CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA DELLE TELECOMUNICAZIONI

STUDIO COMPARATIVO DI PROTOCOLLI DI BROADCAST BASATI SU CODICI FOUNTAIN IN RETI ACUSTICHE SOTTOMARINE

RELATORE: Ch.mo Prof. Michele Zorzi

CORRELATORE: Dott. Ing. Paolo Casari

LAUREANDO: Federico Guerra

A.A. 2007-2008

UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA SPECIALISTICA
IN
INGEGNERIA DELLE TELECOMUNICAZIONI

STUDIO COMPARATIVO DI PROTOCOLLI DI BROADCAST BASATI SU CODICI FOUNTAIN IN RETI ACUSTICHE SOTTOMARINE

RELATORE: Ch.mo Prof. Michele Zorzi

CORRELATORE: Dott. Ing. Paolo Casari

LAUREANDO: *Federico Guerra*

Padova, 8 luglio 2008

Al mio grande Amore Silvia, a mia madre Claudia,
a mio padre Cosimo ed a sua moglie Ilaria, a mia sorella Laura.

Indice

Sommario **1**

1 Introduzione **2**

2 Il canale sottomarino acustico **4**

2.1 Introduzione 4
2.2 Metriche di canale 6
2.2.1 Il ritardo di propagazione 6
2.2.2 La dispersione del ritardo (*Delay-spread*) 6
2.2.3 Lo spostamento e la dispersione Doppler 6
2.2.4 Il fattore di dispersione del canale (*Channel spread factor*) 7
2.3 Attenuazione del segnale 7
2.3.1 Geometria della propagazione 7
2.3.2 Attenuazione per assorbimento 8
2.3.3 Attenuazione complessiva 9
2.4 Il *fading* 10
2.5 La tempo-varianza 10
2.5.1 La diffusione superficiale (*Surface Scattering*) 11
2.5.2 La diffusione delle bolle (*Bubble Scattering*) 12
2.6 Il rumore 13
2.7 SNR 15
2.8 L'ampiezza di banda e la capacità 16
2.8.1 Definizione empirica di ampiezza di banda 16
2.8.2 Una definizione di ampiezza di banda basata sulla capacità 17
2.9 Lo strato fisico in reti sottomarine 20
2.10 Lo strato *MAC* in reti sottomarine 22
2.10.1 Protocolli *CSMA* 23

2.10.2 Protocolli *CDMA* 23
2.10.3 Sviluppi futuri 24
2.11 Lo strato di *routing* in reti sottomarine 25
2.11.1 I protocolli proattivi 25
2.11.2 I protocolli reattivi 25
2.11.3 I protocolli geografici 26
2.11.4 Sviluppi futuri 26
2.12 Lo strato di trasporto in reti sottomarine 27
2.12.1 Sviluppi Futuri 28
2.13 Possibili Scenari 29
2.13.1 Reti acustiche in due dimensioni 29
2.13.2 Reti acustiche in tre dimensioni 30
2.13.3 Reti acustiche di mezzi mobili in tre dimensioni 31

3 I codici a fontana **33**
3.1 Introduzione 33
3.2 Il canale a cancellazione 33
3.3 La fontana lineare aleatoria 35
3.4 I codici LT 37
3.4.1 Il codificatore 37
3.4.2 Il decodificatore 38
3.4.3 Progettazione della distribuzione dei gradi 39

4 Il *broadcast* nelle reti di sensori *wireless* **43**
4.1 Introduzione 43
4.2 Caratteristiche comuni 44
4.2.1 Il *Jitter* 44
4.2.2 L'intervallo aleatorio di stima (RAD) 44
4.3 Categorizzazione dei protocolli 45
4.3.1 *Flooding* semplice 45
4.3.2 Protocolli probabilistici 46
4.3.3 Protocolli geografici 46
4.3.4 Protocolli ad identificazione dei vicini 47
4.4 Protocolli con codici a fontana 51
4.4.1 Il protocollo *CRBCast* 51

4.4.2 Il protocollo *FTS* 53

5 Broadcast nelle reti acustiche sottomarine 56

5.1 Introduzione 56

5.2 Ipotesi preliminari 57

5.2.1 Il canale sottomarino, lo strato fisico e *MAC* 57

5.2.2 Topologie di rete 58

5.3 Uso di codici a fontana per il *broadcast* 60

5.3.1 Descrizione del protocollo progettato 60

5.3.2 La creazione delle *policy* 62

5.3.3 Parametri di progetto dei codici a fontana 67

5.3.4 Statistiche Osservate 67

5.3.5 Obiettivi 69

5.4 Il protocollo CRBCast 69

5.4.1 L'ottimizzazione della probabilità di inoltro 69

5.4.2 La Fase I modificata 71

5.4.3 La Fase II modificata 71

5.4.4 Metriche osservate 71

5.4.5 Obiettivi 73

6 La simulazione 74

6.1 L'ambiente simulativo *NS2* 74

6.1.1 Schema di funzionamento 75

6.2 *NS2-MIRACLE* 75

6.2.1 La classe *Module* 76

6.2.2 Le classi *Cross-Layer Message* e *Node-Core* 76

6.2.3 La classe *Plugin* 77

6.3 Il canale sottomarino di *MIRACLE* 77

6.4 I moduli *MIRACLE* sviluppati 78

6.4.1 Il modulo dei codici a fontana 78

6.4.2 Il modulo di controllo del protocollo *multihop* 80

6.4.3 Il modulo di controllo *CRBCast* 81

7 Risultati 84

7.1 I codici a fontana in ambiente *multi hop* 85

7.1.1 L'efficienza del protocollo proposto 85
7.1.2 Il confronto tra *single hop* e *multi hop* 92
7.2 Il protocollo *CRBCast* 104
7.2.1 Il numero medio di pacchetti codificati trasmessi 104
7.2.2 La probabilità media di fallire un round di trasmissione . . 106
7.2.3 Le sessioni sprecate 108
7.2.4 Le efficienze di tramissione 110
7.2.5 I ritardi 112
7.2.6 La copertura della rete 114
7.3 Il confronto dei due protocolli 120
7.3.1 Il ritardo complessivo 120
7.3.2 L'efficienza totale 121
7.3.3 La *CDF* della copertura di rete 121

8 Conclusione **125**

A Ulteriori risultati **126**
A.1 I codici a fontana in ambiente *multi hop* 126
A.1.1 L'efficienza del protocollo proposto 126
A.1.2 Il confronto tra *single hop* e *multi hop* 131

Bibliografia **143**

Sommario

In questa tesi si affrontano alcune problematiche relative al broadcast affidabile ed efficiente nelle reti acustiche sottomarine. Scopo di questa tesi è quello di porre le basi per un protocollo di *ARQ Ibrido* (*HARQ*) basato sui codici a fontana per il broadcast nelle reti acustiche sottomarine, il protocollo *OFBP* (*Optimized Fountain-based Broadcast Protocol*). Esso prevede la collaborazione tra i nodi della rete per ottenere la massima affidabilità ed efficienza energetica.
Dopo un'analisi del canale sottomarino e della sua modellazione matematica ci si soffermerà in particolare sullo studio dei codici a fontana in ambito sia *single hop* sia *multi-hop* confrontandone le prestazioni con quelle di un altro protocollo di broadcast con simili obiettivi, il *CRBCast*. I risultati sul confronto tra *single-hop* e *multi-hop* dimostrano che, sotto determinate condizioni, le prestazioni nei due scenari sono confrontabili, e che dunque è possibile dedurre le metriche *multi-hop* da quelle *single-hop*. I risultati sul confronto tra *OFBP* e *CRBCast* dimostrano infine come *OFBP* sia più efficiente e performante del protocollo *CRBCast*.

Capitolo 1

Introduzione

La ricerca sulle reti acustiche di sensori sottomarine è ancora allo stadio embrionale. Molte sono le problematiche che devono essere affrontate e risolte per poter ottenere lo stesso stesso grado di efficienza delle reti radio tradizionali.
Per molti decenni l'approccio tradizionale alla ricerca sottomarina è stato quello di posizionare la strumentazione *in situ* per poi raccogliere i dati ottenuti al termine della missione. Questo approccio ha tuttavia ha numerosi svantaggi:

- **Assenza di controllo in tempo reale**. I dati raccolti non sono accessibili se non al termine dell'esperimento che in media può durare dei mesi.
- **Nessuna possibilità di riconfigurare i sistemi utilizzati**. Non è possibile interazione alcuna tra la stazione di controllo e gli strumenti utilizzati, cosa che impedisce eventuali ricofigurazioni dei sensori e la possibilità di condurre esperimenti adattivi
- **Assenza di rivelazione guasti**. In caso di guasti o configurazioni errate potrebbe non essere possibile la rivelazione di tali eventi se non alla fine dell'esperimento stesso.
- **Capacità di memoria limitata**. L'assenza di trasmissioni impone la memorizzazione di una grande mole di dati su dispositivi a bordo di capacità limitata.

L'utilizzo di comunicazioni *wireless* è quindi essenziale per lo svolgimento di qualsiasi attività sottomarina. Alcuni tentativi sono stati condotti utilizzando onde radio con risultati poco soddisfacenti, poichè le uniche onde capaci di propagarsi

in acqua salata hanno frequenze molto basse, dell'ordine di *30 - 300 Hz*, e richiedono antenne di grandi proporzioni e una potenza di trasmissione elevata. Si è successivamente tentato con le comunicazioni con mezzi ottici che seppur non limitate dall'attenuazione marina soffrono tuttavia l'elevata dispersione del mezzo, tanto da limitarne l'uso su brevissime distanze. Le comunicazioni acustiche rimangono a tutt'oggi l'unico metodo efficace per comunicare su lunghe distanze. Lo sviluppo e lo studio di sistemi trasmissivi acustici e di protocolli flessibili è quindi un elemento centrale della ricerca attuale e futura, che tiene in considerazione di molteplici aspetti, di cui si ricordano i seguenti:

- La larghezza di banda in trasmissione è severamente limitata.
- Il canale sottomarino estremamente variabile a causa di effetti *fading* e *multipath*.
- Il ritardo di propagazione maggiore di cinque ordini di grandezza rispetto a quello delle onde radio in aria.
- La durata delle batterie dei sensori è limitata.
- Si possono verificare frequenti rotture a causa delle elevate pressioni e della corrosione dovuta all'acqua salata.

Alla luce di queste considerazioni, i protocolli di *broadcast* rivestiranno un ruolo essenziale per le future applicazioni di analisi e raccolta campioni, monitoraggio del territorio, esplorazioni sottomarine, prevenzione di catastrofi naturali, navigazione assistita, sorveglianza tattica del territorio etc. Tali protocolli devono essere affidabili nella trasmissione dati ed efficienti nella gestione delle risorse energetiche e nel minimizzare il ritardo di disseminazione dell'informazione. I codici a fontana dunque sono un buon candidato per la realizzazione di tali obiettivi.
Dopo una descrizione del canale sottomarino e dei suoi modelli matematici correntemente accettati, si introdurranno i codici a fontana e li si contestualizzerà nell'ambito dei protocolli di *broadcast*.
Successivamente si presenteranno il protocollo *OFBP*[1] ed il protocollo *CRBCast* e se ne descriverà in dettaglio l'implementazione nel simulatore di rete utilizzato. Si esporranno e si analizzeranno infine i risultati ottenuti confrontando le prestazioni dei due protocolli sopracitati.

[1] Optimized Fountain Broadcast

Capitolo 2

Il canale sottomarino acustico

2.1 Introduzione

L'esigenza di sistemi di trasmissione sempre più complessi, ha richiesto una modellizzazione sempre più accurata del canale sottomarino acustico e delle sue principali caratteristiche, tra le quali *fading*, rumore, effetti *multi-path*, *Doppler spread*. Tali fattori determinano la variazione spazio-temporale del canale, e rendono la banda fortemente dipendente da distanza e frequenza di trasmissione.
Alla luce di queste considerazioni si possono classificare i vari canali in base alla distanza, alla direzione di propagazione ed alla profondità:

Denominazione	distanza [Km]	Larghezza di banda [Hz]
Molto Lunga	1000	<1
Lunga	10-100	2-5
Media	1-10	≈10
Breve	0.1-1	20-50
Molto breve	<0.1	>100

Figura 2.1: Classificazioni dei canali in base alla distanza

PROPAGAZIONE	DESCRIZIONE	CARATTERISTICHE
Verticale	L'onda sonora si propaga in verticale, salendo o scendendo di profondità lungo il cammino. Tale modalità viene tipicamente utilizzata nelle comunicazioni tra sensori e *sink*.	Le distorsioni dovute a *fading* da cammini multipli e da *scattering* sono meno pronunciate.
Orizzontale	si verifica quando la comunicazione è parallela al fondale.	La dispersione temporale del segnale è elevata.

Figura 2.2: Classificazioni dei canali in base alla direzione di propagazione

PROFONDITÀ	DESCRIZIONE	CARATTERISTICHE
Acque profonde (*deep water*)	La trasmissione avviene in acque profonde oltre le centinaia di metri. Tipicamente si definiscono tali le acque oltre la piattaforma continentale.	La geometria di propagazione è quasi sferica, le riflessioni sono poco frequenti.
Acque poco profonde (*shallow water*)	La trasmissione avviene in acque profonde meno di un centinaio di metri.	La geometria di propagazione è cilindrica. Il segnale è fortemente attenuato da *scattering*, *fading* da cammini multipli.

Figura 2.3: Classificazioni dei canali in base alla profondità

Si procede ora alla descrizione dei processi stocastici che caratterizzano il canale stesso.

2.2 Metriche di canale

2.2.1 Il ritardo di propagazione

Una caratteristica del canale acustico sottomarino è la sua velocità di propagazione $c \approx 1500 \left[\frac{m}{s}\right]$ che è circa cinque ordini di grandezza inferiore rispetto a quella delle onde elettromagnetiche nel canale radio $c \approx 3 \cdot 10^8 \left[\frac{m}{s}\right]$. Questo induce dei tempi di propagazione molto elevati e non più trascurabili come nel caso di trasmissioni radio. Tale ritardo è un altro elemento limitante per la progettazione di protocolli e di sistemi di trasmissione efficienti.

2.2.2 La dispersione del ritardo (*Delay-spread*)

Una statistica essenziale del canale è il suo *delay-spread*, l'intervallo temporale tra la prima rilevazione di un segnale e l'ultimo arrivo dell'ultima eco dovuta alle riflessioni ambientali. Tale intervallo può essere grande in relazione al tempo di trasmissione di un simbolo, a volte può coprire centinaia di tempi di simbolo di trasmissione.

2.2.3 Lo spostamento e la dispersione Doppler

Ulteriore elemento da considerare è lo spostamento Doppler :

$$f_d = f_0 \frac{v}{c} \tag{2.1}$$

dove f_0 è la frequenza originale del segnale trasmesso e v è la velocità relativa tra trasmettitore e ricevitore. A causa della bassa velocità di propagazione dell'onda acustica anche una minima velocità relativa può dare un grande spostamento doppler e quindi distorsione in ricezione.
Se un segnale subisce riflessioni, ciascuna eco ricevuta avrà in generale uno spostamento Doppler differente: il *Doppler spread* o dispersione Doppler, è l'estensione spettrale (nell'intorno della frequenza centrale f_0) contenente tutti gli spostamenti Doppler relativi ai cammini multipli ricevuti.
Un'altra statistica molto importante è la dispersione Doppler normalizzata al T_s tempo di simbolo in trasmissione

$$\bar{B}_w = B_w T \tag{2.2}$$

che dà nozione della dispersione causata dalla dispersione Doppler sul sistema di trasmissione utilizzato. Ad una minore $\bar{B}_w$ corrisponde una maggiore resistenza alla dispersione stessa. Si noti come sia possibile diminuire tale prodotto sia aumentanto il rate di trasmissione sia tramite metodi noti per combattere la distorsione indotta dal canale, come ad esempio l'equalizzazione.

2.2.4 Il fattore di dispersione del canale (*Channel spread factor*)

Tale valore è un numero reale che ha lo scopo di dare un valore riassuntivo di dispersione totale del canale. Esso è dato dal prodotto della dispersione Doppler per la dispersione del ritardo.

2.3 Attenuazione del segnale

2.3.1 Geometria della propagazione

Come per le onde elettromagnetiche, le onde acustiche disperdono la propria energia su una superficie che cresce all'aumentare della distanza percorsa. A distanze relativamente brevi tale superficie può essere rappresentata da una sfera, che provoca un decadimento della potenza ricevuta dell'ordine di R^{-2}, dove R è la distanza dalla sorgente.
Tuttavia, ad una certa distanza dalla sorgente le riflessioni dovute alla superficie ed al fondale marino impongono una geometria di propagazione cilindrica. Nella regione di propagazione cilindrica il decadimento della potenza è dunque proporzionale a R^{-1}. Tale transizione avviene tipicamente a distanze ben maggiori della profondità del mezzo.
A tale proposito, è utile definire il parametro k chiamato *spreading factor*:

$$k = \begin{cases} 1 & \text{prop. cilindrica} \\ 1.5 & \textit{practical spreading} \\ 2 & \text{prop. sferica} \end{cases}$$

dove $k = 1.5$ è detto *practical spreading* in quanto è il valore comunemente utilizzato per scenari di interesse pratico.
La differenza con il canale radio è evidente in quanto in quest'ultimo i tipici valori

di k variano da 2 a 4, rappresentanti rispettivamente il caso di propagazione in spazio libero ed lil caso *two-ray ground reflection* (in cui si assume vengano ricevuti due segnali sovrapposti, quello diretto e quello riflesso dal terreno).

2.3.2 Attenuazione per assorbimento

Un secondo meccanismo di perdita di segnale è dovuto alla conversione dell'engergia in calore. Tale assorbimento è fortemente dipendendente dalla frequenza. Esso è indicato con $a(f)$, come descritto nelle seguenti formule empiriche dovute a *Thorp*, dove la frequenza f è espressa in $[kHz]$ e $a(f)$ è indicato in $\left[\frac{dB}{Km}\right]$:

$$\begin{aligned} 10\log a(f) &= 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f} + 2.75\cdot 10^{-4} f^2 + 0.003 \; f > 100Hz \\ 10\log a(f) &= 0.002 + 0.11\frac{f^2}{1+f^2} + 0.011 f^2 \; f < 100Hz \end{aligned}$$

Come si vede nella figura seguente il fattore di assorbimento aumenta rapidamente al crescere della frequenza. Se a brevi distanze dalla sorgente l'attenuazione di propagazione è dominante, a distanze maggiori il fattore d'assorbimento è l'elemento che maggiormente la potenza ricevuta.

