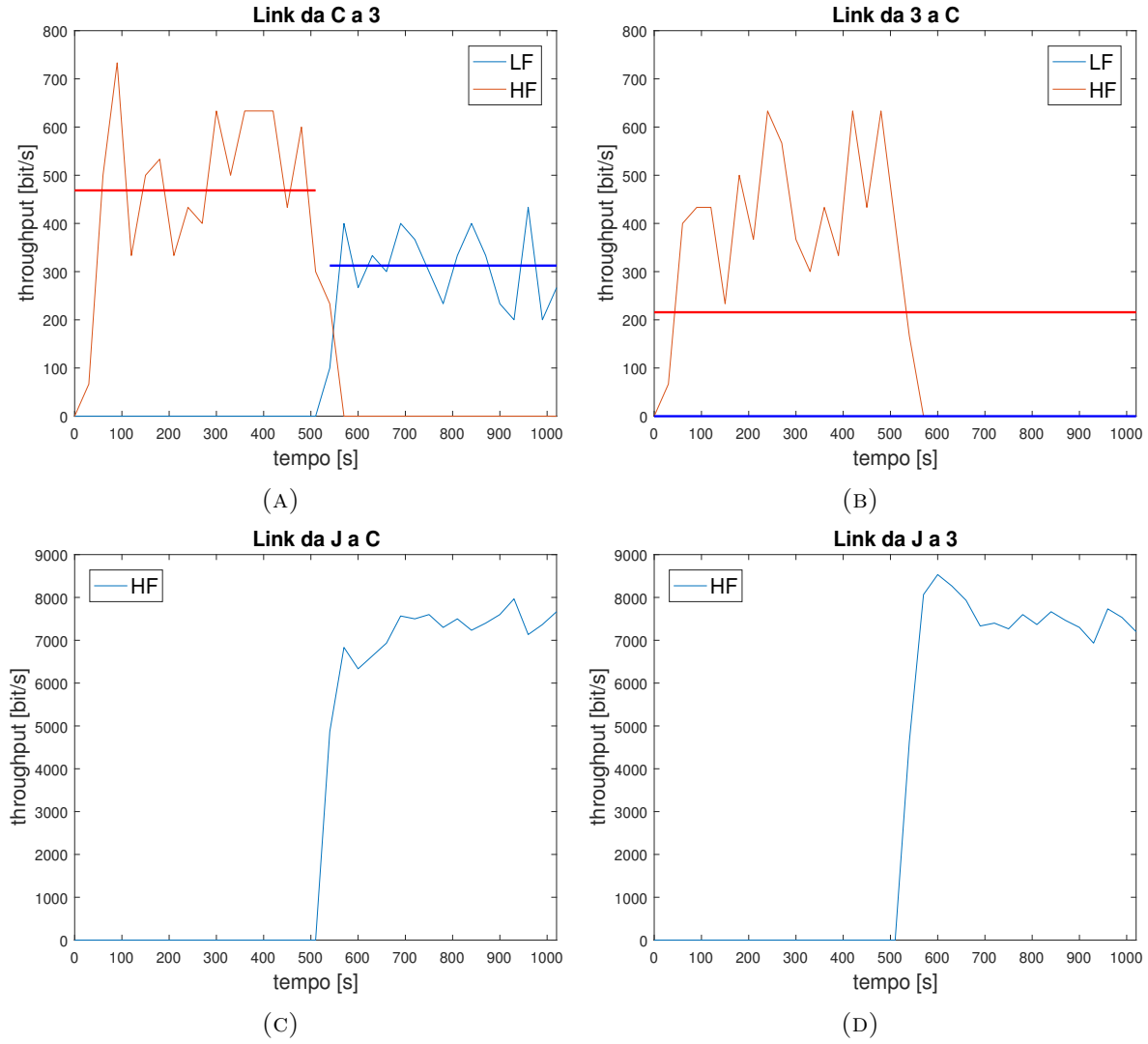


FIGURA 5.8: Jammer e multimodalità: prestazioni della rete.