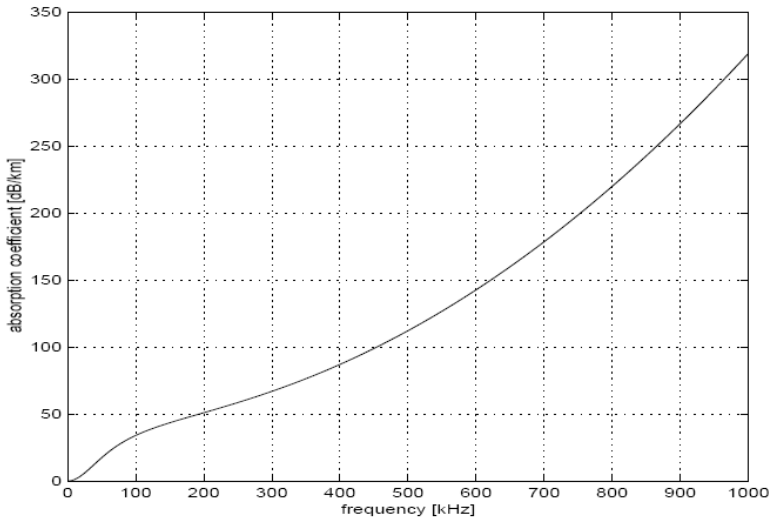

Figura 2.4: Coefficiente d'assorbimento, $a(f)$ $\left[\frac{dB}{Km}\right]$

2.3.3 Attenuazione complessiva

L'attenuazione totale o *path loss* che subisce un segnale acustico ad una distanza l e ad una frequenza f è data da:

$$A(l, f) = l^k a(f)^l \tag{2.3}$$

dove k e $a(f)$ sono il fattore di *spreading* ed il fattore di assorbimento definiti nelle sotto-sezioni precedenti. Convertendo $A(l, f)$ in dB, la formula precedente può essere riscritta come:

$$10 \log A(l, f) k \cdot 10 \log l + l \cdot 10 \log a(f) \tag{2.4}$$

$A(l, f)$ definisce l'attenuazione per un singolo cammino privo di ostacoli. Se un tono di frequenza f e potenza P è trasmesso lungo tale cammino, la potenza ricevuta è data da

$$P_{Rx} = \frac{P}{A(l, f)} \tag{2.5}$$

Se invece sono presenti più cammini distinti, ciascuno di lunghezza l_p, $p = 0, ..., P-1$, la funzione di trasferimento del canale è la sequente:

$$H(l, f) = \sum_{p=0}^{P-1} \frac{\Gamma_p}{\sqrt{A(l_p, f)}} e^{-j2\pi f \tau_p} \tag{2.6}$$

dove l_0 è la distanza tra la sorgente ed il ricevitore, Γ_p modella tutte le attenuazioni del cammino p-simo e $\tau_p = \frac{l_p}{c}$ è il ritardo del cammino p-simo, dove c è la velocità di propagazione, con $c = 1500 \left[\frac{Km}{s}\right]$.
Se la trasmissione non è direzionale,cioè se i cammini p-simi hanno cioè energia comparabile o superiore al cammino diretto, allora la potenza ricevuta è:

$$P_{Rx} = P|H(l, f)|^2 \tag{2.7}$$

2.4 Il *fading*

Come descritto nelle sezione precedente, l'oceano può essere modellato alle frequenze d'interesse come una guida d'onda avente come pareti la superficie e il fondo oceanico. Tali supercie di discontinuità sono causa di riflessioni e rifrazioni e quindi di *fading* da cammini multipli che causa delle macro-variazioni nella risposta del canale stesso; in acque poco profonde (*shallow water*) tale processo è molto pronunciato.
In acque profonde al contrario tale processo è poco presente; in quest'ultimo caso infatti la guida d'onda si forma grazie alla propensione delle onde acustiche a curvare verso le zone a di rifrazione inferiore: tale regione è chiamata *asse del canale acustico profondo* e sulle sue superfici di discontinuità si creano le riflessioni e rifrazioni che danno origine al *fading* da cammini multipli [2].
A causa di quest'ultimo effetto si possono ottenere delle deviazioni verso l'alto che possono raggiungere la superficie dell'oceano focalizzandosi nel punto dove esse sono riflesse: tale processo è periodico. La regione di spazio tra due punti di fuoco è detta *Zona di Convergenza* e la sua lunghezza tipica è dell'ordine dei $60Km - 100Km$.
La caratteristica essenziale del *multi-path* è la sua dipendenza spaziale, cosa che può essere adeguatamente sfruttata in ricezione, per combattere gli effetti dell' *ISI* (*Inter Symbol Interference*) dovuti al canale.
Un'ulteriore caratteristica del multipath, è la formazione di *Zone d'ombra* zone in cui la somma vettoriale delle onde dei singoli cammini è fortemente distruttiva. In acque profonde, esse hanno la tendenza a presentarsi a profondità maggiori di $1000m$ e a distanze tipiche dell'ordine delle decine di Km. In acque poco profonde tale evento tende a verificarsi a profondità dell'ordine dei $100m$ e a distanze dell'ordine dei $3Km$.

2.5 La tempo-varianza

Oltre agli effetti macroscopici descritti nelle sezioni precedenti, sono presenti ulteriori micro-fluttuazioni della risposta impulsiva del canale. Tali processi sono causati, come i precedenti, della formazione di riflessioni multiple rapidamente tempo-varianti.

2.5.1 La diffusione superficiale (*Surface Scattering*)

La diffusione superficiale è causata dalla turbolenza della superficie stessa. Se l'oceano fosse calmo, un segnale incidente sulla sua superficie verrebbe riflesso quasi perfettamente e la sua unica distorsione sarebbe uno spostamento di fase pari a π. La presenza di onde causate dal vento genera tuttavia numerose riflessioni , che si traducono in dispersione del segnale, in dispersione Doppler (*Doppler spread*). Tali irregolarità della superficie ondosa dell'acqua possono essere modellate come un processo Gaussiano a media nulla, la cui densità spettrale di potenza è completamente caratterizzata dalla velocità del vento. Tuttavia un'analisi più semplice è possibile per acque poco profonde (*shallow water*) [6].
Sia d la distanza tra sorgente e ricevitore, e siano h_1 e h_2 le rispettive profondità. Le onde generate dal vento causano variazioni nell'altezza del punto di riflessione $h_{1,2}(t) = h_{1,2} + \Delta h(t)$; quest'ultime a loro volta causano variazioni temporali nella lunghezza del cammino riflesso

$$l(t) \approx l_0 + 2\Delta h(t)\cos\theta_0 = l_0 + \Delta l(t) \tag{2.8}$$

dove l_0 è la lunghezza nominale del cammino, θ_0 è l'angolo nominale d'incidenza e si è assunto inoltre che la massima variazione d'altezza della superficie sia $\Delta h_m \ll h_{1,2}$. A causa della variabilità della lunghezza del cammino, la componente sinusoidale $A\cos\omega t$ appare al ricevitore modulata in fase da una forma d'onda

$$\Delta\phi(t) = \frac{\omega}{c}\Delta l(t) \tag{2.9}$$

Data la massima deviazione, o l'indice di modulazione $\Delta\phi_m$ della fase $\Delta\phi(t)$ e la frequenza delle onde f_w, la banda risultante del segnale modulato in fase ricevuto sarà

$$B_w = 2f_w(1 + \Delta\phi_m) \tag{2.10}$$

In termini di altezza r.m.s. delle onde, la dispersione Doppler stimata causata dalle onde superficiali può essere espressa come

$$B_w = 2f_w\left(1 + \frac{2\omega\cos\phi_0}{c}h_w\right) \tag{2.11}$$

Utilizzando la velocità del vento w in $\left[\frac{m}{s}\right]$ e come valori sperimentali

$$f_w = \frac{2}{w} \tag{2.12}$$

$$h_w = 0.005w^{2.5} \tag{2.13}$$

Si può ottenere infine il valore della dispersione doppler B_w e successivamente il valore della dispersione Doppler normalizzata $\bar{B}_w$.
A titolo d'esempio, si consideri la sorgente ed il ricevitore alla stessa profondità $h_0 = 10m$, alla distanza $r = 500m$ ed una frequenza portante $f_0 = 15kHz$. Con una velocità del vento moderata $w = 20\frac{m}{s}$ si ottiene una dispersione Doppler per f_0 pari a $B_w = 10Hz$. Se si trasmettesse su tale canale ad un rate di $\frac{1}{T_s} = 1kHz$, e se fossero presenti venti più forti, si avrebbe una dispersione Doppler normalizzata $\bar{B}_w \approx 10^{-2}$ che è considerato come il limite per il *tracking* del canale con modulazione di tipo coerente. Da questa analisi è possibile notare come la dispersione Doppler sia più pronunciata in acque poco profonde e su trasmissioni a corto raggio, dove le frequenze utilizzate sono più elevate. Per distanze maggiori la dispersione Doppler è minore, tuttavia è predominante la componente dispersiva dovuta al *multi-path*. Nonostante questo si può affermare che il fattore di dispersione del canale decresce macroscopicamente con la distanza.

2.5.2 La diffusione delle bolle (*Bubble Scattering*)

La formazione di bolle dovuta al frangersi delle onde sulla superficie marina può avere un grande influenza sulla propagazione acustica alle alte frequenze sia in acque poco profonde che in quelle profonde. Strati di bolle nei pressi della superficie si traducono spesso in un'attenuazione significativa dei segnali ad alta frequenza riflessi dalla superficie, nell'ordine dei $3dB$ per ogni interazione successiva con la superficie stessa. Studi recenti hanno quantificato meglio la relazione fra la densità di bolle e l'entità dell'attenuazione per diffusione per una singola interazione con la superficie.
Per una velocità del vento $w \leq 6\frac{m}{s}$ tale effetto è trascurabile. Per valori superiori, l'attenuazione è crescente con la velocità stessa fino ad arrivare ad una perdita totale di segnale (attenuazione $\approx 20dB$) per velocità del vento $w \approx 10\frac{m}{s}$.
La penetrazione improvvisa di "sciami" di bolle nella colonna d'acqua è un altro evento che provoca una forte attenuazione del segnale, con dei rate a volte anche dell'ordine dei $26\frac{dB}{m}$, provocando sensibili interruzioni del segnale ricevuto.

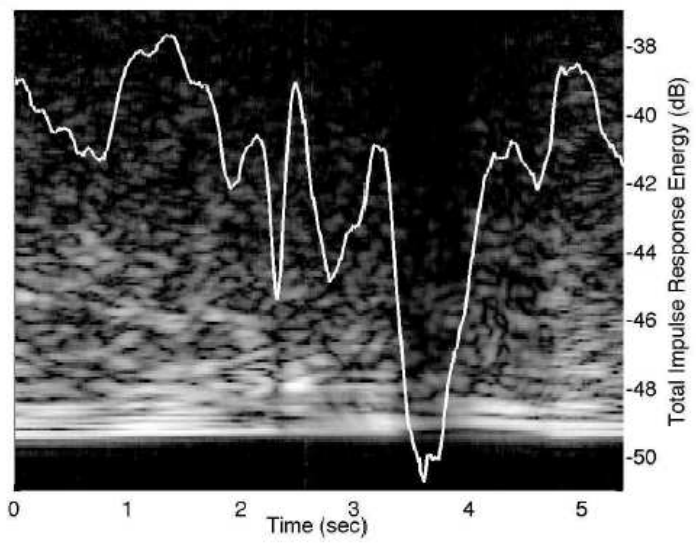

Figura 2.5: Esempio di *outage* nel segnale ricevuto.

2.6 Il rumore

Il rumore può essere classificato in rumore d'ambiente (*ambient noise*), ed in rumore prodotto dall'uomo (*man-made noise*). Il primo è riferito ai movimenti idro-dinamici (movimenti dell'acqua, tra cui correnti, tempeste, piogge, venti etc.) a movimenti sismici ed a eventi biologici, mentre il secondo è si riferisce alla navigazione ed altri elementi quali pompe, impianti idro-elettrici etc.
Il rumore, ai fini statistici, può tuttavia essere descritto da quattro eventi principali: dalla turbolenza, dalle onde, dal rumore termico, e dalla navigazione; ciascuno di queste componenti può essere modellata da un processo Gaussiano a densità spettrale di potenza continua. Si presentano qui quattro formule empiriche per le densità spettrale di potenza in $[dB\ re\ \mu Pa]$ dei sopracitati tipi di rumore, in funzione della frequenza in $[kHz]$

$$10\log N_t(f) = 17 - 30\log f \tag{2.14}$$

$$10\log N_s(f) = 40 + 20(s-0.5) + 26\log f - 60\log(f+0.03) \tag{2.15}$$

$$10\log N_w(f) = 50 + 7.5w^{\frac{1}{2}} + 20\log f - 40\log(f+0.4) \tag{2.16}$$

$$10\log N_{th}(f) = -15 + 20\log f \tag{2.17}$$

Il rumore di turbolenza $N_t(f)$ interessa soltanto le basse frequenze $f < 10Hz$, mentre il rumore dovuto alla navigazione $N_s(f)$ è dominante nella regione $10Hz <$

$f < 100Hz$ ed è governato dal fattore di navigazione (*shipping factor*) s, i cui valori vanno da 0, o assenza di navigazione, a 1, o alta densità di navigazione. Il movimento superficiale dovuto alle onde causate dal vento è il fattore dominante nella regione di spettro prevalentemente utilizzata dai sistemi di trasmissione, $100Hz < f < 100kHz$; il rumore termico diviene invece elevato nella regione $f > 100kHz$.
Nella figura seguente è raffigurata la densità spettrale di potenza totale $N(f) = N_t(f) + N_s(f) + N_w(f) + N_{th}(f)$, in assenza di venti (linea continua), con venti di moderata entità, $w \approx 10\frac{m}{s}$ (linea a puntini) e per diversi valori del fattore di navigazione s. Il rumore decade con la frequenza, limitando difatto inferiormente l'ampiezza di banda utile. Si nota infine in una certa regione spettrale la densità di potenza del rumore decade in modo logaritmico secondo la formula:

$$10 \log N(f) \approx N_1 - \eta \log f \tag{2.18}$$

L'approssimazione è mostrata in figura (linea a trattini e puntini) per $N_1 = 50\ dB\ re\ Pa$ e per $\eta = 18\frac{dB\ re\ \mu Pa}{decade}$

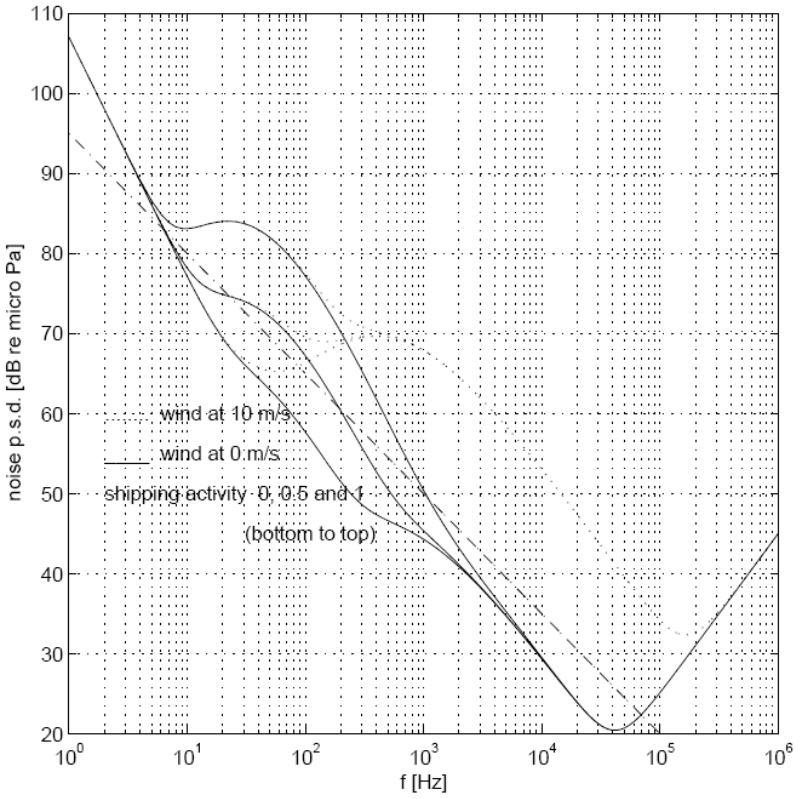

Figura 2.6: Densità spettrale di potenza del rumore.

2.7 SNR

Grazie alle definizioni di $A(l, f)$ e di $N(f)$ è possibile valutare l'*SNR* (*Signal to Noise Ratio*, anche detto rapporto segnale-rumore) al ricevitore ad una distanza l e per un tono in trasmissione alla frequenza f e avente potenza P. L'SNR in ipotesi di banda stretta può essere calcolato come

$$SNR(l, f) = \frac{P/A(l, f)}{N(f)\Delta f} \tag{2.19}$$

dove Δf è la larghezza di banda del rumore al ricevitore (in ipotesi di banda stretta nell'intorno della frequenza centrale f). Il prodotto $A(l, f)N(f)$ determina la dipendenza dalla frequenza dello SNR. Il fattore $1/\left(A(l, f)N(f)\right)$ è illustrato nella Figura 2.7 .

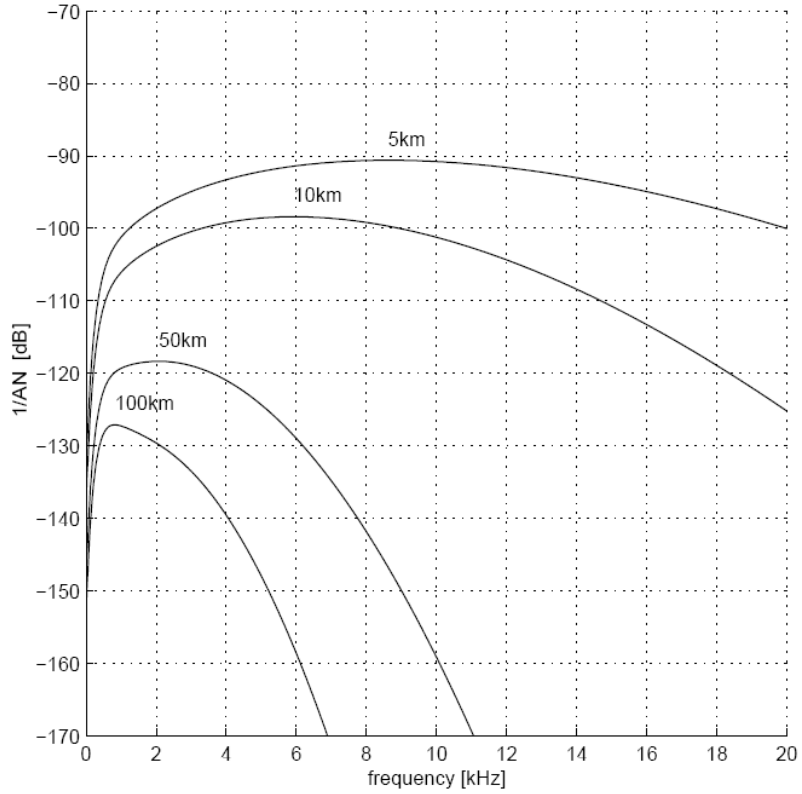

Figura 2.7: La componente dell'SNR dipendente dalla frequenza.

Per ogni distanza di trasmissione l è possibile notare l'esistenza di una frequenza ottima $f_o(l)$ per la quale si ottiene il massimo SNR a banda stretta. L'andamento di tale frequenza ottima è evidenziato nella Figura 2.8, in funzione della distanza di trasmissione l.

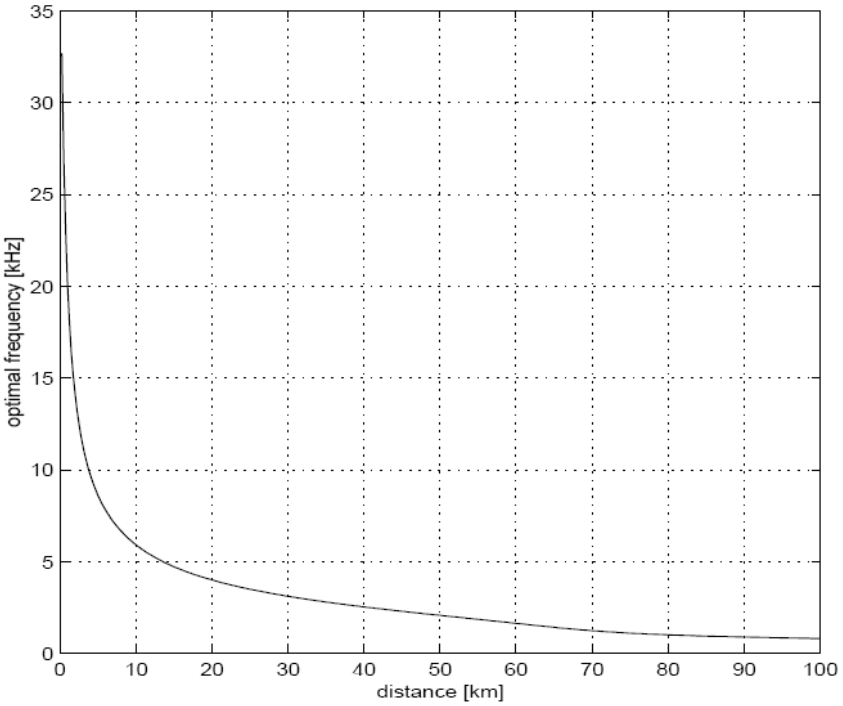

Figura 2.8: la frequenza ottima $f_o(l)$.

Appare evidente come sia possibile, data l, scegliere la frequenza $f_o(l)$ corrispondente e aggiustare la potenza di trasmissione per raggiungere il livello di SNR desiderato.

2.8 L'ampiezza di banda e la capacità

2.8.1 Definizione empirica di ampiezza di banda

Una possibile definizione di ampiezza di banda è quella ai $3dB$. Si definisce ampiezza di banda ai $3dB$ $B_3(l)$ come il *range* di frequenza nell'intorno della frequenza centrale ottima $f_0(l)$ per la quale $SNR(l, f) > SNR(l, f_0(l))/2$, in altri termini tale per cui sia $A(l, f)N(f) < 2A(l, f_0(l))N(f_0(l)) = 2AN_{min}(l)$.
Dopo aver settato l'ampiezza di banda $B(l) = [f_{min}(l), f_{max}(l)]$ nell'intorno di $f_0(l)$, la potenza di trasmissione $P(l)$ può essere settata per ottenere il livello di SNR in banda stretta desiderato. In alternativa è possibile settare la potenza in trasmissione in accordo con il livello di SNR corrispondente alla banda $B(l)$. Se si indica con $S_l(f)$ la densità spettrale di potenza del segnale trasmesso alla

distanza l, allora la potenza trasmessa può essere espressa come

$$P(l) = \int_{B(l)} S_l(f)df \tag{2.20}$$

e l' SNR come

$$SNR(l, B(l)) = \frac{\int_{B(l)} S_l(f)A^{-1}(l, f)df}{\int_{B(l)} N(f)df} \tag{2.21}$$

In quest'ultima definizione, sia l' SNR sia $P(l)$ dipendono dalla densità spettrale di potenza del segnale trasmesso. Nel caso più semplice, tale densità è piatta $S_l(f) = S_l$ per $f \in B(l)$ e 0 altrimenti. La potenza totale trasmessa è allora $P(l) = S_l B(l)$. Se si richiede che l' SNR ricevuto sia almeno uguale ad una soglia specificata SNR_0 , la potenza di trasmissione è data da:

$$P_3(l) = SNR_0 B_3(l) \frac{\int_{B_3(l)} N(f)df}{\int_{B_3(l)} A^{-1}(l, f)df} \tag{2.22}$$

dove $B_3(l)$ è la banda ai $3dB$ definita in precedenza. Sebbene tale definizione possa sembrare soddisfacente, non vi è alcuna garanzia della sua ottimalità. Potrebbe essere infatti possibile ottenere un migliore utilizzo delle risorse grazie ad una differente distribuzione dell'energia del segnale trasmesso sulla banda di sistema. In altre parole, si potrebbe giocare sulla densità spettrale di potenza del segnale trasmesso $S_l(f)$ in accordo con i valori di $A(l, f)$ e $N(f)$.

2.8.2 Una definizione di ampiezza di banda basata sulla capacità

Si supponga di dividere l'ampiezza di banda totale in sotto-bande centrate intorno alla frequenze $f_i, i = 1, 2, \cdots$ di ampiezza Δf tale per cui $A(l, f)$ possa essere assunto costante e $N(f)$ possa essere approssimato bianco con densità spettrale di potenza costante data da $N(f_i)$; la capacità totale può essere espressa come:

$$C(l) = \sum_i \Delta f \log_2 \left[1 + \frac{S_l(f_i)A^{-1}(l, f_i)}{N(f_i)}\right] \tag{2.23}$$

Se si massimizza la capacità $C(l)$ rispetto a $S_l(f)$, tenendo conto che la potenza totale trasmessa $P(l)$ è finita, si ottiene la distribuzione ottimale di energia [1]. La distribuzione spettrale di potenza del segnale trasmesso dovrebbe soddisfare il principio *water-filling*

$$S_l(f) + A(l, f)N(f) = K_l \tag{2.24}$$

dove K_l è una costante il cui valore deve essere determinato dalla potenza trasmessa $P(l)$ in modo tale che $S_l(f) > 0$.
La potenza $P(l)$ può essere scelta allo scopo di ottenere un SNR_0 desiderato. L'SNR corrispondente alla distribuzione ottimale di energia è dato da:

$$\begin{aligned} SNR(l, B(l)) &= \frac{\int_{B(l)} S_l(f) A^{-1}(l,f) df}{\int_{B(l)} N(f) df} \\ &= K_l \frac{\int_{B(l)} A^{-1}(l,f) df}{\int_{B(l)} N(f) df} - 1 \end{aligned} \tag{2.25}$$

La potenza in trasmissione allora è

$$P(l) = \int_{B(l)} S_l(f) df = K_l B(l) - \int_{B(l)} A(l,f) N(f) df \tag{2.26}$$

Se tale potenza deve essere tale che

$$SNR(l, B(l)) \geq SNR_0 \tag{2.27}$$

allora la distribuzione ottimale di energia $S_l(f)$ può essere ottenuta dalla seguente procedura numerica. Per ogni distanza l si comincia cercando la frequenza ottima $f_0(l)$, e si imposta di conseguenza il valore iniziale di K_l a $K_l^{(0)} = AN_{min}(l)$. Si procede quindi iterativamente, aumentando K_l ad ogni passo di un piccolo valore, finchè la condizione (2.27) è soddisfatta. In particolare , se K_l^n è il valore corrente di K_l per il quale lo SNR sia ancora minore della soglia SNR_0, si eseguono le seguenti operazioni:

1. si determina $B^{(n)}(l)$ come la regione spettrale per la quale $A(l,f)N(f) \leq K_l^{(n)}$;
2. si calcola $SNR^{(n)}$ da (2.25) utilizzando $B^{(n)}(l)$ e $K_l^{(n)}$;
3. si confronta $SNR^{(n)}$ con SNR_0. Se $SNR^{(n)} < SNR_0$, si aumenta K_l di un piccolo incremento e si itera la procedura.

Quando $SNR^{(n)}$ raggiunge o è maggiore di SNR_0 la procedura termina. In quest'ultimo caso il valore corrente di $K_l^{(n)}$ è il valore che assumerà K_l ed il valore $B_l^{(n)}$ è quello che assumerà $B(l)$. La distribuzione ottima di energia è:

$$S_l(f) = \begin{cases} K_l - A(l,f)N(f) & f \in B(l) \\ 0 & \text{altrimenti} \end{cases} \tag{2.28}$$

e la potenza totale $P(l)$ è ottenuta da (2.26). Infine, la capacità del canale è data da:

$$C(l) = \int_{B(l)} \log_2 \left[\frac{K_l}{A(l,f)N(f)} \right] df \tag{2.29}$$

mentre la capacità definita inizialmente tramite lo schema euristico che utilizza una distribuzione uniforme d'engergia nell'ampiezza di banda ai $3db$ $B_3(l)$ è data da:

$$C_3(l) = \int_{B_3(l)} \log_2 \left[1 + \frac{P_3(l)/B_3(l)}{A(l,f)N(f)} \right] \tag{2.30}$$

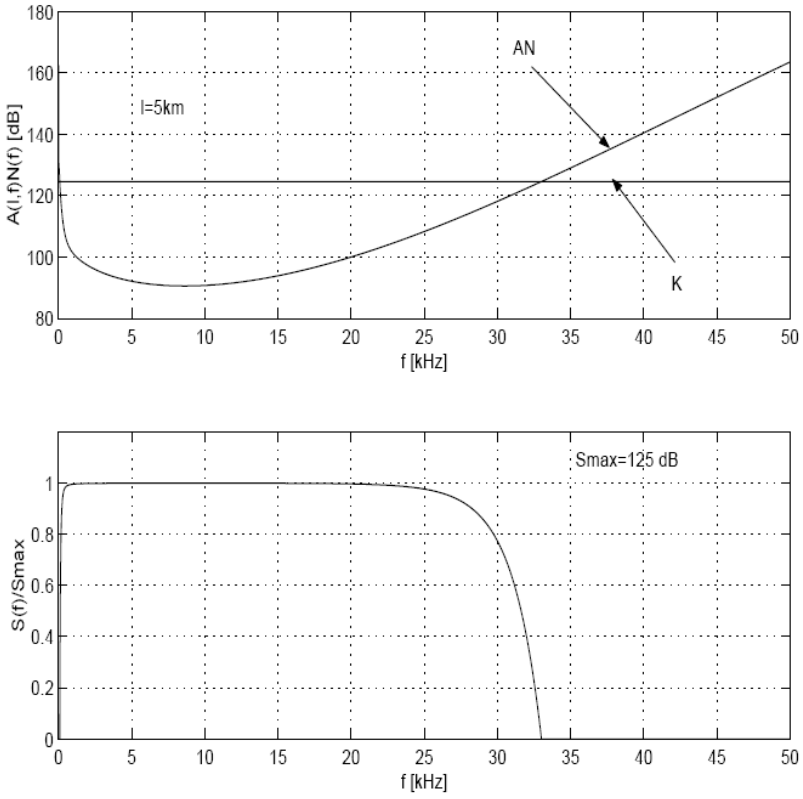

Figura 2.9: Il risultato della procedura per trovare la densità spettrale di potenza ottima del segnale in trasmissione ad una distanza di $5Km$; la figura in alto mostra $A(l,f)N(f)$ ed il livello costante K_l per il quale l' SNR ricevuto è uguale a $SNR_0 = 20dB$; la figura in basso mostra la densità spettrale di potenza risultante.

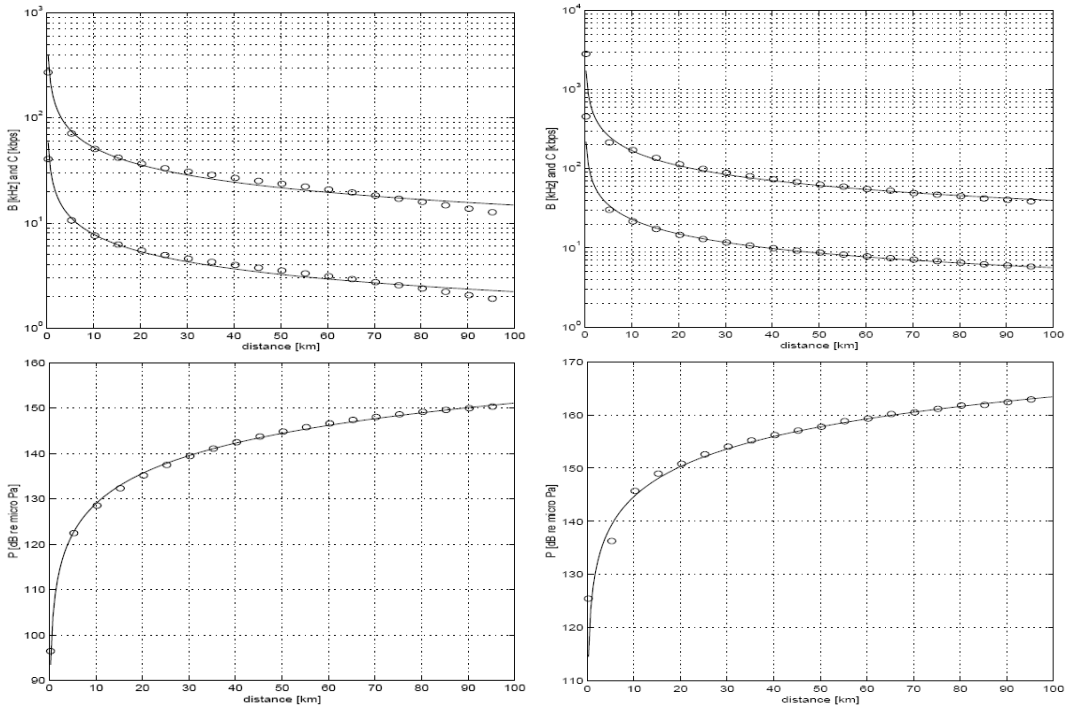

Figura 2.10: Banda e capacità (figure in alto) e potenza in trasmissione (figure in basso) necessarie per ottenere $SNR_0 = 20dB$. Si sono utilizzate la distribuzione di energia uniforme e l'ampiezza di banda ai $3dB$ (le figure a sinistra) e la distribuzione di energia ottima e l'ampiezza di banda ottima (figure a destra).

2.9 Lo strato fisico in reti sottomarine

Fino all'inizio dell'ultimo decennio, a causa delle estreme caratteristiche del canale sottomarino, i sistemi di trasmissione erano basati su modulazioni non-coerenti come la modulazione *FSK*[1] che dipendono dal rilevamento di energia in una certa banda e non richiedono ad esempio *tracking* della fase, che è reso difficile dall'elevata dispersione Doppler descritta nelle sezioni precedenti.
Negli schemi di modulazione *FSK* sviluppati, gli effetti del *multi-path* sono soppressi inserendo dei tempi di guardia tra un simbolo ed il successivo, per essere certi che l'eventuale coda di segnali secondari termini prima dell'arrivo del simbolo seguente. Delle frequenze di guardia dinamiche possono essere utilizzate tra i toni in frequenza per adattare il sistema alla dispersione Doppler del canale. Sebbene gli schemi di modulazione non-coerente siano caratterizzati da un alta

[1]Frequency Shifting Key

efficienza della potenza utilizzata, altrettanto non si può dire riguardo all'efficienza dell'ampiezza di banda.
Per questo motivo si è successivamente passati allo sviluppo di sistemi di modulazione coerente per sistemi a lunga distanza e ad alto *throughput*[2]. Negli ultimi anni, modulazioni pienamente coerenti come la *PSK*[3] e la *QAM*[4] sono stata utilizzate, grazie alla recente disponibilità di sempre più potenti *DSP*[5]. Si sono potute così sfruttare anche le tecniche di equalizzazione di canale per combattere gli effetti dell'*ISI*, invece di tentarne la soppressione. Equalizzatori *DFE*[6] sono comunemente utilizzati per il tracking della risposta impulsiva tempo-variante del canale e per fornire un alto *throughput* in trasmissione. Nel caso il canale sia velocemente tempo-variante è necessario combinare l'utilizzo di un *DFE* con un *PLL*[7] che ha il compito di stimare e compensare le differenze di fase, in maniera rapida e stabile. La figura seguente presenta l'evoluzione dai sistemi di modulazione non coerente ai recenti sistemi coerenti:

Modulazione	Anno	Rate [Kbps]	Ampiezza di banda [kHz]	Distanza [Km]
FSK	1984	1.2	5	3_s
PSK	1989	500	125	0.06_d
FSK	1991	1.25	10	2_d
PSK	1993	0.3 - 0.5	0.3 - 1	200_d - 90_s
PSK	1994	0.02	20	0.9_s
FSK	1997	0.6 - 2.4	5	10_d - 5_s
DPSK	1997	20	10	1_d
PSK	1998	1.67 - 6.7	2 - 10	4_d - 2_s
16-QAM	2001	40	10	0.3_s

Figura 2.11: Evoluzione dei sistemi di trasmissione sottomarini. s e d sono sinonimi di acque poco e molto profonde

La modulazione *DPSK*[8] può essere vista come una soluzione intermedia tra

[2]Si definisce *throughput* l'inverso del numero medio di simboli trasmessi per ogni singolo bit ricevuto senza errori.
[3]Phase Shifting Key
[4]Quadrature Amplitude Modulation
[5]Digital Signal Processor
[6]Decision-Feedback Equalizer
[7]Phase Locked Loop
[8]Differential Phase Shifting Key

le modulazioni non-coerenti e quelle coerenti se si considera l'efficienza spettrale. Essa infatti decodifica l'informazione basandosi sulla differenza di fase tra un simbolo ed il precedente; per questo motivo può essere chiamata una modulazione parzialmente coerente. Sebbene questa strategia renda meno stringenti i requisiti di *tracking* della fase della portante, essa ottiene una probabilità d'errore maggiore rispetto ad una modulazione *PSK* di pari *rate* di trasmissione.
Se si esamina la tabella precedente, si nota che i primi sistemi coerenti in fase avevano un'efficienza di banda maggiore (bit rate / ampiezza di banda occupata), ma le prestazioni non erano ancora nettamente superiori a quelle delle modulazioni non-coerenti. Solo grazie all'implementazione dei *DFE* si è compiuto il salto di qualità definitivo. Tuttavia tali filtri hanno una complessità tale da non poter soddisfare il requisito di operare in tempo reale. E' stato necessario quindi trovare un compromesso progettando filtri sub-ottimi.
Un'altra soluzione promettente è la modulazione *OFDM*[9] che utilizza sottoportanti ortogonali a banda stretta, tali da poter approssimare attenuazione e rumore come costanti sull'estensione di ciascuna sottobanda. Tale modulazione ottiene buone prestazioni ed un'elevata efficienza spettrale in presenza di elevato *multipath*.
Infine si fa notare come molte delle modulazioni sopracitate richiedano una stima di canale efficace, quale può essere la stima *data-aided*, tramite cioè delle sequenze di simboli concordate a priori.

2.10 Lo strato *MAC* in reti sottomarine

Negli ultimi anni, la ricerca nel campo delle reti radio *ad-hoc* è stata intensiva e produttiva. Tuttavia, i risultati ottenuti non sono direttamente applicabili alle reti acustiche sottomarine, per le sostanziali differenze tra i canali spiegate nelle sezioni precedenti.
Le prime ricerche in tal senso hanno evidenziato come *MAC*[10] basati su *TDMA*[11] abbiano un'efficienza di banda limitata, a causa dei lunghi tempi di guardia richiesti per operare in ambiente sottomarino ed ai lunghi tempi di propagazione che rendono difficile la corretta sincronizzazione dei nodi.

[9] Orthogonal Frequency-Division Multiplexing
[10] Medium Access Control
[11] Time Division Multiple Access

Nemmeno i protocolli *FDMA*[12], ad oggi, non hanno avuto sorte migliore, a causa della scarsa banda disponibile e della vulnerabilità delle singole sotto-bande ai fenomeni di *fading*.
Altri ricerche sono state svolte su protocolli di accesso basati su *CSMA*[13] e *CDMA*[14] e su combinazioni di più caratteristiche qui descritte.

2.10.1 Protocolli *CSMA*

Un primo esempio di questa categoria è il protocollo *Slotted FAMA*[15]. Esso combina sia l'interrogazione della portante (*Carrier Sensing*) sia il dialogo tra la sorgente ed il ricevitore prima della trasmissione dati. Durante il dialogo iniziale, sono scambiati pacchetti di controllo per evitare trasmissioni multiple contemporanee. Sebbene la divisione in *slot* temporali elimini il bisogno di pacchetti di controllo di grandi dimensioni provvedendo così ad un risparmio energetico, essa richiede l'aggiunta di tempi di guardia alla durata dello slot per ovviare a possibili sfasamenti nella sincronizzazione tra i nodi. Inoltre, a causa dell'elevato tempo di propagazione, il meccanismo di *handshaking* può portare a regime ad un *throughput* minore; senza tralasciare il fatto che il meccanismo di *carrier sense* non sia del tutto efficace sempre per lo stesso motivo.
Un altro esempio è il protocollo *PCAP*[16]. Il suo obiettivo è quello di controllare il tempo speso per la creazione della connessione, e di evitare possibili collisioni provvedendo allo *scheduling* delle attività dei sensori. Sebbene il *throughput* ottenuto da quest'ultimo protocollo sia elevato rispetto a quello di molti altri protocolli comunemente utilizzati nelle reti radio di sensori, non è considerabile come soluzione flessibile per applicazioni con requisiti eterogenei.

2.10.2 Protocolli *CDMA*

Tali protocolli si basano su modulazioni a *Espansione di Spettro* (*Spread Spectrum*) e sfruttano la naturale robustezza di tali tecniche al *fading* selettivo in frequenza dovuto al *multipath*, espandendo la banda del segnale su tutto lo spet-

[12]Frequency Division Multiple Access
[13]Carrier Sense Multiple Access
[14]Code Division Multiple Access
[15]Floor Acquisition Multiple Access
[16]Propagation-delay-tolerant Collision Avoidance Protocol

tro disponibile. Questo rende possibile l'utilizzo di filtri *Rake* al ricevitore per sfruttare la diversità temporale creata. In questo modo si diminuisce il numero di ritrasmissioni, con conseguente riduzione del consumo energetico.
Un primo confronto è stato compiuto tra la modulazione *DSSS*[17] e quella *FHSS*[18]. Nella prima il simbolo dati viene suddiviso più sottosimboli di minore durata e quest'ultimi vengono poi codificati tramite codici aventi proprietà di buona auto-correlazione e cross-correlazione, atti cioè a minimizzare la interferenza multi-utente (*MAI*[19]). Nella seconda invece utenti differenti utilizzano differenti sotto-bande per la trasmissione dei *chip* secondo delle sequenze di *hop* ortogonali aventi anch'esse delle buone proprietà di auto-correlazione e di cross-correlazione. In tale confronto si è provato che l'utilizzo della modulazione *DSSS* ha ottenuto una probabilità di errore minore.
Si è successivamente tentato l'approccio combinato di trasmissioni multi-portante e di *DSSS*. Tale approccio ha portato ad un aumento del *throughput* rispetto alla controparte a singola portante, e potrebbe inoltre essere utilizzato per supportare applicazione che richiedono qualità di servizio (*QOS*) differenti.

2.10.3 Sviluppi futuri

Sebbene molti altri studi siano stati fatti in tutti i settori dell'accesso condiviso, la superiorità di uno rispetto agli altri è ancora tutta da accertare. I protocolli futuri tuttavia dovranno affrontare e risolvere numerose problematiche tra cui:

- lo studio approfondito della lunghezza efficiente dei pacchetti dati al fine di massimizzare l'efficienza di utilizzo del canale;
- lo sviluppo di codificatori e decodificatori a bassa complessità per limitare l'energia richiesta da tali sistemi;
- un confronto del consumo energetico dei protocolli distribuiti e centralizzati;
- la progettazione di famiglie di codici per protocolli CDMA con alta auto-correlazione e minima cross-correlazione per ottenere una minima interferenza multiutente.