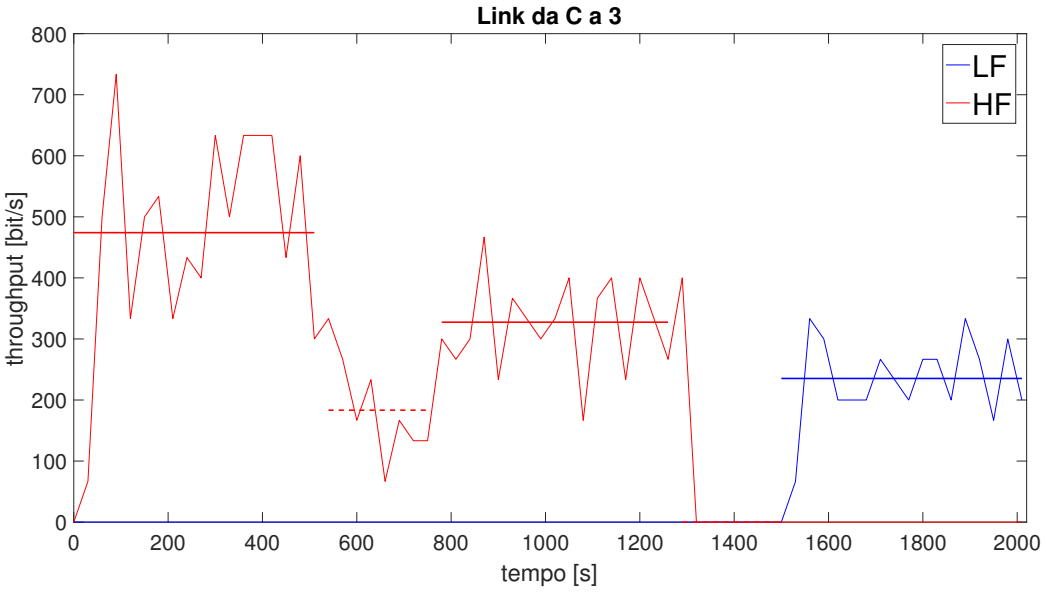
In questa simulazione sono di notevole importanza i valori di PDR ottenuti, nella prima parte della simulazione entrambi i nodi considerati (3 e C) hanno PDR superiori al 90%, questo valore rimane circa inalterato per il nodo C che cambia canale di trasmissione utilizzando l'interfaccia LF mentre crolla a 0% per il nodo 3 abbassando così il PDR totale al 46.12%. La Tabella 5.6 riassume tutti i PDR rilevanti della terza simulazione.

Link	**HF**	**LF**	**Link inverso**	**HF**	**LF**
3 → C	46.12%	-	C → 3	96.56%	95.63%
J → C	36.74%	-	C → J	-	-
J → 3	38.10%	-	3 → J	-	-

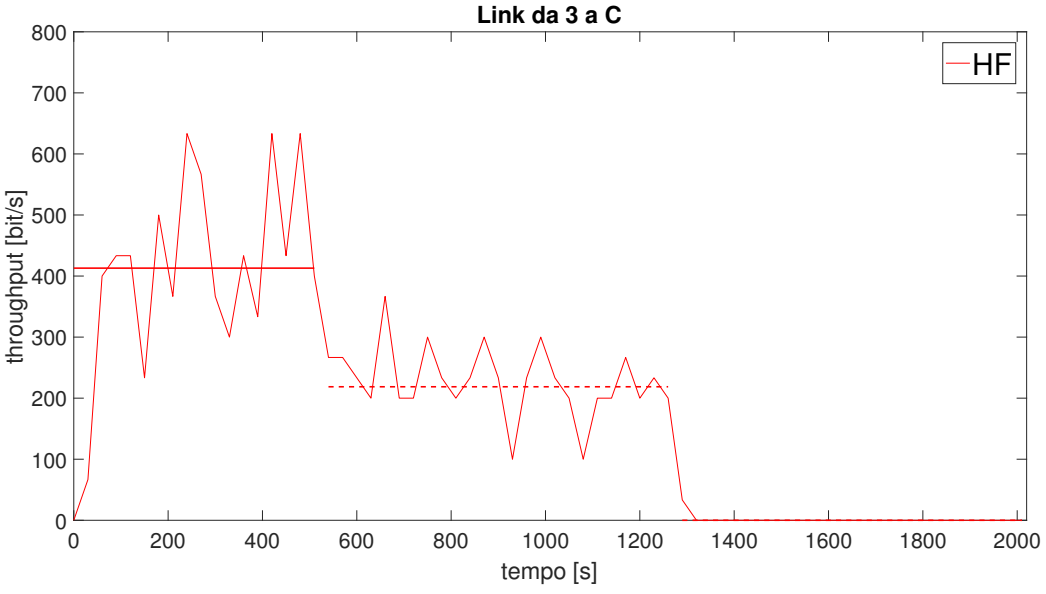
TABELLA 5.6: Jammer e multimodalità: Packet Delivery Ratio della rete.