[17] Direct Sequence Spread Spectrum
[18] Frequency Hopping Spread Spectrum
[19] Multiple Access Interference

2.11 Lo strato di *routing* in reti sottomarine

I protocolli di routing per reti radio *ad hoc* sono stati studiati a fondo nel corso degli ultimi decenni. Tuttavia come nel caso dei protocolli di *MAC*, essi presentano degli svantaggi se applicati senza la minima accortezza alle reti sottomarine. I protocolli esistenti possono essere divisi in

1. *proattivi*
2. *reattivi*
3. *geografici*

2.11.1 I protocolli proattivi

Protocolli come *DSDV* [19], *OLSR* [18] causano alto traffico sulla rete al fine di stabilire delle rotte sia la prima volta sia ogni volta che la topologia di rete è modificata a causa di spostamento o rottura di un nodo, poichè la nuova topologia deve essere comunicata a tutti i nodi presenti.
In questo modo ogni nodo è in grado di comunicare con ogni altro nodo della rete, cosa che potrebbe essere ridondante nel canale sottomarino, data la bassa densità di sensori in tali topologie. Se si unisce tutto ciò al fatto che l'ampiezza è limitata e al fatto che il ritardo di propagazione è elevato risulta chiaro come i protocolli di questa categoria non siano adatti per le reti sottomarine.

2.11.2 I protocolli reattivi

Appartengono a questa categoria protocolli come *AODV* [20] e *DSR* [21]; essi sono più appropriati per ambienti dinamici come le reti sottomarine, ma sono affetti da alti tempi di latenza per la convergenza di una topologia e richiedono un *flooding* di pacchetti di controllo allo scopo di stabilire nuove rotte.
Considerato il fatto che un'alta latenza in campo radio si traduce in una maggiore latenza nella topologia sottomarina, appare evidente come anch'essi non costituiscano un buon approccio per tali reti. A supporto di ciò, si consideri che le topologie subacquee sono di solito quasi-statiche (se si escludono le reti di *AUV*[20]).

[20] Autonomous Underwater Vehicle

2.11.3 I protocolli geografici

I protocolli geografici offrono potenzialmente un'alta scalabilità, al prezzo tuttavia di un dispendio di risorse per la stima della posizione dei nodi. I ricevitori GPS[21] utilizzati nelle reti radio per la localizzazione non funzionano correttamente in ambiente sottomarino: le onde elettromagnetiche trasmesse nella banda di $1.5GHz$ non si propagano in acqua. Nonostante ciò la localizzazione in tali ambienti è possibile, e molti protocolli sono stati sviluppati a tale scopo. Molti di essi sfruttano la bassa velocità di propagazione del suono, che permette un *timing* accurato dei segnali permettendo così il calcolo della distanza relativa tra due nodi. Tale distanza può essere poi utilizzata per calcolare l'esatta posizione dei nodi stessi, ed instaurare comunicazioni basate sulla posizione dei nodi.
Tuttavia l'ottimizzazione energetica in tale settore non è ancora stata raggiunta.

2.11.4 Sviluppi futuri

L'aspetto che i protocolli di routing di prossima generazione dovranno rispettare rigorosamente è l'ottimizzazione e delle risorse engergetiche. A causa delle dipendenza peculiare dell'ampiezza di banda dalla distanza, spesso instaurare un collegamento diretto tra due nodi si rivela essere la scelta peggiore. Si consideri infatti la statistica del ritardo in una rete *multi-hop*:

$$T = \tau_p + T_{tx}^{(n)} \tag{2.31}$$

il tempo di propagazione τ_p dipende esclusivamente dalla velocità di propagazione del suono in acqua e dalla distanza tra la sorgente ed i nodi riceventi mentre la seconda componente $T_{tx}^{(n)}$ è dipendente dal rate di trasmissione e dal numero n di nodi interposti tra il nodo sorgente e quello destinazione. Nelle reti radio terrestri, il tempo di trasmissione è linearmente dipendente dal numero di nodi intermedi, poichè lo stesso messaggio dovrebbe essere trasmesso più volte allo stesso *rate* (se consideriamo lo stesso protocollo per tutti i nodi coinvolti). Nelle reti sottomarine tale dipendenza invece è *sub-lineare*, poichè all'aumentare del numero di nodi la distanza tra di essi diminuisce. Di conseguenza è possibile trasmettere un *rate* maggiore, ottenendo conseguentemente un consumo energetico minore, a scapito di un limitato ritardo dovuto alla comunicazione *multi-hop*.

[21] Global Positioning System

2.12 Lo strato di trasporto in reti sottomarine

Lo strato di trasporto sarà necessario nelle reti sottomarine di prossima generazione principalmente per le sue caratteristiche di *trasporto affidabile*, di *controllo di flusso* e di *controllo di congestione*. Molte delle odierne implementazioni del *TCP*[22] sono incompatibili con l'ambiente subacqueo, perchè le caratteristiche di controllo di congestione e di flusso sono basate su un meccanismo a finestra che risiede a sua volta su un'accurata stima del *RTT*[23]. Poichè il canale acustico è caratterizzato da un elevato valore di *RTT* si otterrebbe un *throughput TCP* minore di quello massimo realizzabile. Inoltre, la sua elevata varianza, dovuta alle forti distorsione provocate dal canale, renderebbe un compito arduo settare correttamente i timeout di cui il protocollo *TCP* necessita. I protocolli di trasporto esistenti basati su un rate di trasmissione fisso non hanno sorte migliore, poichè anch'essi si basano su messaggi di controllo per adattare dinamicamente tale rate. Un primo tentativo è stato fatto con il protocollo *SDRT*[24] [24]. L'idea principale è quella di utilizzare codici *Tornado* per recuperare i pacchetti errati e per ridurre il numero di trasmissioni. I pacchetti sono trasmessi blocco per blocco e ciascun blocco è inoltrato per ogni *hop*. Una prima versione prevedeva l'invio di tutti pacchetti appartenenti allo stesso blocco prima ancora di aver ricevuto un primo feedback positivo, portando così ad uno spreco di energia. Tale problema è stato risolto inserendo un meccanismo di controllo a finestra. I pacchetti entro tale finestra vengono trasmessi ad un rate maggiore di quelli fuori da essa.
Un possibile svantaggio di *SDRT* è che la codifica e la decodifica *Tornado* sono operazioni computazionalmente onerose, anche se utilizzano soltando delle operazioni di *XOR*[25]. Nel protocollo *SDRT* inoltre, non c'è meccaniscmo alcuno che garantisca l'affidabilità *end-to-end*, poichè l'instradamento avviene *hop* per *hop*. Ogni nodo, in aggiunta deve procedere alla decodifica e alla ricodifica per la trasmissione all'hop successivo.

[22] Transmission Control Protocol
[23] Round-Trip Time
[24] Segmented Data Reliable Transport
[25] *or* esclusivo bit a bit

2.12.1 Sviluppi Futuri

Una soluzione completa ed efficiente dovrà essere basata sui seguenti principi:

Zone d'ombra. Sebbene la gestione di tali eventi richieda l'aiuto di uno strato di rete per il corretto reinstradamento, un protocollo di trasporto dovrebbe considerare tali eventi.

Minimo consumo d'engergia. Un protocollo di trasporto dovrebbe essere progettato specificatamente per minimizzare il consumo d'energia.

Trasmissioni basate su rate dinamici. Un protocollo di trasporto dovrebbe poter permettere l'adattare del proprio rate di trasmissione alle condizioni di canale e della rete.

Reazione rapida alle congestioni. Un protocollo di trasporto dovrebbe adattarsi alle eventuali congestioni in modo rapido e diminuire il tempo di reazione di conseguenza. Tale comportamento dovrebbe essere prerogativa dei nodi intermedi e non del nodo centrale (*sink*).

Interazione *Cross-Layer*. Perdite di connessione o errori sul pacchetto dovrebbero innescare il protocollo a prendere le giuste contromisure. Tali scelte critiche dovrebbero essere supportate da informazione presenti agli strati sottostanti.

Affidabilità. Un meccanismo di affidabilità *hop*-per-*hop* è una buona soluzione per l'efficienza energetica complessiva. Tuttavia dovrebbe essere presente anche un meccanismo che garantisca l'affidabilità *end-to-end*.

2.13 Possibili Scenari

In questa sezione si vuole dare una breve panoramica delle topologie di rete sottomarine tipiche:

Topologie statiche *2-D*. Sono di solito costituite da sensori ancorati al fondale marino. Tipiche applicazioni sono il monitoraggio ambientale e quello dei movimenti delle placche continentali.

Topologie statiche *3-D*. Sono costituite da sensori ancorati al fondale marino e che possono controllare dinamicamente la propria profondità. Il tipico utilizzo è quello della sorveglianza ambientale o del monitoraggio di fenomeni oceanici, quali correnti d'acqua, inquinamento, etc.

Topologie dinamiche *3-D*. Sono costituite da una parte infrastrutturata e quindi statica e da una parte in movimento, spesso mezzi *AUV*.

2.13.1 Reti acustiche in due dimensioni

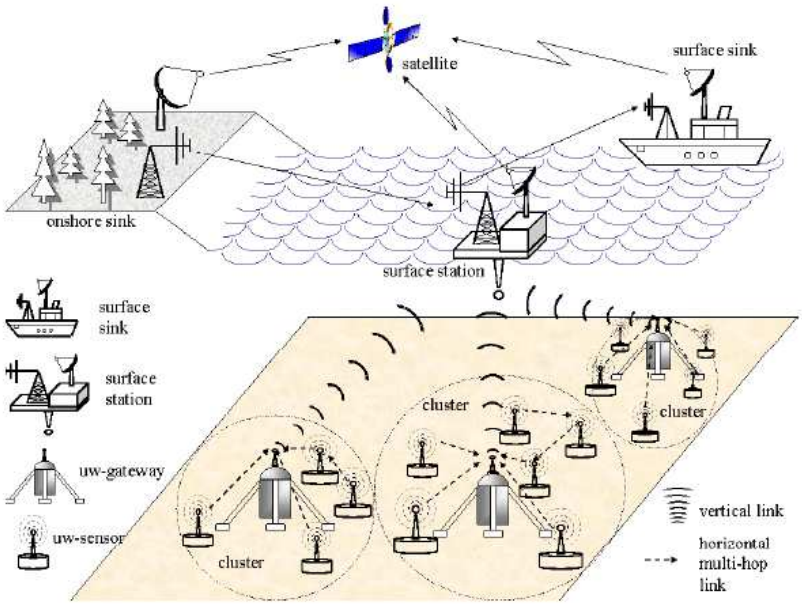

Figura 2.12: Architettura di una rete acustica sottomarina di sensori in due dimensioni

Un tipico scenario di riferimento è quello presente in Figura 2.12. Un gruppo di sensori è ancorato al fondale marino. I sensori sono interconnessi da uno o più

gateway sottomarini per mezzo di collegamenti acustici. Tali *gateway* hanno il compito di trasmettere le informazioni della cella (*cluster*) sottomarina al nodo centrale o stazione superficiale. Per ottenere tale obiettivo essi sono equipaggiati con due trasduttori acustici, chiamati trasduttore orizzontale e verticale. Il primo dei due è utilizzato per le comunicazioni intra-cellulari di controllo e raccolta dati, mentre il secondo è utilizzato per comunicare con la stazione superficiale. In applicazione in acque profonde, i trasduttori verticali devono supportare comunicazioni a lungo raggio. La stazione superficiale è equipaggiata di trasduttore acustico in grado di gestire comunicazioni parallele con i *gateway* ed è attrezzata di trasmettitore radio per le comunicazioni superficiali.
I sensori possono essere connessi ai *gateway* via collegamento diretto o tramite cammini *multi-hop* per una trasmissione energicamente efficiente come descritto nelle sezioni precedenti.

2.13.2 Reti acustiche in tre dimensioni

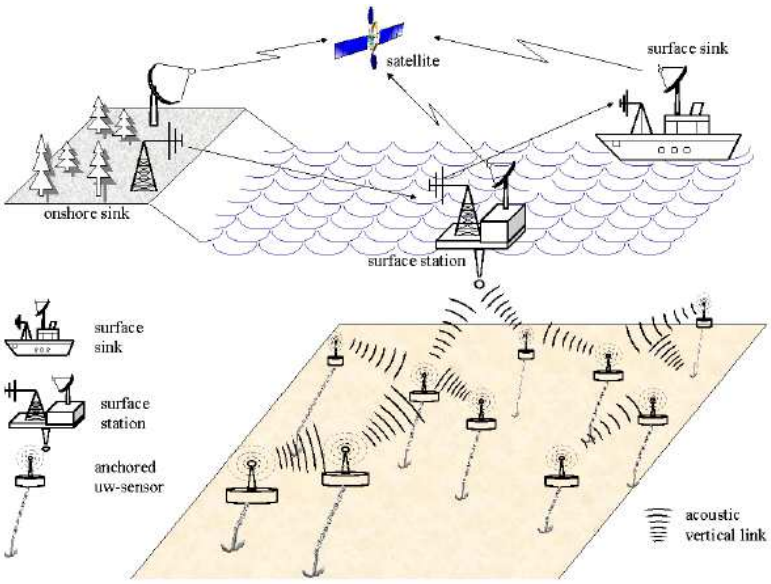

Figura 2.13: Architettura di una rete acustica sottomarina di sensori in tre dimensioni

Le reti acustiche di sensori in tre dimensioni sono utilizzate solitamente per rilevare ed osservare fenomeni che non possono essere adeguatamente dai sensori ancorati al fondale oceanico, quando è necessaria cioè una cooperazione tra i no-

di. In tali topologie i sensori fluttuano a profondità diverse grazie a due possibili approcci:

- si collega il sensore ad una boa superficiale tramite un cavo regolabile in lunghezza. Tuttavia le boe superficiali possono ostruire la navigazione, essere facilmente identificate dal nemico in caso di intrusione, essere soggette alle intemperie.
- Si ancora il sensore al fondale e lo si attrezza con una boa gonfiabile tramite pompa. Tale boa spinge il sensore verso la superficie: regolando la lunghezza del cavo di ancoraggio è possibile controllare la profondità del sensore stesso.

2.13.3 Reti acustiche di mezzi mobili in tre dimensioni

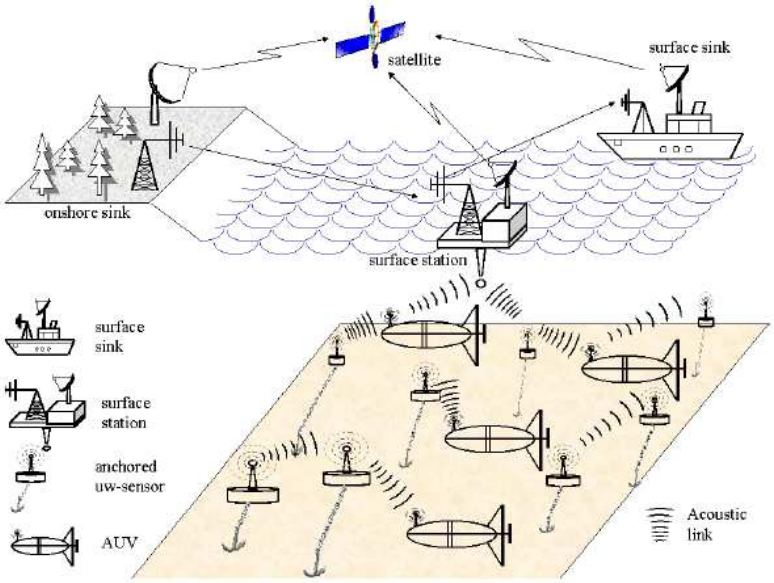

Figura 2.14: Architettura di una rete acustica sottomarina di AUV in tre dimensioni

I mezzi AUV possono operare senza essere controllati e alimentati da remoto e per questo motivo sono ampiamente utilizzati negli studi oceanografici. L'integrazione di tali mezzi con le reti statiche di sensori acustiche richiederà nuovi algoritmi di coordinamento come ad esempio:

Raccolta dati adattiva. Questo include tutte le strategie di controllo per comandare i veicoli mobili per posizionarsi nei luoghi in cui la loro raccolta

dati sarà più utile. Ad esempio la loro densità potrebbe essere dinamicamente controllata ed aumentata nelle zone in cui è richiesto un maggior numero di campioni per il fenomeno osservato.

Auto configurazione. Questo include le procedure di controllo per rilevare automaticamente delle assenza di connessione a causa di rottura di nodi o di canale fortemente disturbato e per richiedere l'intervento di uno o più *AUV*. Inoltre tali mezzi mobili potranno essere utilizzati per la riparazione o l'installazione dei sensori stessi.

Altro obiettivo sarà il progetto efficiente dei mezzi *AUV* stessi: il primo tra questi è senza dubbio migliorare l'indipendenza dal controllo umano. Inoltre sono necessarie delle strategie per la coordinazione autonoma, per l'aggiramento di ostacoli e strategie di navigazione. Si richiede infine la capacità si ricaricare le proprie batterie e l'engergia solare potrebbe essere adeguatamente sfruttata a tale scopo.
Allo stato attuale sono presenti due categorie di *AUV*: i *drifter* ed i *glider*. I primi sono progettati per scivolare con la corrente oceanica e sono in grado di risalire la colonna d'acqua. Essi sono utilizzati per raccogliere dei campioni ad una data profondità. I secondi invece sono dei veicoli a batteria che usano delle pompe idrauliche per variare il proprio volume e controllare così il proprio movimento.

Capitolo 3

I codici a fontana

3.1 Introduzione

I *codici a fontana* [11] sono codici *rateless* a pacchetto. Sono detti a fontana perchè il codificatore può produrre un quantitativo potenzialmente illimitato di pacchetti codificati. La lunghezza dei simboli che formano le parole di codice può essere arbitraria, di l-bit per simbolo. Se i dati originali sono composti da k simboli, allora ogni simbolo codificato può essere generato indipendentemente da altri simboli di codice in un numero medio di $O(\ln(k/\delta))$ operazioni su simboli d'ingresso, mentre i simboli d'informazione possono essere recuperati da qualsiasi set di $k + O(\sqrt{k}\ln^2(k/\delta))$ di simboli codificati con probabilità di $1 - \delta$ e con una media di operazioni su simboli di $O(k \cdot \ln(k/\delta))$
. Come detto in precedenza i codici a fontana sono *rateless*, ovvero il numero di simboli di codice può essere generato all'occorrenza fino a quando il ricevitore non abbia recuperato i dati originali, quali che siano le condizioni di canale. Poichè l'efficienza di codifica $k/(k+O(\sqrt{k}\ln^2(k/\delta)))$ è prossima all'unità, i codici a fontana sono quasi-ottimi sotto questo punto di vista.

3.2 Il canale a cancellazione

I canali a cancellazione sono di estrema importanza per le reti di telecomunicazioni. Si pensi ad esempio ad un trasferimento file tramite rete *Internet*: il file originale viene segmentato in pacchetti che, dopo essere stati spediti, sono o non sono ricevuti dal destinatario. Si consideri ancora l'utilizzo di codici con

una buona probabilità di correzione e di un canale rumoroso: occasionalmente la decodifica fallisce e l'intero pacchetto viene scartato.

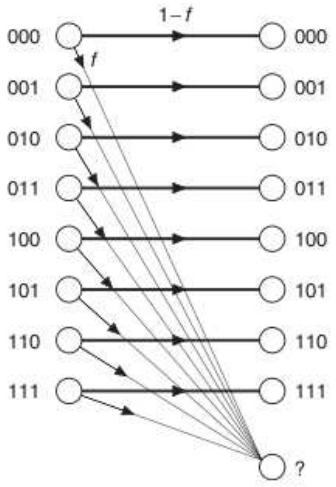

Figura 3.1: Il canale a cancellazione *8-ario*

Un semplice modello di canale a cancellazione *q-ario* è quello presente in Figura 3.1 : esso ha probabilità condizionata:

$$p(y|x) = \begin{cases} 1-f & y = x \\ f & y \neq x \end{cases} \quad (3.1)$$

$$x, y \in \{1, \cdots, q\}$$

dove x è il simbolo in ingresso, y è il simbolo in uscita, $\{1, \cdots, q\}$ è l'alfabeto dei simboli, e f è la probabilità di cancellazione del simbolo trasmesso.

Metodologie comuni per la comunicazione su tali canali sono l'utilizzo di tecniche di ARQ[1], in cui il ricevitore richiede alla sorgente i singoli pacchetti cancellati dal canale. Tuttavia in caso di alta probabilità di cancellazione f il numero di ritrasmissioni richieste potrebbe saturare il canale, o portare alla ricezione di copie multiple dei pacchetti stessi. Il teorema di *Shannon* di codifica di canale afferma che la capacità del canale è di $C = (1-f)\lceil \log_2 q \rceil$ bit sia che si utilizzi o no il canale di *feedback*. Una comunicazione affidabile dovrebbe essere quindi possibile ad un tale *rate* con l'aiuto di un appropriato codice *FEC*.

L'inefficienza dei protocolli ARQ è particolarmente evidente nel caso del *broadcast*, dove la capacità di canale sarebbe nella quasi totalità utilizzata dalle richieste dei pacchetti mancanti per valori di f elevati[2]. I codici a blocco più utilizzati nei canali a cancellazione sono i codici *Reed-Solomon*. Per un codice *R-S* di tipo (n, k) e di alfabeto $q = 2^l$ è necessario ricevere correttamente un insieme di qualsiasi k pacchetti codificati su n per poter decodificare con successo i k pacchetti di informazione. Tuttavia essi sono pratici per valori piccoli di k, n e q: le implementazioni standard degli algoritmi di codifica e decodifica hanno inoltre un costo dell'ordine di $k(n-k)\log_2 n$; infine, per ogni codice a blocco è

[1] Automatic Repeat reQuest

[2] si tratta del ben noto problema del *NACK implosion*

necessario stimare a priori la probabilità di cancellazione f del canale, e scegliere di conseguenza il *rate* di codifica $R = k/n$. In caso di stima errata non è possibile aumentare il valore di n per estendere la capacità correttiva del codice.

3.3 La fontana lineare aleatoria

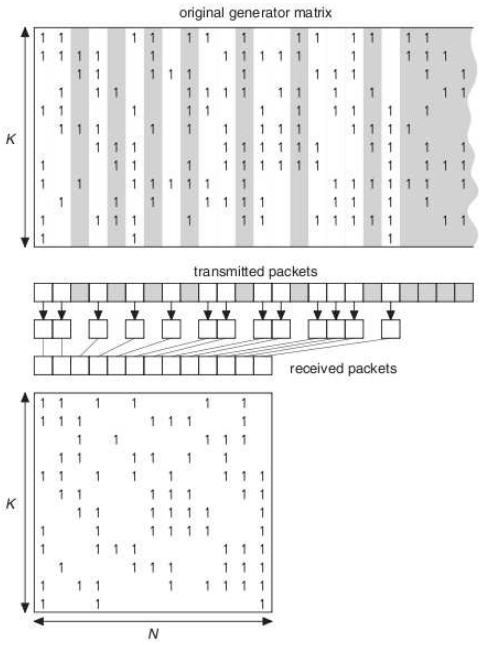

Figura 3.2: La matrice generatrice di un codice lineare aleatorio in trasmissione (in alto) ed in ricezione (in basso).

Si consideri la codifica di un *file* di dimensione K pacchetti, $s_1, \cdots, s_K$, dove per pacchetto si intende l'unità elementare che è trasmessa con successo o è cancellata dal canale. Ad ogni ciclo di *clock*, indicato con n, il codificatore genera K bit aleatori $\{G_{kn}\}$ ed il pacchetto trasmesso t_n è la somma modulo 2 dei pacchetti originali per i quali il coefficiente G_{kn} sia 1.

$$t_n = \sum_{k=1}^{K} s_k G_{kn} \qquad (3.2)$$

Tale somma può essere compiuta tramite successive operazioni di *xor* dei pacchetti stessi.

In riferimento alla figura 3.2, ogni insieme i di K bit aleatori è associato alla colonna i-esima della matrice generatrice di codice $\mathbf{G}$.
Si supponga che dopo la trasmissione dei pacchetti codificati il canale cancelli un numero aleatorio di pacchetti tale per cui il destinatario riceva N pacchetti. Si supponga inoltre che il ricevitore conosca quali siano le colonne G_{kn} della matrice $\mathbf{G}$ associate ai pacchetti ricevuti, ad esempio tramite la comunicazione da parte del codificatore del *seed* utilizzato per generare la sequenza pseudo-casuale di bit.
La probabilità di decodifica con successo al ricevitore può essere calcolata come segue [12]. Se $N < K$ il ricevitore non ha pacchetti a sufficienza per il recupero del file. Se $N = K$ è possibile il recupero: è necessario infatti che la matrice $\mathbf{G}$

di dimensioni $K \times K$ sia invertibile modulo 2 per poter permettere il calcolo di $\mathbf{G}^{-1}$ tramite *Gaussian elimination* ed il recupero dei vettori di codifica

$$s_k = \sum_{n=1}^{K} t_n G_{kn}^{-1} \tag{3.3}$$

La probabilità di corretta decodifica dato $N = K$ si riconduce quindi al calcolo della probabilità che una matrice $K \times K$ abbia l'i-esima colonna linearmente indipendente da tutte le altre:

$$P_c = \prod_{i=0}^{K-1} \left(1 - 2^{-(K-i)}\right) \tag{3.4}$$

Nel caso $N > K$ con $N = K + E$ si ha una corretta decodifica se è possibile individuare nella matrice generatrice in ricezione $\mathbf{G}$ di dimensione $K \times N$ una sottomatrice invertibile di dimensione $K \times K$. Sia $1 - \delta$ la probabilità che questo evento si verifichi, dove con δ si indica la probabilità dell'evento complementare. Tale probabilità di fallimento δ è mostrata in Figura 3.3 con E sull'asse delle ascisse per un valore di $K = 100$. Per ogni valore di K la probabilità di fallimento è limitata da :

$$\delta(E) \leq 2^{-E} \tag{3.5}$$

Tale bound è mostrato in Figura 3.3 dalla sottile linea a puntini.

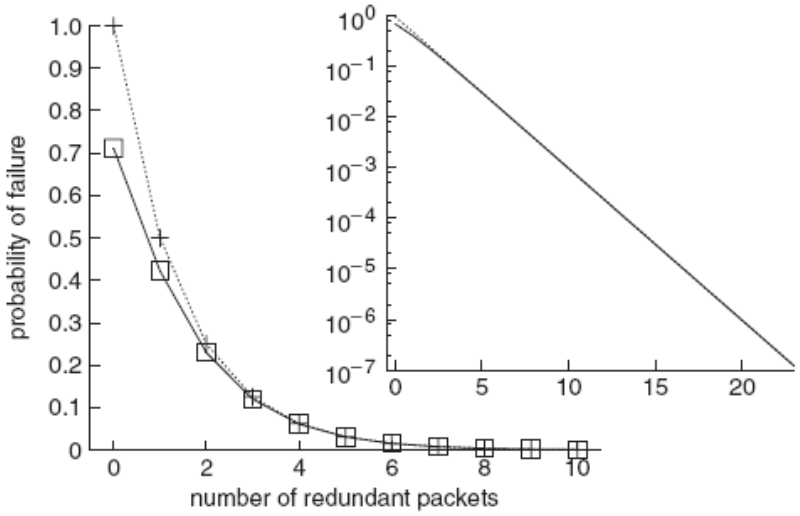

Figura 3.3: *Performance della fontana lineare aleatoria.* La linea continua mostra la probabilità di fallimento al crescere di E (a sinistra), e la stessa in scala logaritmica (a destra). La linea sottile a puntini rappresenta il limite superiore (3.5).

Riassumendo, il numero di pacchetti richiesti per avere una probabilità di successo $1-\delta$ è $N \approx K + \log_2(1/\delta)$; il costo medio di codifica per pacchetto è di $K/2$ operazioni sui pacchetti, il costo medio in decodifica è il costo medio dell'inversione matriciale $\mathbf{G}^{-1}$ dell'ordine di $O\left(K^3\right)$, mentre il costo delle operazioni di decodifica dei pacchetti tramite i coefficienti della matrice inversa è dell'ordine di $K^2/2$ operazioni su pacchetto.
I codici a fontana sono quindi dei codici quasi-perfetti per il canale a cancellazione ed all'aumentare di K essi possono avvicinarsi a piacere al limite di capacità di canale di Shannon. Il costo computazionale è stato migliorato da Michael Luby nel 2002 [11], secondo l'algoritmo descritto nelle sezioni seguenti.

3.4 I codici LT

I codici LT conservano le buone prestazioni dei codici a fontana descritti in precedenza, mentre riducono drasticamente la complessità di codifica e decodifica.

3.4.1 Il codificatore

Ogni pacchetto codificato t_n è ottenuto dai pacchetti sorgente $s_1, \cdots, s_K$ nel modo seguente:

1. si sceglie il grado d_n di tale pacchetto da una distribuzione di gradi $\rho(d)$; la scelta corretta di ρ dipende dalle dimensioni K del file originale come sarà ampiamente discusso successivamente;

2. si scelgono casualmente in modo uniforme d_n pacchetti di sorgente si calcola t_n come la somma *bit*-a-*bit* modulo 2 dei d_n pacchetti.

Tali operazioni di codifica individuano un grafo che connette i pacchetti codificati ai pacchetti originali. Se il grado medio $\bar{d}$ è significativamente più piccolo di K allora il grafo è sparso, e la matrice generatrice in ricezione è sparsa allo stesso modo.

3.4.2 Il decodificatore

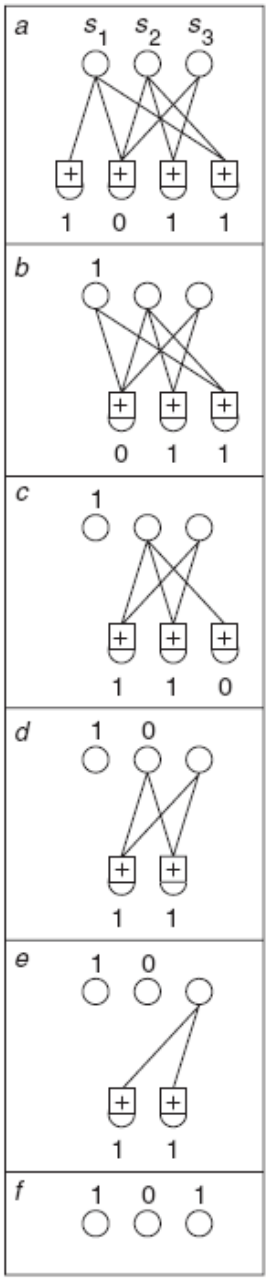

Figura 3.4: Esempio di decodifica con $K = 3$, $N = 4$, 1 bit per pacchetto .

La decodifica di un codice a grafo sparso è ancora più semplice nel caso di canale a cancellazione. Il compito del decodificatore è quello di recuperare il vettore $\mathbf{s}$ da $\mathbf{t} = \mathbf{sG}$, dove $\mathbf{G}$ è la matrice associata al grafo. Come nel caso dei codici a fontana supponiamo che il decodificatore sia in grado di ricostruire tale matrice grazie al *seed* utilizzato in fase di creazione di $\mathbf{G}$.
Il modo più semplice per cercare di risolvere il problema è il metodo detto di *message passing*. Tuttavia tale metodo ottiene un basso costo computazionale a scapito dell'efficienza di decodifica, ammettendo cioè l'esistenza di una probabilità di fallimento non nulla. Si chiamino i pacchetti codificati t_n *check node*, allora:

1. Si cerca un *check node* t_n tale che sia connesso ad un solo pacchetto di sorgente s_k. Se esso non è presente il processo di decodifica fallisce.
   (a) Si pone $s_k = t_n$.
   (b) Si somma (*bit*-a-*bit*, modulo 2) a tutti i $t_{n'}$ che sono connessi a s_k:
   $t_{n'} = t_{n'} + s_k$, $\forall\ n'$ tale che $G_{n'k} = 1$.
   (c) Si rimuovono tutti gli archi connessi al pacchetto sorgente s_k.
2. Si riparte dal punto (1) finchè non sono stati determinati tutti gli s_k.

Tale processo di decodifica è illustrato in Figura 3.4 per il caso $K = 3$, $N = 4$ e per un numero di bit per pacchetto pari a 1. Sono presenti tre pacchetti sorgente e quattro pacchetti codificati ricevuti all'inizio dell'algoritmo.

Alla prima iterazione l'unico *check node* che è connesso ad un solo $s_i = s_1$ è t_1. Si setta allora $s_1 = t_1$, si scarta il check node, si somma s_1 a tutti i *check node* a lui connessi e si cancella l'arco $t_1 \rightarrow s_1$. Si procede iterativamente fino al recupero completo dei tre pacchetti d'informazione.

3.4.3 Progettazione della distribuzione dei gradi

La distribuzione di probabilità dei gradi di un codice LT è la parte critica dell'intero processo: occasionalmente i pacchetti codificati devono avere un grado alto, $d_n \approx K$, per scongiurare l'evento in cui siano pacchetti di sorgente non connessi al grafo. Gran parte dei pacchetti inoltre deve avere un grado basso, per far sì che il processo di decodifica possa cominciare e procedere nel corso delle iterazioni, e per tenere sotto controllo il numero di operazioni sui pacchetti.
La proprietà che una distribuzione $\rho(d)$ dovrebbe avere per essere considerata soddisfacente è che la complessità in codifica e decodifica dovrebbe scalare linearmente con il numero di archi presenti nel grafo; tale proprietà tuttavia si basa ancora una volta sul grado medio $\bar{d}$. Idealmente, per limitare la ridondanza, si vorrebbe che il grafo in ricezione avesse un solo *check node* di grado uno ad ogni successiva iterazione. Si ricava quindi la seguente distribuzione anche detta, *distribuzione solitone ideale*:

$$\rho(d) = \begin{cases} \dfrac{1}{K} & d = 1 \\ \dfrac{1}{d(d-1)} & d = 2, \cdots, K \end{cases} \tag{3.6}$$

grazie alla quale $\bar{d} \approx \ln K$. Tuttavia nella pratica tale distribuzione dà scarsi risultati, perchè le fluttuazioni intorno al valore atteso sono tali da far fallire il processo di decodifica: sono tali cioè da non permettere la creazione di *check node* di grado 1 ad ogni iterazione successiva del processo di decodifica.
La *distribuzione solitone robusta* introduce i parametri c e δ per ovviare a tale problema: essa è progettata, cioè, per fornire un numero medio di *check node* di grado uno pari a:

$$S \equiv c \ln\left(\frac{K}{\delta}\right)\sqrt{K} \tag{3.7}$$

invece che 1, durante tutto il processo di decodifica. Il parametro δ è un limite alla probabilità che la decodifica fallisca dopo la ricezione di K' pacchetti codificati.

Il parametro c è una costante di ordine 1. Si definisca $\tau(d)$ la funzione positiva:

$$\tau(d) = \begin{cases} \frac{s}{K} \cdot \frac{1}{d} & d = 1, 2, \cdots, (K/S) - 1 \\ \frac{s}{K} \log(S/\delta) & d = K/S \\ 0 & d > K/S \end{cases} \tag{3.8}$$

a tale funzione poi si sommi la *distribuzione ideale solitone* e si normalizzi tale somma per ottenere:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z} \tag{3.9}$$

dove $Z = \sum_d (\rho(d) + \tau(d))$. Il numero di pacchetti codificati richiesti in ricezione per ottenere una corretta decodifica con probabilità di almeno $1 - \delta$ deve essere $K' = KZ$.

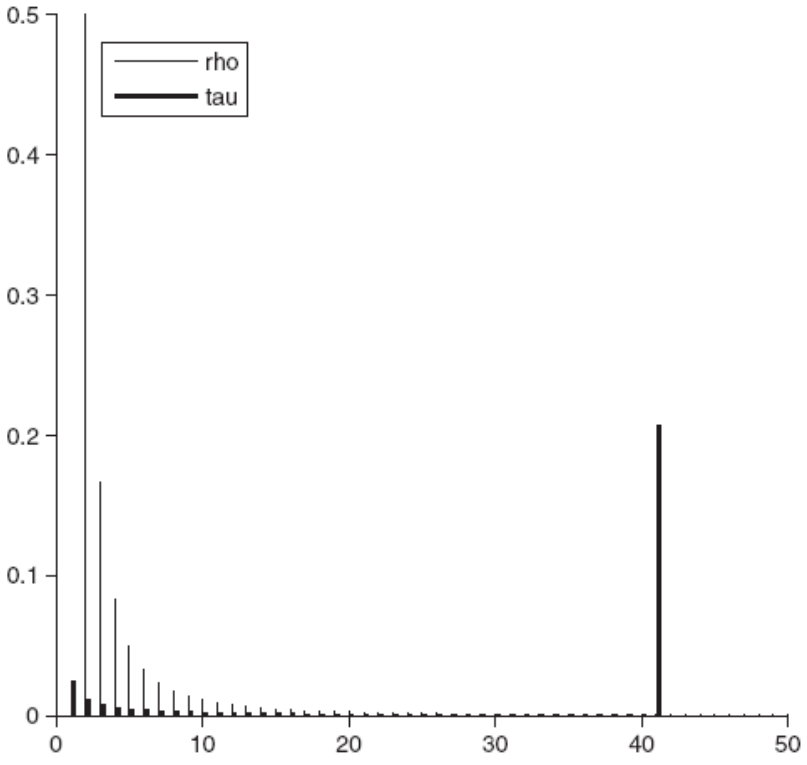

Figura 3.5: Le distribuzioni $\rho(d)$ e $\tau(d)$ per il caso $K = 10000$, $c = 0 : 2$, $\delta = 0.005$, che danno $S = 244$, $K/S = 41$, $Z \simeq 1.3$. La distribuzione $\tau(d)$ è maggiore per $d = 1$ e per $d/K = S$.

L'analisi di Luby [11] spiega quale sia il ruolo della funzione $\tau(d)$ per valori piccoli di d: essa ha il ruolo di permettere l'inizio della fase di decodifica, mentre l'impulso presente a $d = K/S$ assicura che ciascun pacchetto della sorgente abbia un arco nel grafo in ricezione. Il risultato chiave ottenuto da Luby è che la ricezione di $K' = K + 2\ln(S/\delta)S$ pacchetti codificati assicuri la corretta decodifica con probabilità $1 - \delta$.

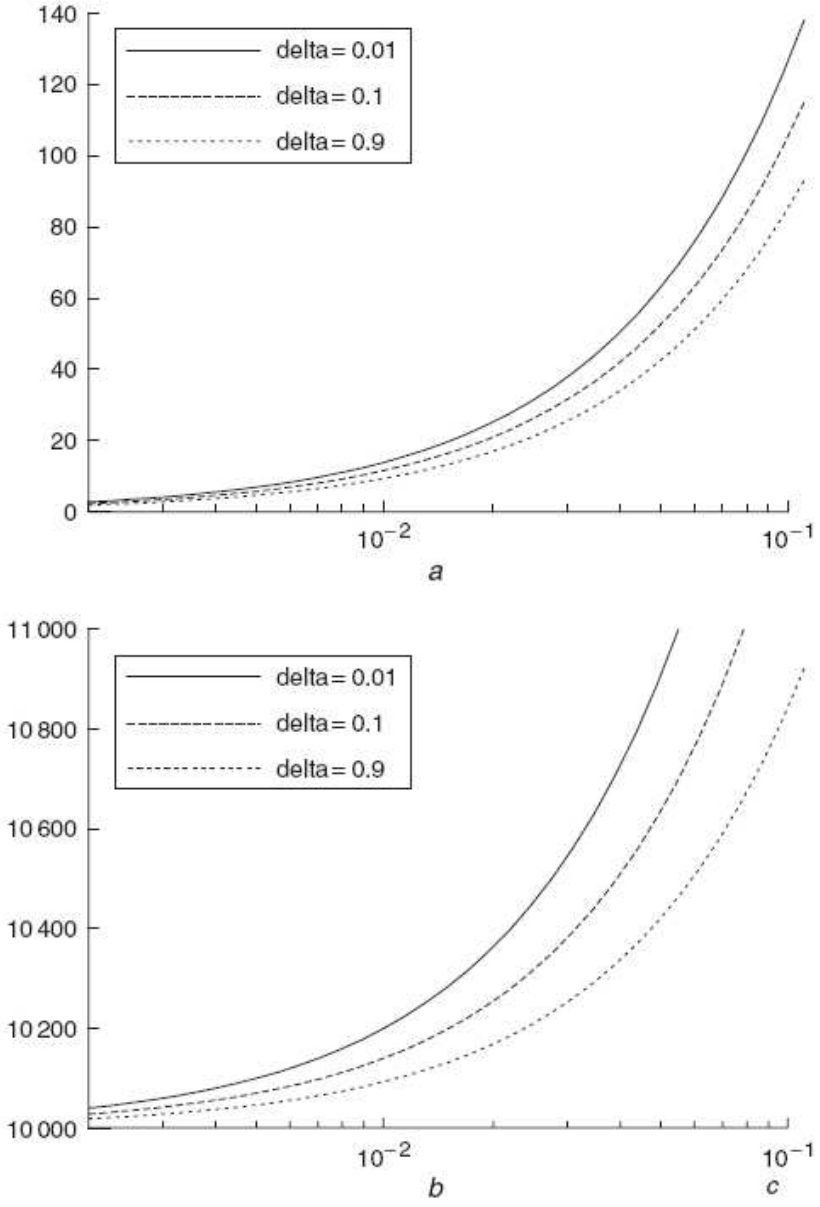

Figura 3.6: Il numero di *check node* di grado 1 S (riquadro a) e la quantità K' (riquadro b) al variare di c e δ, per $K = 10000$.