5.5 Scenario completo

La quarta e ultima simulazione racchiude tutte le casistiche precedentemente considerate, utilizzando sia multimodalità che adattività come risposta all'uso di un jammer sulle frequenze HF. Questa simulazione vuole essere quanto più simile ad uno scenario reale, quindi i nodi impiegheranno del tempo per accorgersi del cambiamento e agire di conseguenza, il throughput medio in queste fasi di transizione è evidenziato da una linea tratteggiata orizzontale. La topologia della rete è analoga a quella della seconda simulazione, inizialmente i nodi trasmettono senza interferenze da parte del nodo J, dopo 500 s il jammer si attiva con potenza ridotta (i parametri sono uguali alla simulazione sull'adattività), inevitabilmente i nodi 3 e C iniziano a diminuire il throughput. Dopo altri 250 s il nodo C decide di adattarsi cambiando modulazione e potenza come definito nella seconda simulazione mentre il nodo 3 continua a trasmettere senza cambiare settaggi. All'istante 1250 s il jammer aumenta la potenza e diminuisce il periodo CBR utilizzando i parametri definiti nella terza simulazione, di conseguenza i nodi C e 3 non riescono più a trasmettere alcun pacchetto. Dopo 250 s il nodo C decide di cambiare interfaccia fisica e comincia a trasmettere messaggi al nodo 3 con frequenza LF. In Figura 5.9 si possono notare questi cambiamenti, il nodo 3 subisce passivamente le mutazioni del canale di trasmissione ottenendo una riduzione del throughput inevitabile fino a bloccare tutte le comunicazioni quando il jammer utilizza la massima potenza. Al contrario il nodo C cerca di contrastare questi cambiamenti utilizzando l'adattività quando possibile e ripiegando sulla multimodalità quando il canale è inutilizzabile. Questi grafici dimostrano chiaramente quanto queste tecnologie possano aiutare ad ottimizzare le trasmissioni subacquee. Con DESERT è ora possibile simulare scenari di questo tipo e testare protocolli adattivi e multimodali.



(A)



(B)

FIGURA 5.9: Scenario completo: prestazioni della rete.

Capitolo 6

Conclusioni

In questa tesi sono state analizzate, implementate e simulate reti subacquee multimodali adattive, in aggiunta è stato implementato un dispositivo jammer che può anche essere utilizzato come fonte di rumore in reti sottomarine. Il capitolo 4 mostra tutte le nuove librerie create che estendono il simulatore DESERT, permettendo di simulare scenari più generali con dispositivi che reagiscono attivamente ai cambiamenti delle condizioni del canale di trasmissione. L'adattività e la multimodalità saranno fondamentali in futuro per ottimizzare reti subacquee ed è importante per i ricercatori avere a disposizione un simulatore semplice e completo. Un esempio pratico di possibile sviluppo futuro potrebbe essere la creazione di un un modulo multimodale che rispetto la diagnostica di canale (errori, rumore, throughput) decida di adattare la rete al meglio. All'interno della tesi sono stati descritti in modo particolareggiato i comandi disponibili e sono stati forniti degli esempi consultabili su GitHub. Il capitolo 5 fornisce alcuni scenari significativi in cui viene dimostrata l'effettiva importanza delle tecniche studiate ma i campi di utilizzo di adattività e multimodalità sono molto più ampi. Un esempio può essere l'uso di queste tecnologie per suddividere il traffico in base alle QoS richieste da una specifica tipologia di pacchetto e garantire determinate prestazioni in base a questa suddivisione. Una possibile estensione di questo lavoro sarebbe quella di continuare a definire comandi adattabili per ampliare le possibilità del simulatore DESERT.

Bibliografia

[1] N. Benvenuto and M. Zorzi, *Principles of communications networks and systems*. Chichester, UK: Wiley, 2011.

[2] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT Underwater: An NS-Miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols," in *2012 Oceans - Yeosu*, pp. 1–10, May 2012.

[3] N. Baldo, M. Miozzo, F. Guerra, M. Rossi, and M. Zorzi, "Miracle: The Multi-Interface Cross-Layer Extension of NS2," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, Mar 2010.

[4] A. Smerdon, F. Bustamante, and M. Baker, "The SWIG acoustic standard: An acoustic communication standard for the offshore energy community," in *2016 IEEE Third Underwater Communications and Networking Conference (UComms 2016)*, (Lerici, Italy), pp. 1–4, Aug 2016.

[5] J. Potter, J. Alves, D. Green, G. Zappa, K. McCoy, and I. Nissen, "The JANUS underwater communications standard," in *2014 Underwater Communications and Networking (UComms 2014)*, Sep 2014.

[6] F. Campagnaro, R. Francescon, P. Casari, R. Diamant, and M. Zorzi, "Multimodal Underwater Networks: Recent Advances and a Look Ahead," in *The International Conference on UnderWater Networks and Systems (WUWNet '17)*, (Halifax, NS, Canada), Nov 2017.

[7] X. Lurton, *An Introduction to Underwater Acoustics Principles and Applications*. Springer-Verlag Berlin Heidelberg, 2010.

[8] R. Urick, *Principles of Underwater Sound*. New York: McGraw-Hill, 1983.

[9] M. Stojanovic, "On the Relationship Between Capacity and Distance in an Underwater Acoustic Channel," in *The International Conference on UnderWater Networks and Systems (WUWNet '06)*, (Los Angeles, California, USA), Sep 2006.

[10] R. Coates, *Underwater Acoustic Systems*. New York: Wiley, 1989.

[11] F. Campagnaro, R. Francescon, F. Guerra, F. Favaro, P. Casari, R. Diamant, and M. Zorzi, "The DESERT underwater framework v2: Improved capabilities and extension tools," in *2016 IEEE Third Underwater Communications and Networking Conference (UComms 2016)*, (Lerici, Italy), pp. 1–5, Aug 2016.

[12] P. Casari, C. Tapparello, F. Guerra, F. Favaro, I. Calabrese, G. Toso, S. Azad, R. Masiero, and M. Zorzi, "Open source suites for underwater networking: WOSS and DESERT Underwater," *IEEE Network*, vol. 28, pp. 38–46, Sep 2014.

[13] "NS2 Main Page." `http://nsnam.sourceforge.net/wiki/index.php/Main_Page`.

[14] "oTcl Project Page." `http://otcl-tclcl.sourceforge.net/otcl/`.

[15] "SIGNET Main Page." `http://signet.dei.unipd.it/`.

[16] "NS-MIRACLE Library: Multi InteRfAce Cross Layer Extension for NS." `http://telecom.dei.unipd.it/ns/miracle/doxygen/`.

[17] "DESERT Underwater Main Page." `http://desert-underwater.dei.unipd.it/`.

[18] "DESERT Underwater GitHub." `https://github.com/uwsignet/DESERT_Underwater`.

[19] M. Goetzl and I. Nissen, "GUWMANET-multicast routing in underwater acoustic networks," in *2012 Military Communications and Information Systems Conference (MCC 2012)*, pp. 1–8, Gen 2012.

[20] "EvoLogics S2C R 48/78 Underwater Acoustic Modem." https://www.evologics.de/en/products/acoustics/s2cr_48_78.html.

[21] "EvoLogics S2CR 18/34 Underwater Acoustic Modem." https://www.evologics.de/en/products/acoustics/s2cr_18_34.html.

[22] J. Kalwa, "The RACUN-project: Robust Acoustic Communications in Underwater Networks - an Overview," in *OCEANS 2011 IEEE - Spain*, pp. 1–6, Jun 2011.