Nella pratica, i codici LT possono essere configurati in modo tale un file di dimensioni $K \simeq 10000$ pacchetti possa essere recuperato con un *overhead* di circa il 5%. La Figura 3.7 mostra gli istogrammi del numero di pacchetti richiesto per alcune configurazioni dei parametri.
La Figura 3.8 infine illustra gli andamenti temporali di tre tentativi di decodifica. È caratteristica di buon codice LT che il processo di decodifica non parta a pieno regime se non dopo aver raggiunto un valore di N leggermente maggiore di K.

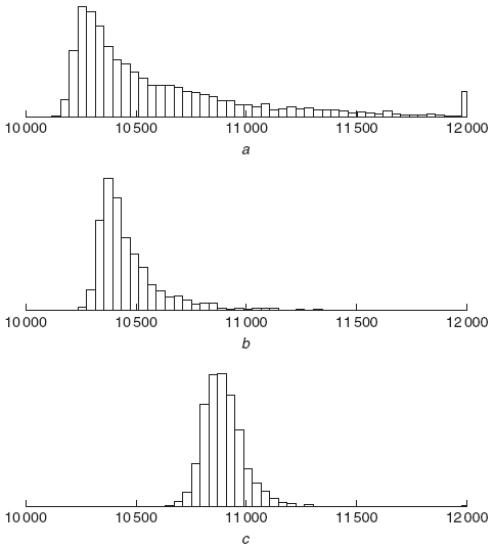

Figura 3.7: Numero di pacchetti codificati N richiesti per la corretta decodifica di un file di dimensioni $K = 10000$ pacchetti. Riquadro a: $c = 0.01$, $\delta = 0.5$, $S = 10$, $K/S = 1010$, $Z \simeq 1.01$. Riquadro b: $c = 0.03$, $\delta = 0.5$, $S = 30$, $K/S = 337$, $Z \simeq 1.03$. Riquadro c: $c = 0.1$, $\delta = 0.5$, $S = 99$, $K/S = 101$, $Z \simeq 1.1$.

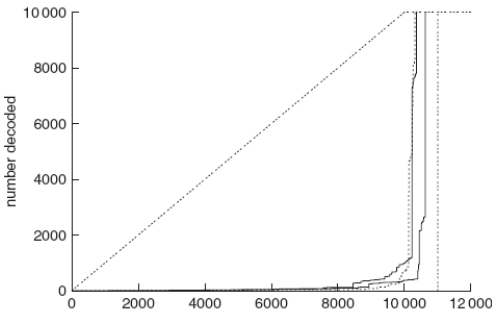

Figura 3.8: Sono illustrati in figura tre risultati sperimentali, tutti per codici LT con parametri $c = 0.03$, $\delta = 0.5$, $S = 30$, $K/S = 337$, $Z \simeq 1.03$ e $K = 10000$. L'asse verticale rappresenta il numero di pacchetti decodificati in funzione del numero di pacchetti ricevuti. La linea verticale a destra è a $N = 11000$, ad un *overhead* cioè del 10%

Capitolo 4

Il *broadcast* nelle reti di sensori *wireless*

4.1 Introduzione

Il *broadcast* efficiente nelle reti è stato ed è tutt'ora un tema di grande importanza. Esso infatti è essenziale per il corretto funzionamento della rete stessa: si pensi ad esempio al caso della riprogrammazione dei nodi nelle reti di sensori, o della tempestiva comunicazione di informazioni essenziali alla sopravvivenza della rete stessa, o ancora alla progettazione di protocolli di routing per reti mobili di sensori etc.
I fattori chiave cui un protocollo di *broadcast* deve aderire il più possibile sono:

- affidabilità;
- efficienza energetica;
- bassa complessità computazionale;
- scalabilità;
- bassi tempi di latenza.

Definiti i termini di paragone, si procede quindi ad un sintetico confronto tra i protocolli di *broadcast* nelle reti di sensori radio.

4.2 Caratteristiche comuni

4.2.1 Il *Jitter*

Si consideri il caso in cui il nodo sorgente trasmetta un pacchetto di *broadcast*. Poichè la velocità di propagazione è quella delle onde elettromagnetiche nell'aria, tutti i nodi vicini ricevono quasi simultaneamente tale trasmissione. Se si assume che tutti i nodi abbiano lo stesso *hardware* con simili tempi di elaborazione, tali nodi procedono al reinoltro del pacchetto istantaneamente. Per risolvere tale problema, i protocolli introducono dei ritardi uniformi nello *scheduling* delle ritrasmissioni dei pacchetti, o *jitter*. Tale sfasamento permette ad un nodo di ottenere il canale per primo, mentre i nodi rimanenti rilevano il canale occupato e deferiscono l'inoltro del messaggio.

4.2.2 L'intervallo aleatorio di stima (*RAD*)

Molti protocolli attuali richiedono che un nodo tenga traccia dei pacchetti ridondanti ricevuti durante un breve intervallo temporale, al fine di decidere se effettuare o no il *broadcast* stesso. Tale lasso temporale, detto Intervallo aleatorio di stima (RAD[1]), è scelto casualmente da una distribuzione uniforme $u \in \mathcal{U}(0, T_{max}]$ dove T_{max} è il massimo intervallo temporale possibile.
Il RAD è quindi in grado di prevenire eventuali collisioni in trasmissione. Un aspetto critico è la sua progettazione. Un primo approccio è quello di inviare i pacchetti di *broadcast* allo strato MAC dopo un breve tempo aleatorio: in questo caso i pacchetti rimangono in attesa nella coda dell'interfaccia di rete (IFQ[2]) finchè lo strato MAC non ottiene il controllo del canale. Durante l'attesa è possibile che il nodo riceva altri pacchetti ridondanti permettendo così allo strato di rete di decidere. In caso di decisione negativa, lo strato di rete informa lo strato MAC che provvede a scartare i pacchetti in coda. Un secondo approccio è quello di utilizzare un intervallo maggiore del caso precedente, e trattenendo i pacchetti nello strato di rete fino alla scadenza del *timer*. Come prima la valutazione avviene al termine del RAD, allo scadere del quale i pacchetti vengono o inoltrati allo strato MAC o scartati: non è possibile quindi che lo strato di rete possa rimuovere dei

[1] Random Assessment Delay
[2] InterFace Queue

pacchetti in coda precedentemente inoltrati allo strato sottostante.

4.3 Categorizzazione dei protocolli

Sono stati considerati e classificati dodici protocolli di broadcast secondo le categorie seguenti [15]:

***Flooding* semplice:** richiede che tutti i nodi reinoltrino tutti i pacchetti ricevuti.

Protocolli probabilistici: utilizzano una semplice conoscenza della topologia di rete per assegnare una probabilità di inoltro a ciascun nodo.

Protocolli geografici: presuppongono che i nodi abbiano una distanza di trasmissione comune; i nodi potranno inoltrare i pacchetti ricevuti solo se tale operazione permetterà di aumentare l'area di copertura in maniera adeguata.

Protocolli a conoscenza dei vicini: essi mantengono una conoscenza della topologia della rete tramite pacchetti di controllo, tramite quest'ultimi sono prese le decisioni relative alla fase di reinoltro.

Si noti che le categorie sono riportate in ordine crescente di complessità e di requisiti di risorse. Lo scopo dei costi aggiuntivi di ciascuna classe rispetto alla precedente è quello di ridurre il numero di ritrasmissioni ridondanti.

4.3.1 *Flooding* semplice

Il primo algoritmo presentato è il più primitivo. Tale protocollo prevede che la sorgente inoltri un pacchetto ai suoi vicini. Ciascuno di essi a turno reinoltra tale pacchetto una sola volta. La sequenza così descritta continua fino a che tutti nodi della rete raggiungibili hanno ricevuto tale pacchetto.
Il *Flooding* Semplice è stato proposto come tecnica di *broadcast* e di *multicast*[3] nelle reti altamente dinamiche, e con bassa densità di nodi.

[3]Inoltro dei dati da una sorgente ad un sottoinsieme della rete

4.3.2 Protocolli probabilistici

Schema Probabilistico

Tale protocollo è simile al *Flooding* Semplice: i nodi procedono all'inoltro con una probabilità determinata. Nelle topologie dense è probabile che alcuni nodi abbiano in comune lo stesso set di nodi ricevitori; per questo motivo tale protocollo permette un risparmio di risorse senza compromettere l'affidabilità del *broadcast* stesso. Tuttavia nelle reti sparse tale eventualità è più rara: in questi casi ,infatti, è preferibile impostare una probabilità di inoltro alta per permettere la copertura dell'intera rete.

Schema basato su timer

Le metodologie a timer sfruttano la relazione inversamente proporzionale tra il numero di ricezioni di un determinato pacchetto e la probabilità che tale nodo sia in grado di coprire con il suo inoltro un'ampia area utile di rete.
Alla ricezione di un *nuovo* pacchetto p_i, il nodo inizializza un nuovo contatore $c_{p_i} = 1$ e setta il timer RAD. Durante il RAD il contatore c_{p_i} è incrementato di una unità per ciascuno pacchetto ridontante p_i' ricevuto. Allo scadere del RAD se il contatore c_{p_i} è minore di un valore predefinito c_{soglia} il pacchetto viene reinoltrato. In caso contrario il pacchetto viene scartato. Studi approfonditi hanno accertato che valori di $c_{soglia} > 6$ non portano ad alcun beneficio aggiuntivo. Anche nel caso degli schemi basati su timer si otterà il comportamento descritto in precedenza: in reti altamente popolate alcuni nodi non procedono al reinoltro, mentre in reti sparse è molto probabile che tutti i nodi ritrasmettano.

4.3.3 Protocolli geografici

Nei protocolli geografici i nodi sfruttano la conoscenza della topologia della rete per ritrasmettere i dati in maniera efficiente. Si consideri ad esempio un nodo x: esso riceve un pacchetto da un vicino y posto ad una distanza di un metro. Se x reinoltrasse tale pacchetto l'area di copertura aggiuntiva rispetto a quella già ottenuta da y sarebbe molto bassa. All'estremo opposto, se x fosse collocato sul bordo dell'area coperta da una trasmissione di y allora un'eventuale sua ritrasmissione porterebbe ad un aumento di superficie coperta decisamente maggiore. Tali protocolli sfruttano allora la ricezione dei pacchetti ridondanti per valutare

l'area di copertura addizionale. Si fa notare tuttavia come non si consideri il fatto che esistano o meno nodi in tale area.

Schemi basati sulla distanza

Un nodo appartenente a tale categoria paragona la distanza che corre tra se stesso ed il nodo da cui ha ricevuto un pacchetto di broadcast[4]. Alla ricezione di un pacchetto mai ricevuto si inizializza un timer *RAD* e si rimane in attesa di eventuali pacchetti ridondanti. Allo scadere del timer in questione si esaminano tutte le distanze dei pacchetti ricevuti alla ricerca di una distanza d_i minore di una distanza soglia d_{soglia}. In questo caso, il nodo non procede al reinoltro dei dati.

Schemi basati sulla posizione

Tali schemi usano una stima più precisa dell'area aggiuntiva in copertura per la decisione sul reinoltro dei dati: ciascun nodo infatti deve essere in grado di determinare autonomamente la propria posizione. Ogni volta che un nodo trasmette un pacchetto aggiunge la propria posizione nell'header del pacchetto stesso. In ricezione il nodo preleva la posizione del trasmettitore e procede al calcolo dell'area di copertura aggiuntiva. Se tale valore è minore del valore di soglia il nodo non procede al reinoltro e tutte le future ricezione dello stesso pacchetto verrano scartate a priori. In caso contrario, esso procederà all'inizializzazione del timer *RAD*, allo scadere del quale ripeterà la procedura di stima per ogni pacchetto ridondante ricevuto. Tale procedura sarà ripetuta fino a che il pacchetto non viene scartato o non viene trasmesso.

4.3.4 Protocolli ad identificazione dei vicini

Flooding* con *Self Pruning

Il protocollo in questione richiede che ciascun nodo abbia una lista dei propri vicini ad un passo tramite degli invii periodici di pacchetti di controllo. Ad ogni trasmissione il nodo trasmettitore include nell'header del pacchetto inoltrato la lista sopra citata. In ricezione si procede ad un confronto tra la lista dei vicini

[4]gli autori di [16] hanno precisato che a tale scopo può essere utilizzata la potenza del segnale ricevuto; non è quindi strettamente necessario l'utilizzo di dispositivi *GPS*.

ricevuta e quella in proprio possesso. Se il nodo conclude che la propria eventuale ritrasmissione del pacchetto non porterebbe alla copertura di alcun nodo aggiuntivo scarta il pacchetto, in caso contrario procede alla sua trasmissione.

Scalable Broadcast Algorithm (*SBA*)

Il protocollo *SBA* richiede che ciascun nodo abbia una lista dei propri vicini a due passi. Tale lista in congiunzione con l'identità del nodo trasmettitore del pacchetto permette al nodo ricevente di determinare se la propria ritrasmissione sarebbe utile o meno. La lista dei nodi vicini a due passi può essere ottenuta tramite l'invio periodico di pacchetti di controllo: ciascuno di essi contiene infatti l'identità unica del nodo trasmettitore e la sua lista. L'invio e la conseguente ricezione dei pacchetti di controllo dai propri vicini è possibile ottenere la lista cercata.
Si supponga che il nodo b riceva un pacchetto di broadcast dal nodo a. Poichè il nodo a è un vicino, b conosce tutti i vicini in comune con a che hanno ricevuto lo stesso pacchetto in questione. Se b ha ulteriori nodi ai quali trasmettere, esso programma la trasmissione del pacchetto allo scadere di timer *RAD*. Se b riceve un pacchetto ridondante da un altro vicino, la procedura si ripete. Il processo descritto continua finchè non scade il timer *RAD* ed il pacchetto è spedito, oppure finchè il pacchetto viene scartato.

Dominant Pruning

Tale algoritmo utilizza anch'esso la conoscenza dei vicini a due passi. Diversamente dall'algoritmo *SBA*, esso richiede che i nodi trasmettitori selezionino un set (anche tutto l'insieme) dei vicini ad un passo. Solo tale set è abilititato al successivo reinoltro del pacchetto trasmesso. I nodi appartenenti al set in questione vengono notificati della scelta grazie al proprio identificativo presente nell'header del pacchetto dati ricevuto.
Alla ricezione di un pacchetto dati, un nodo provvede quindi al controllo della presenza del proprio identificativo nell'header del pacchetto; se questo è il caso, utilizza un algoritmo *Greedy Set Cover* per determinare quale subset dei propri vicini ad un passo debba ritrasmettere, sapendo quali nodi sono stati già coperti dal nodo trasmettitore. L'algoritmo *Greedy Set Cover* sceglie ricorsivamente tra

i vicini ad un passo quelli che coprono il maggior numero di nodi a due passi, e ricalcola il set di copertura finchè tutti i nodi a due passi sono collegati al grafo.

Inoltro Multipunto (*Multipoint Relaying*)

Il protocollo di Inoltro Multipunto è simile al protocollo di *Dominant Pruning*. Anche in questo caso infatti i nodi abilitati al reinoltro sono scelti esplicitamente dai nodi trasmettitori. Tuttavia la procedura di selezione del set dei nodi ad un passo abilitati alla ritrasmissione è la seguente:

1. si identificano tutti i vicini a due passi che possono essere raggiunti che possono essere raggiunti solamente dai vicini ad un passo. Tali vicini ad un passo sono selezionati;

2. si determina il set di copertura risultante;

3. tra i nodi ad un passo non selezionati al punto (1), si cercano quelli che coprirebbero il maggior numero di vicini a due passi non appartenenti al set di copertura del punto (2);

4. si itera dal punto (2) fino a che tutti i nodi a due passi non sono coperti.

Ad Hoc Broadcast Protocol (*AHBP*)

Il protocollo *Ad Hoc Broadcast* utilizza un approccio simile al protocollo di Inoltro Multipunto. Anche la procedura di selezione del set dei nodi ad un passo abilitati al reinoltro è la medesima. *AHBP* differisce dall'inoltro Multipunto in tre punti:

1. Un nodo che utilizzi *AHBP* informa tutti i nodi ad un passo abilitati inserendo il loro identificativo nell'header del pacchetto trasmesso. Al contrario il protocollo d'inoltro multipunto informa tali nodi tramite pacchetti di controllo.

2. In *AHBP*, quando un nodo riceve un pacchetto di *broadcast* ed è stato selezionato alla ritrasmissione, tale nodo utilizza la propria conoscenza dei nodi a due passi per determinare quali tra i propri vicini abbiano ricevuto lo stesso pacchetto. Tali nodi sono considerati già coperti e sono rimossi dal grafo utilizzato per la selezione dei prossimi nodi abilitati alla ritrasmissione.

3. *AHBP* è progettato per operare su reti di sensori ad alta mobilità. Si supponga che il nodo a riceva un pacchetto di *broadcast* dal nodo b, e a non abbia b nella propria tabella dei vicini a due passi. In *AHBP* il nodo a si auto-abilita alla ritrasmissione.

Algoritmo di Broadcast basato su *CDS* (*CDS-Based Broadcast Algorithm*)

Tale protocollo si basa sul protocollo *AHBP*. Tuttavia mentre quest'ultimo considera solo la sorgente del pacchetto di broadcast ricevuto per determinare il set di nodi ad un passo abilitati al reinoltro, l'algoritmo *CDS*[5] considera anche il set precedentemente calcolato da nodi di livello superiore nell'albero di *broadcast*. Ad esempio si supponga che il nodo a abbia selezionato i nodi b, c, d in quest'ordine. Alla ricezione del pacchetto di broadcast il nodo c il protocollo *CDS* richiede che per la creazione del set iniziale c aggiunga i vicini di a ed *anche* di b in quanto b è di livello superiore. Allo stesso modo il nodo d dovrà aggiungere i vicini di a, b, c. Una volta stabilito il set iniziale l'algoritmo prosegue alla selezione dei nodi vicini abilitati al broadcast come farebbe il protocollo *AHBP*.

Lightweight and Efficient Network-Wide Broadcast (*LENWB*)

Il protocollo *LENWB* richiede anch'esso la creazione di una tabella dei vicini a due passi ottenuta tramite scambio di pacchetti di controllo. Tuttavia invece di scegliere esplicitamente quali nodi abilitare per le successive ritrasmissioni, tale scelta è effettuata implicitamente. In *LENWB*, ogni nodo decide di ritrasmettere basandosi sulla conoscenza dei nodi vicini ad un passo che sono potenzialmente in grado di ritrasmettere. Per ottenere ciò è necessario sapere quali vicini abbiano ricevuto un pacchetto dalla sorgente comune e quali vicini hanno una prorità maggiore di ritrasmissione. Tale priorità è proporzionale al numero di vicini: maggiore è il numero, maggiore è la priorità. Poichè un nodo è abilitato alla ritrasmissione dai nodi vicini di più alto livello, esso può proattivamente calcolare se tutti i propri vicini di livello inferiore saranno in grado di ricevere quelle ritrasmissioni; se ciò non avvenisse esso procederebbe al *broadcast* dei dati.

[5]Connected Dominating Set

4.4 Protocolli con codici a fontana

In questa sezione si passeranno in rassegna due protocolli che sfruttano i codici a fontana per il *broadcast*: il protocollo *CRBCast*, ed il protocollo *FTS*[6]. Il primo protocollo sarà ripreso nei capitoli successivi in quanto oggetto di analisi e confronto.

4.4.1 Il protocollo *CRBCast*

Il protocollo *CRBCast*[7] unisce l'efficienza dei codici a fontana ad un algoritmo di *broadcast* semplice e scalabile: il protocollo *PBCast*[8]. Esso appartiene alla categoria precedentemente descritta dei protocolli di broadcast probabilistici: ogni nodo infatti decide con probabilità p se procedere all'inoltro dei pacchetti dati ricevuti, riducendo così la trasmissione di pacchetti ridondanti ed il consumo energetico.

Analisi asintotica del protocollo *PBCast*

Si procede ora ad un'analisi essenziale delle prestazioni asintotiche del *PBCast*. Per un'analisi completa si faccia riferimento a [14].
In *PBCast* ogni nodo inoltra un pacchetto una ed una sola volta con probabilità p. Si assuma che ogni nodo della rete che proceda a tale inoltro sia colorato di nero, in caso contrario sia colorato di bianco. Sia B il set dei *nodi neri* e W il set dei *nodi bianchi*. Sia $G_B(p,r) = G(N,r) \setminus W$ il sottografo di $G(N,r)$ indotto da B, dove N è il numero di nodi della rete, r è il *range* di trasmissione uguale per ciascun nodo.
Si consideri il caso del *broadcast* di n_p pacchetti. Sia R_1 la frazione di nodi che abbiano ricevuto un determinato pacchetto durante la fase *PBCast*. Se i nodi sono disposti uniformemente R_1 denota la probabilità che un nodo della rete riceva il determinato pacchetto. Poichè le trasmissioni degli n_p pacchetti sono indipendenti la probabilità che un nodo riceva tutti gli n_p pacchetti, sotto queste ipotesi, è $R_1^{n_p}$. Tuttavia non è possibile assumere plausibile tale diffusione. Ad esempio i nodi vicini alla sorgente ricevono con probabilità 1 tutti gli n_p

[6]Fractional Transmission Scheme
[7]Collaborative Rateless BroadCAST
[8]Probabilistic BroadCAST

pacchetti, mentre questo non è altrettanto vero per i nodi appartenenti al bordo della topologia. È però possibile calcolare un *bound* per tale probabilità:

$$R_1^{n_p} \leq R_{n_p} \leq R_1 \tag{4.1}$$

Poichè il consumo energetico è proporzionale al numero di tramsissioni, sarebbe desiderabile ottenere un ottimo valore di N_{tx}/n_p, il numero totale di trasmissioni richieste per pacchetto d'informazione. Poichè non tutti i *nodi neri* ricevono un pacchetto da ritrasmettere, il valore N_{tx}/n_p è superiormente limitato dal numero di *nodi neri* più la sorgente: in media, $N_{tx}/n_p = pN + 1$. Inoltre, nell'area descritta dai nodi che ricevono un determinato pacchetto, in media una frazione p di essi sono nodi che trasmettono. Perciò si ottiene:

$$\frac{N_t x}{n_p} = pNR_1 \tag{4.2}$$

La fase I

Nella prima fase gli n_p pacchetti originali d'informazione vengono codificati in $n_p\gamma$ pacchetti, per mezzo della codifica a fontana. I pacchetti così ottenuti sono poi inoltrati su tutta la rete per mezzo del protocollo *PBcast*.
Al termine della prima fase, alcuni nodi, chiamati *nodi completi* avranno ricevuto correttamente un numero n di pacchetti codificati $n_p \leq n \leq n_p\gamma$ tale da permettere la decodifica dei pacchetti originali. I nodi che non ricadono in tale categoria sono detti *nodi incompleti*. Il numero di *nodi completi* al termine della fase I ed il numero di trasmissioni per pacchetto possono essere approssimate da $NR_{n_p\gamma}$ e $N_{tx}/n_p = pNR_1\gamma$ come discusso nella sezione precedente. Il parametro $\gamma \geq 1$ è l'*overhead* imposto dalla codifica a fontana ed è scelto in modo tale che la probabilità di corretta decodifica $P_R(n_p, \gamma) \approx 1$.

La fase II

La seconda fase si basa su una semplice collaborazione tra i *nodi completi* ed *incompleti* in modo tale che ciascun nodo completo trasmetta il numero di pacchetti necessario ai propri vicini *incompleti* per la corretta decodifica. Gli eventuali nuovi nodi completi così creati ripetono la procedura. Il processo continua finchè tutti i nodi della rete sono completi. Sono necessari allo scopo due brevi messaggi di

handshake: i messaggi di avviso (*advertisement*) indicati con ADV ed i messaggi di richiesta (*request*) indicati con REQ.
Non appena un nodo diviene completo spedisce periodicamente un messaggio ADV ai propri vicini. Ciascun nodo incompleto che riceve tale messaggio risponde con un messaggio REQ indicando il numero di pacchetti richiesti. Si consideri ad esempio il caso in cui un *nodo incompleto* abbia già ricevuto n_1 pacchetti; alla ricezione del messaggio ADV esso risponde con un messaggio REQ in cui richiederà $n_p\gamma - n_1$ pacchetti codificati. Si fa notare come i nodi completi, in questa fase, provvedano alla ricodifica dei pacchetti d'informazione già ottenuti, in modo da sfruttare la diversità intrinseca alla codifica a fontana.
Se un nodo incompleto riceve messaggi ADV multipli da più nodi completi vicini, esso risponderà a quello con identificativo più basso. Nel caso in cui un nodo completo riceva messaggi REQ multipli esso invierà il massimo dei pacchetti richiesti. Infine, nel caso in cui un nodo incompleto non riceva risposta ad un messaggio REQ inviato, esso provvede al suo reinvio.

Advertisement **probabilistico nella Fase II**

Nelle reti ad alta densità è possibile introdurre una probabilità p_{adv} per ridurre il numero totale di messaggi ADV inviati.

Recupero a più stadi nella Fase II

Nel protocollo *CRBCast*, ogni *nodo completo* trasmette il massimo numero di pacchetti richiesto. È possibile tuttavia che sia sufficiente un numero minore di pacchetti codificati, perchè è possibile che al round successivo tale nodo sia completato da un altro nodo. Per questo motivo è possibile porre un limite superiore n_{max} al numero massimo di pacchetti codificati trasmissibili.

4.4.2 Il protocollo ***FTS***

Il protocollo *FTS* [14] è un'evoluzione del protocollo *CRBCast*. Esso si basa sull'ipotesi che ogni nodo conosca la distanza in termini di passi sia dalla sorgente sia da ogni nodo della rete. Se, in aggiunta, ciascun nodo conosce la propria posizione, l'efficienza energetica introdotta dal protocollo può essere ulteriormente incrementata. In questo caso infatti ciascun nodo può adattare la potenza di

trasmissione a quella richiesta per una corretta ricezione da parte del vicino più distante. Tale versione del protocollo è detta FTS_{adapt}.
FTS si basa sull'idea che più vicini completi di un nodo incompleto u possono collaborare al completamento di u. È sufficiente, infatti, che ciascun nodo completo spedisca una frazione dei pacchetti codificati richiesti da u.
Si consideri quindi una rete $N(n, r)$ di n nodi statici con raggio di trasmissione r. Per ogni nodo u è possibile definire l'insieme dei vicini come

$$N_r(u) = \{v \mid d(v, u) \leq r,\ \forall v \in N(n, r)\} \tag{4.3}$$

Si definisce infine $H(v)$ la lunghezza del cammino più breve che collega la sorgente s al nodo v.
In *FTS* la direzione di trasmissione lungo un collegamento tra due nodi è quella che va dal nodo che ha minor numero di passi dalla sorgente al nodo con maggior numero di passi. In caso di egual numero di passi la direzione è quella che va dall'identificativo minore a quello maggiore. In questo modo un nodo v si aspetta di ricevere dati dai nodi appartenenti al suo *set dei nodi padre* $\mathcal{P}_r(v)$ definito come:

$$\begin{aligned} \mathcal{P}_r(v) &= \{w \in N_r(v) | H(w) < H(v)\} \\ &\quad \cup \{w \in N_r(v) |\, (H(w) = H(v)) \wedge (id(w) < id(v))\} \end{aligned} \tag{4.4}$$

Allo stesso tempo ogni nodo w è responsabile della trasmissione dei pacchetti codificati all'insieme dei nodi *figli*

$$\mathcal{C}_r(w) = N_r(w) \setminus \mathcal{P}_r(w) \tag{4.5}$$

È necessario quindi determinare la frazione di dati che w deve spedire, α_w.

Lo schema *FTS*

FTS include tre fasi: la *fase iniziale di scambio delle frazioni*, la *fase di riduzione delle frazioni*, la *fase di trasmissione dati*.

La fase iniziale di scambio delle frazioni. Ogni nodo v determina i vicini appartenenti al proprio insieme $\mathcal{P}_r(v)$ di cardinalità $k_v = |\mathcal{P}(v)|$. Da ciascuno di essi v si aspetta una frazione $1/k_v$. Una volta calcolata tale frazione, v la comunica a tutti i nodi appartententi a $\mathcal{P}_r(v)$. Allo stesso tempo ogni nodo w raccoglie le frazioni inviate da tutti i nodi in $\mathcal{C}_w$, ne calcola il massimo

e invia a ciascuno dei nodi in $\mathcal{C}_w$ la frazione di pacchetti pari a tale massimo. Al termine di questa fase, la somma di tutte le frazioni che ciascun nodo riceve potrebbe essere maggiore dell'unità.

La fase di riduzione delle frazioni. In questa fase, ciascun nodo v chiede ai nodi appartenenti a $\mathcal{P}_r(v)$ di ridurre le proprie frazioni di f_v in modo tale che le frazioni totali ricevute da v abbiano somma pari ad uno. Ciascun nodo w riduce le sue frazioni del minimo tra tutte le richieste ricevute ottnendo una nuova frazione che invierà ad i suoi nodi figli α'_w.

La fase di trasmissione dati. In quest'ultima fase, una volta che un nodo v riceve la frazione $\max(\alpha_w - f_v, 0)$ di pacchetti codificati dal vicino w spedisce un messaggio di *acknowledgement* di parziale completezza ($P - ACK(v \rightarrow w)$ai propri vicini; una volta completato invece trasmetterà un messaggio di *acknoledgement* di completezza ($C - ACK(v)$). Esso quindi procede alla decodifica dell'informazione originale e alla ricodifica di nuovi pacchetti e inizierà la trasmissione di questi ultimi fino a che tutti i suoi figli non sono o completi o non necessiteranno più di pacchetti da v.

Analisi dei vari *overhead*

FTS ha tre messaggi che introducono *overhead* e tempi di latenza. Tuttavia se la rete è considerata statica, il costo dovuto alla prima ed alla seconda fase è sostenuto una ed una sola volta. Nello specifico, la determinazione della distanza dalla sorgente in numero di passi, ed le frazioni iniziali possono essere completati in un tempo $O(n)$ con costo energetico $O(1)$ per ogni nodo. La riduzione delle frazioni invece può essere completata in $O(nr^2(n))$ ad un costo energetico anche in questo caso costante. L'ultimo *overhead* ed anche il più importante è quello dovuto ai messaggi di *acknowledgement* parziali: questi devono essere trasmessi una volta sola e sono necessari solo se si vuole che un nodo smetta di trasmettere una volta che tutti i suoi nodi figli abbiano ricevuto tutti i pacchetti dati pattuiti, prima di spedire tutte le quote di pacchetti codificati. Si può dunque ottenere un risparmio di energia durante la trasmissione delle frazioni dati permettendo che ogni nodo trasmetta una quantità di pacchetti di controllo dell'ordine di $O(nr^2(n))$.

Capitolo 5

Broadcast nelle reti acustiche sottomarine

5.1 Introduzione

Scopo di questa tesi è quello di porre le basi per un protocollo di *ARQ Ibrido* (*HARQ*) basato sui codici a fontana per broadcast in reti acustiche sottomarine. Tale protocollo, chiamato *OFBP*[1] prevede la collaborazione tra i nodi della rete per ottenere la massima affidabilità ed efficienza energetica. A tale scopo si sono progettate delle *policy* ottime di ritrasmissione dei pacchetti codificati e le si sono confrontate con delle *policy* a ridondanza costante.
Si è successivamente analizzato il rendimento del protocollo *CRBCast* nelle reti acustiche sottomarine, si sono apportate delle modifiche alla fase I ed alla fase II, e lo si è confrontato con *OFBP*. Si è deciso per il confronto con il protocollo *CRBCast* perchè esso è relativamente semplice e richiede una trasmissione di un numero minore di pacchetti di controllo rispetto ad un protocollo come *FTS*. Tale requisito è essenziale nelle reti acustiche, dove la velocità di propagazione è ridotta.

[1] Optimized Fountain-based Broadcast Protocol

5.2 Ipotesi preliminari

5.2.1 Il canale sottomarino, lo strato fisico e *MAC*

Per la modellazione e la simulazione del canale sottomarino si sono considerati i valori tipici di utilizzo per il *fattore di spreading* k, per la temperatura, la salinità e la velocità del vento e sono state utilizzate le formule empiriche esposte nel primo capitolo per il rumore ambientale. Per il calcolo dell'SNR si è considerata la formula seguente:

$$SNR(d, B(d)) = \frac{\frac{P_T}{B(d)} \int_{B(d)} A^{-1}(d, f) df}{\int_{B(d)} N(f) df} \tag{5.1}$$

dove d è la distanza, $B(d)$ è l'ampiezza di banda alla distanza d, P_T è la potenza in trasmissione, $A^{-1}(d, f)$ è il fattore attenuativo già discusso. e la sua formula approssimata:

$$\begin{aligned} SNR(d, B(d)) &= \frac{\frac{P_T}{B(d)} \int_{B(d)} A^{-1}(d, f_0) df}{\int_{B(d)} N(f_0) df} \\ &= \frac{\frac{P_T}{B(d)} \cdot B(d) A^{-1}(d, f_0)}{B(d) \cdot N(f_0)} \\ &= \frac{P_T A^{-1}(d, f_0)}{B(d) N(f_0)} \end{aligned} \tag{5.2}$$

dove f_0 è la frequenza di centro banda, e si è ipotizzato che l'attenuazione ed il rumore siano costanti nella banda d'interesse. Si è compiuta successivamente un confronto tra le precisioni dei risultati forniti dalla (5.1) e dalla (5.2) e si è riscontrato una perdita di precisione di ordine trascurabile. Si è dedotto quindi che le ipotesi di banda stretta siano empiricamente verificate per le distanze di utilizzo e si è conseguentemente optato per il calcolo dell'SNR secondo (5.2).
I sensori sono supposti essere equipaggiati di trasduttori atti alla modulazione *BPSK*. Per tale modulazione la probabilità di errore p su un pacchetto di b bit alla distanza d è data da:

$$p = \left(1 - \frac{1}{2} erfc\sqrt{SNR(d, B(d))}\right)^b \tag{5.3}$$

in ipotesi di rumore additivo bianco gaussiano ($AWGN$[2]). Poichè l'ipotesi di banda stretta è stata empiricamente verificata, anche tale formula può essere considerata valida. In queste ipotesi si presuppone che la potenza di trasmissione P_T di ciascun nodo sia quella richiesta per ottenere una $p = 0.25$ ad una distanza d_{tx}.
Si sono utilizzate le seguenti configurazioni in trasmissione:

d_{tx}	$B(d_{tx})$	f_0	$SNR(d_{tx}, f_0)$	P_T
2500 m	15.6 kHz	12.4 kHz	5.9	$1.8 \cdot 10^{13}$ $dB\ re\ \mu Pa$
5000 m	11 kHz	8.5 kHz	5.9	$7.5 \cdot 10^{13}$ $dB\ re\ \mu Pa$
7500 m	8.9 kHz	6.8 kHz	5.9	$1.7 \cdot 10^{14}$ $dB\ re\ \mu Pa$

Figura 5.1: Configurazione di trasmissione utilizzate tali per cui $p = 0.25$

Per semplicità simulativa si è ipotizzato che la probabilità di errore sul pacchetto p fosse la seguente:

$$p = \begin{cases} p & d \leq d_{tx} \\ 1 & d > d_{tx} \end{cases} \tag{5.4}$$

in altre parole si è considerato che la trasmissione al di fuori dell'area descritta dal cerchio di raggio d_{tx} fallisca sempre: tale ipotesi è verosimile, poichè la probabilità di errore sul pacchetto cresce molto rapidamente all'aumentare della distanza, ed è ≈ 1 già per distanze poco maggiori di d_{tx} . Si è ipotizzato infine che il canale di *feedback* sia esente da errori.

5.2.2 Topologie di rete

Si è considerata una rete acustica di sensori planare circolare di raggio R_{rete} e di profondità 1000 m. I nodi, esclusa la sorgente (posizionata al centro), sono stati distribuiti all'interno della rete casualmente in accordo con le realizzazioni di un *processo di Poisson bidimensionale* di parametro λ. Si definisce λ come il numero medio di nodi presenti nell'area del cerchio descritto dalla distanza di trasmissione d_{tx}:

$$\lambda = E\left[N_{nodi}(d_{tx})\right] \tag{5.5}$$

[2]Additive White Gaussian Noise

Dato λ, è possibile determinare il numero medio di nodi presenti nella topologia:

$$\begin{aligned} N_{nodi}(\lambda, R_{rete}, d_{tx}) &= \lambda \cdot \left(\frac{\pi R_{rete}^2}{\pi d_{tx}^2} \right) \\ &= \lambda \cdot \left(\frac{R_{rete}}{d_{tx}} \right)^2 \end{aligned} \tag{5.6}$$

Le configurazioni di λ, d_{tx} e R_{rete} utilizzate sono le seguenti:

id	d_{tx}	R_{rete}	λ	N_{nodi}
1	2500 m	7500 m	2.5	23
2	2500 m	7500 m	5	45
3	2500 m	7500 m	10	90
4	2500 m	7500 m	20	180
5	5000 m	15000 m	2.5	23
6	5000 m	15000 m	5	45
7	5000 m	15000 m	10	90
8	5000 m	15000 m	20	180
9	7500 m	22500 m	2.5	23
10	7500 m	22500 m	5	45
11	7500 m	22500 m	10	90
12	7500 m	22500 m	20	180

Figura 5.2: Topologie di rete utilizzate

Si noti come si sia scelto R_{rete} esattamente il triplo d_{tx}: si è voluto in questo modo creare delle topologie aventi 3 come minimo numero di *hop*.

5.3 Uso di codici a fontana per il *broadcast*

5.3.1 Descrizione del protocollo progettato

Codifica e decodifica dell'informazione

Si consideri la trasmissione *broadcast* di un file dati di dimensioni note. Esso verrà suddiviso in N_{blk} blocchi di K pacchetti di dimensioni s_{pkt} . I pacchetti verranno codificati secondo le specifiche dei codici a fontana in N_{blk} di N_{pkt} delle stesse dimensioni. I dati così codificati verranno spediti un blocco dopo l'altro, previa la corretta decodifica del blocco precedente da parte dell'*intera rete*. Si ricorda che la corretta decodifica può avvenire solo se si è ricevuto correttamente un numero di pacchetti

$$K \leq N_{pkt,rx} \leq N_{pkt} \tag{5.7}$$

Trasmissione e ritrasmissione di un blocco

Si consideri un nodo s e l'insieme dei suoi ricevitori $\mathcal{R}_s$: come specificato nelle sezioni precedenti si assume che quest'ultimi siano forzatamente all'interno dell'area descritta dal cerchio avente raggio d_{tx}, ovvero:

$$\mathcal{R}_s = \{i \in N_{rete} | d(i,s) \leq d_{tx}\} \tag{5.8}$$

Sfruttando la propagazione quasi-sferica del canale sottomarino è possibile trasmettere in sequenza l'intero blocco una sola volta seguito da un messaggio di controllo che informa i ricevitori del termine del blocco; dopo ogni tempo di progazione τ_i, $i \in \mathcal{R}_s$ ogni nodo riceve in sequenza i pacchetti aventi $SNR \geq SNR(d_{tx}, f_0)$ dove $SNR(d_{tx}, f_0)$ appartiene alla tabella di Figura 5.2. Alla fine della fase di ricezione il nodo $i \in \mathcal{R}_s$ tenta la decodifica ed invia un messaggio di controllo contenente il rango residuo $Rank_{res}$ ed il massimo numero di pacchetti ricevuti N_{ack}.
La sorgente colleziona quindi tali statistiche dai ricevitori e tenterà o di completare i ricevitori incompleti secondo un algoritmo descritto nel seguito.

Algoritmo di *broadcast*

Definite la fase di codifica e di trasmissione ad un passo, si vuole ora definire la procedura che consente l'inoltro efficiente dell'informazione originale a tutta la rete:

1. ogni nodo completo invia un messaggio di controllo detto *SYNC*, con cui avvisa i propri vicini in $\mathcal{R}_s$ della propria disponibilità ad instaurare una sessione di inoltro[3].

2. i nodi incompleti alla ricezione di un messaggio *SYNC* inizializzano un timer *RAD* allo scadere del quale rispondono una ed una sola volta tramite un messaggio di controllo *SYNC ACK* al nodo più vicino: grazie alla bassa velocità di propagazione del suono è possibile attraverso stimare la distanza tramite la relazione:
$$d = \frac{t_{gen} - t_{rx}}{v_{suono}} \tag{5.9}$$
dove t_{gen} è il tempo di generazione del pacchetto, presente nell'*header* dello stesso, e t_{rx} è il suo tempo di ricezione.

3. Tramite la fase *SYNC* - *SYNC ACK* si instaura una sessione di trasmissione. **Eventuali nuovi nodi completi devono attendendere un tempo aleatorio prima di poter a loro volta instaurare una nuova sessione**. Grazie a questa direttiva infatti è possibile desincronizzare i nodi completi, in modo da non interferire con eventuali altre sessioni parallele.

4. si itera al punto (1) fino al completamento dell'intera rete.

In un ambiente reale è possibile che si formino delle *direzioni di propagazione* privilegiate, ad esempio dove la probabilità di errore sul pacchetto fosse minore. Tuttavia per rendere la simulazione fattibile ed ottenere un *upper bound* alle prestazioni del protocollo, si è preferito controllare artificialmente tale diffusione sulla base del numero di *hop*: ogni nodo completato infatti entra a fare parte dell'*hop* successivo e non può instaurare alcuna sessione se non al termine di tutte le sessioni appartenti all'*hop* corrente. La diffusione macroscopica media è quindi approssimabile per cerchi concentrici o *tier*. Si faccia riferimento alla

[3]Nel proseguio della trattazione si considereranno equivalenti i termini "sessione," "sessione d'inoltro" e "sessione di trasmissione".

Figura 5.3 : l'hop 0 è la sorgente; i nodi completati da essa entrano a far parte del *hop* successivo e nel *tier* 1. Si procede quindi ad instaurare le sessioni degli *hop* appartenenti a tale *tier*: i nuovi nodi completi entreranno negli *hop* successivi e quindi nel *tier* 2; essi potranno iniziare a loro volta eventuali sessioni solo al termine di tutte le sessioni degli *hop* del *tier* 1.

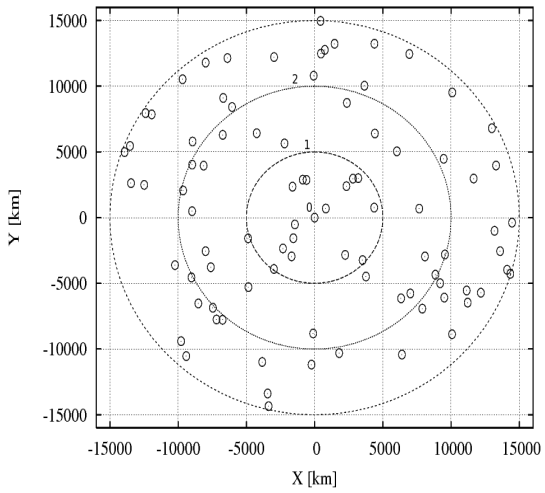

Figura 5.3: Topologia per $\lambda = 4$, $R_{rete} = 15000\ m$, $d_{tx} = 5000\ m$; i cerchi concentrici numerati indicano la successiva diffusione dell'informazione originale.

5.3.2 La creazione delle *policy*

Dato un nodo trasmettitore t, si considerino le distanze[4] dei nodi appartenenti all'insieme $\mathcal{R}_t$: a ciascuna di esse è possibile associare un *SNR* differente secondo la formula (5.2) e di conseguenza una differente probabilità p di errore sul pacchetto. In base a questa osservazione, è possibile partizionare il segmento $[0, d_{tx}]$ in m *classi d'errore* ordinate ed associare ciascun nodo ricevitore ad una classe $\mathcal{C}_m(i)$.

Le *Policy* a ridondanza costante

La politica a ridondanza costante è molto semplice. Il trasmettitore, una volta ricevute tutte le statistiche come descritto nella sottosezione 5.3.1, invia un numero costante di nuovi pacchetti codificati indipendentemente dalle richieste dei suoi nodi ricevitori.

[4]come detto in precedenza stimare la distanza è possibile tramite la stima del tempo di propagazione

Le *Policy* ottime a singola classe

Sia t il nodo trasmettitore e si supponga che l'insieme $\mathcal{R}_t$ di dimensioni R appartenga interamente alla *classe d'errore* $C_m(i)$. L'obiettivo di t è il completamento effciente di tutti gli R nodi in un numero di round massimo pari a L [8].
Durante il *round* i-simo, t trasmette un numero di pacchetti codificati x_i tale che $x_i^{\min} \leq x_i \leq x_i^{\max}$, dove $x_1^{\min} = K$, $x_i^{\min} = 1$, for $i > 1$, e gli $x_i^{\max}$ sono dei limiti imposti in fase di progetto. Il numero di pacchetti codificati spediti agli R vicini al round i-imo incluso è $X_i = \sum_{j=1}^{i} x_j$. Si definisce inoltre $\xi_i(r)$ il numero di pacchetti correttamente ricevuti, tra gli X_i pacchetti, dallo r-simo vicino. La *policy* ottima, obiettivo della seguente analisi, mappa X_{i-1} e $m_{i-1} = \min\{\xi_{i-1}(1), \xi_{i-1}(2), \ldots, \xi_{i-1}(R)\}$ nel numero aggiuntivo di nuovi pacchetti codificati, x_i, che saranno spediti nell'i-simo round nel caso in cui almeno un vicino, al termine del round $i-1$-simo, non è ancora in grado di decodificare i K pacchetti originali (tale evento è chiamato *fallimento* al round $i-1$-simo). Formalmente, $x_i = \mu_i(m_{i-1}, X_{i-1})$. Lo scopo dell'ottimizzazione è quello di trovare una *policy* $\mu_i(\cdot)$, per ciascuno round i, che minimizza il numero totale di pacchetti che dovranno essere spediti per consentire a tutti i vicini la corretta decodifica dei K pacchetti originali. Il costo associato al processo di trasmissione è funzione del numero di round e di x_i, $C(i, x_i)$. Si considera in questa sede $C(i, x_i) = x_i$, cosa che rispecchia il nostro obiettivo di minimizzazione. Ciascun i-simo round è un'epoca di decisione dove, nell'eventualità di *fallimento* al round $i-1$-simo, il trasmittitore deve scegliere un numero di pacchetti addizioni per il round i-simo corrente. Tale decisione è presa in modo tale da minimizzare tutti i costi possibili. Per portare a termine tale minimizzazione, si definisce $J(i, m_{i-1}, X_{i-1})$ come il minimo costo da spendere (*cost-to-go*) all'inizio del round i-simo, ovvero, il costo cumulativo dal round i fino alla decodifica con successo di tutti i vicini, dato che lo stato del sistema *HARQ all'inizio* del round i-simo è rappresentato dalla coppia $\mathcal{S}_i = (m_{i-1}, X_{i-1})$. L'espressione esatta di $J(\cdot)$ è l'obiettivo delle considerazioni seguenti. Si introducono ora delle incognite. Si assegna la stessa probabilità di cancellazione sul pacchetto p a ciascun collegamento tra il trasmettitore ed i suoi ricevitori $r = 1, 2, \ldots, R$. Per semplificare la notazione, si definisce $\mathcal{B}(x, y, p) = \binom{x}{y} p^y (1-p)^{x-y}$. La probabilità congiunta che un dato utente decodifichi correttamente i K pacchetti originali al termine del round $i-1$-simo e che

$\xi_{i-1} = x$, dato X_{i-1}, è calcolabile come:

$$\mathcal{P}_s(i-1, x) = \mathcal{B}(X_{i-1}, X_{i-1} - x, p)\Psi(x)\,. \tag{5.10}$$

Allo stesso modo, la probabilità congiunta che un generico utente r non sia ancora in grado di decodificare (decodifica senza successo, u) alla fine dell'$i-1$-simo round e che $\xi_{i-1} = x$, chiamata $\mathcal{P}_u(i-1, x)$, è ottenuta sostituendo $\Psi(x)$ in (5.10) con $1 - \Psi(x)$. Al round i-simo, per un determinato $\mathcal{S}_i = (m_{i-1}, X_{i-1})$ e per un determinato utente r si può calcolare la probabilità che $\xi_{i-1}(r) = x$, condizionato dal valore m_{i-1} e dall'evento *fallimento*, f:

$$P(\xi_{i-1}(r) = x | m_{i-1}, f) = \frac{P(\xi_{i-1}(r) = x, m_{i-1}, f)}{P(m_{i-1}, f)}\,. \tag{5.11}$$

A causa delle simmetrie in gioco, l'equazione sopra stante e quelle successive non dipendono dall'utente specifico r. Di conseguenza, per chiarezza, si ometterà tale dipendenza nel prosieguo della trattazione. Per calcolare $P(m_{i-1}, f)$, si definisce in primo luogo la probabilità congiunta di avere una decodifica *con successo* (s) or *senza successo* (u) e che $\xi_{i-1} \geq y$ pacchetti sono stati correttamente ricevuti da uno utente determinato alla fine del round $i-1$-simo, dato X_{i-1}, come: (dove si tenuto conto di tutti i possibili *pattern* di errore $0, \ldots, X_{i-1} - y$ nei round $1, 2, \ldots, i-1$):

$$F^{\geq}_{i-1,\chi}(y) = \sum_{e=0}^{X_{i-1}-y} \mathcal{P}_\chi(i-1, X_{i-1} - e)\,, \quad \chi \in \{s, u\}\,, \tag{5.12}$$

L'equazione (5.12) è utilizzata per trovare la coda della distribuzione per il minimo m_{i-1} (per brevità indicato con y nell'equazione succesiva) nel modo seguente:

$$\begin{aligned} F^{\geq}_{i-1}(y, R) &= \sum_{r=1}^{R} \binom{R}{r} F^{\geq}_{i-1,u}(y)^r F^{\geq}_{i-1,s}(y)^{R-r} = \\ &= (F^{\geq}_{i-1,u}(y) + F^{\geq}_{i-1,s}(y))^R - F^{\geq}_{i-1,s}(y)^R \end{aligned} \tag{5.13}$$

Si noti come $F^{\geq}_{i-1,u}(y) + F^{\geq}_{i-1,s}(y) = \sum_{e=0}^{X_{i-1}-y} \mathcal{B}(X_{i-1}, e, p)$ non dipenda da $\Psi(\cdot)$. La distribuzione congiunta di m_{i-1} e l'evento *fallimento* (f) alla fine del round $(i-1)$-esimo (il denominatore in (5.11) è infine ottenuta come:

$$P(m_{i-1}, f) = F^{\geq}_{i-1}(m_{i-1}, R) - F^{\geq}_{i-1}(m_{i-1} + 1, R)\,. \tag{5.14}$$

Si procede al calcolo del numeratore di (5.11). A questo scopo, si calcola in primo luogo la probabilità $P(\xi_{i-1} = x, m_{i-1} \geq y, f)$:

$$P(\xi_{i-1} = x, m_{i-1} \geq y, f) = \begin{cases} f(x,y) & 0 \leq y \leq x \text{ and} \\ & 0 \leq x \leq X_{i-1} \\ 0 & \text{otherwise}\,, \end{cases} \tag{5.15}$$

con:

$$\begin{aligned} f(x,y) &= \mathcal{P}_s(i-1,x) F^{\geq}_{i-1}(y, R-1) + \\ &+ \mathcal{P}_u(i-1,x) \Bigg(\sum_{e=0}^{X_{i-1}-y} \mathcal{B}(X_{i-1}, e, p) \Bigg)^{R-1} \end{aligned} \tag{5.16}$$

dove i due termini nella somma precedente tengono conto del completo ($\mathcal{P}_s(\cdot)$) o incompleto ($\mathcal{P}_u(\cdot)$) recupero dell'utente specificato fino al round $(i-1)$-esimo incluso. Ora, $P(\xi_{i-1} = x, m_{i-1} = y, f)$ può essere ricavata da (5.15) come:

$$\begin{aligned} P(\xi_{i-1}{=}x, m_{i-1}{=}y, f) &= P(\xi_{i-1}{=}x, m_{i-1}{\geq}y, f) - \\ &- P(\xi_{i-1}{=}x, m_{i-1}{\geq}y{+}1, f) \end{aligned} \tag{5.17}$$

È possibile ora calcorare (5.11), che è conseguentemente utilizzata per definire altre due equazioni, $G^{\geq}_{i,u}(x)$ e $G^{\geq}_{i,s}(x)$, rappresentanti la probabilità congiunta di avere una decodifica *senza successo* (u) o *con successo* (s) per un determinato utente alla fine del round $i > 1$, e $\xi_i \geq x$, dato m_{i-1}:[5]

$$\begin{aligned} G^{\geq}_{i,\chi}(x) &= \sum_{z=m_{i-1}}^{X_{i-1}} P(\xi_{i-1} = z | m_{i-1}, f) \Bigg[\sum_{e=0}^{x_i} \mathcal{B}(x_i, e, p) \times \\ &\times g_\chi(z + x_i - e)\, \mathbb{1}\{z + x_i - e \geq x\} \Bigg] \end{aligned} \tag{5.18}$$

dove il numero totale di pacchetti ricevuti fino all'$i-1$-simo round, X_{i-1}, è limitata da $\sum_{j=1}^{i-1} x_j^{\min} \leq X_{i-1} \leq \sum_{j=1}^{i-1} x_j^{\max}$, il minimo numero di pacchetti correttamente ricevuti, m_{i-1}, è tale che $0 \leq m_{i-1} \leq X_{i-1}$, ed il numero di pacchetti x_i spediti nel round i-simo deve soddisfare $x_i^{\min} \leq x_i \leq x_i^{\max}$. Quindi, i limiti per x sono

[5]Per un'analisi esatta, si dovrebbe tenere conto di tutti i pacchetti correttamente ricevuti da ciascun utente fino al round i-simo. Tuttavia, il calcolo della *policy* ottima in queste ipotesi è computazionamente improponibile. Per ovviare a ciò, si descrive l'evoluzione del processo tenendo traccia solamente del minimo m_{i-1}. Si è riscontrato in [8] infatti che i risultati ottenuto sono simili a quelli dell'analisi esatta.

$m_{i-1} \leq x \leq X_{i-1} + x_i$. Inoltre, $\chi \in \{s, u\}$, $g_s(x) = \Psi(x)$ e $g_u(x) = 1 - \Psi(x)$, $\mathbb{1}\{\cdot\}$ è la funzione indicatrice, che da uno one quando l'espressione tra parentesi è vera e zero altrimenti. La si è utilizzata in questa sede per tenere conto dei casi dove $\xi_i \geq x$. Gli indici z e e tengono nota rispettivamente del numero totale di pacchetti correttamente ricevuti nei round $1, 2, ..., i-1$ e del numero di pacchetti errati tra gli x_i spediti nel round i-simo. Si calcola infine la distribuzione congiunta di m_i e dell'evento *fallimento* alla fine del round i-simo, dato m_{i-1} (si assume $m_0 = 0$) e che si è avuto un evento di *fallimento* al round $i-1$-simo:

$$\Omega(m_i|m_{i-1}) = \begin{cases} P'(m_1, f) & i=1 \\ G_i^{\geq}(m_i, R) - G_i^{\geq}(m_i + 1, R) & i>1\,, \end{cases} \tag{5.19}$$

dove $G_i^{\geq}(m_i, R) = (G_{i,u}^{\geq}(m_i) + G_{i,s}^{\geq}(m_i))^R - G_{i,s}^{\geq}(m_i)^R$ (si riguardi il calcolo di (5.13)) e $P'(m_1, f)$ è data (5.14), sostituendo X_{i-1} con x_1 nei calcoli. Si è quindi in grado di scrivere l'equazione di ottimalità (5.20) per il problema affrontato:

$$J(i, m_{i-1}, X_{i-1}) = \min_{\ell(i) \leq x_i \leq x_i^{\max}} \left\{ C(i, x_i) + \sum_{m_i = m_{i-1}}^{m_{i-1}+x_i} \Omega(m_i|m_{i-1}) J(i{+}1, m_i, X_{i-1}{+}x_i) \right\} \tag{5.20}$$

In (5.20) $i = 1, 2, \ldots, L$, $J(L+1, \cdot, \cdot) = T$ è il costo finale in cui si incorre fallendo la decodifica dei K pacchetti originali in L round. Si noti che tale costo dovrebbe essere grande al punto che all'esistenza di una soluzione con successo (decodifica completa per tutti i nodi) in meno di $L+1$ round essa sia scelta dal programma ottimizzatore. Come in precedenza, $\sum_{j=1}^{i-1} x_j^{\min} \leq X_{i-1} \leq \sum_{j=1}^{i-1} x_j^{\max}$, $m_{i-1} = 0, 1, \ldots, X_{i-1}$. Oltretutto, $\ell(i)$ è data da:

$$\ell(i) = \begin{cases} x_1^{\min} & i=1 \\ x_i^{\min} & i>1 \text{ e } m_{i-1} \geq K \\ \min(K - m_{i-1}, x_i^{\max}) & i>1 \text{ e } m_{i-1} < K\,. \end{cases} \tag{5.21}$$

infatti, nel round $i > 1$, se $m_{i-1} < K$ almeno $K - m_{i-1}$ pacchetti addizionali devono essere spediti in modo che tutti gli r ricevitori con $\xi_i(r) = m_{i-1}$ possano ricevere almeno K pacchetti (che è il minimo numero di pacchetti richiesti per la decodifica).

La strategia di trasmissione ottimale, ovvero, lo x_i da usare per ogni round i-simo e per ogni stato $\mathcal{S}_i = (m_{i-1}, X_{i-1})$, è utilizzato risolvendo a ritroso (5.20).

Le *Policy* ottime multi classe

Si considerino ora R ricevitori e C *classi d'errore*. La classe c ha $R_c > 0$ utenti, tutti con la stessa probabilità di errore sul pacchetto p_c e $\sum_c R_c = R$. Sia X_{i-1} il numero totale di pacchetti codificati trasmessi fino al round $i-1$-simo incluso e m_{i-1}^c sia il minmo tra i pacchetti correttamente ricevuti dagli utenti della classe c al termine del round $(i-1)$-esimo. Per ciascuna classe c, la *policy* ottima $\mu^c(\cdot)$ è trovata secondo l'analisi della sezione prededente.
Al round i, si sceglie il numero di nuovi pacchetti codificati da trasmettere come: $x_i = \max_{i=1,2,\ldots,C}\{\mu_i^c(m_{i-1}^c, X_{i-1})\}$. Tale politica, che è molto semplice e pratica da applicare, è quasi ottima. Il criterio che si cerca di perseguire con la regola data è quello di soddisfare *tutti* gli utenti. Infatti, si vuole trasmettere la quantità minima di ridondanza che è ottima per la classe peggiore.
Il calcolo esatto della *policy* ottima globale sarebbe una generalizzazzione dell'analisi nella Sottosezione 5.3.2. Tuttavia ciò porterebbe ad uno spazio degli stati molto esteso, tale da rendere troppo oneroso il calcolo stesso [8].

5.3.3 Parametri di progetto dei codici a fontana

Per le simulazioni si sono utilizzati i seguenti parametri:

- blocchi di $K = 32$ pacchetti codificati di dimensione $s_k = 1000$ *bit*;
- ridondanza massima per round $x_i^{\max} = 16$;
- numero massimo di round per sessione $L = 5$;
- dimensione dei pacchetti di controllo $s_{ctrl} = 200$ *bit*.

5.3.4 Statistiche Osservate

Si vogliono ora definire e discutere le statistiche osservate nelle simulazioni:

Affidabilità del *broadcast* $p_{broadcast}$ **.** Con tale metrica si vuole misurare la probabilità media di portare a termine correttamente l'*intera* trasmissione *broadcast*;

Numero medio di *hop* n_{hop} **.** Con tale metrica si vuole testare la bontà dell'algoritmo progettato: minore sarà tale metrica, maggiore l'efficienza.

Avanzamento medio d_{proj} . Si consideri un nodo completo x diverso dalla sorgente s che abbia appena terminato una sessione di trasmissione. Per esso si può identificare la semiretta che ha origine in s passante per x e verso di percorrenza da s a x. Si considerino i nodi completati da x e si calcolino le loro proiezioni q_i sulla semiretta considerata. Si considerino quindi i segmenti orientati aventi estremi in x e q_i: se il segno è positivo si è avuto per costruzione un *avanzamento* verso il bordo della topologia. d_{proj} rappresenta quindi la media di tale metrica effettuata su tutte le sessioni. Se $x = s$ si sceglie una direzione orientata casuale e si aplicano ad essa i ragionamenti introdotti.

Raggio di completezza r_{compl} . Si consideri lo stesso nodo x del caso precedente; si definisce *raggio di completezza* il minimo raggio $r_{compl} < d_{tx}$ tale per cui all'interno dell'area del cerchio da esso generato siano presenti *esclusivamente* dei nodi completi. r_{compl} rappresenta la media su tutte le sessioni instaurate per una topologia.

Numero medio di pacchetti codificati spediti per sessione, n_{pkt}^{tx} . Si tratta del numero totale di pacchetti codificati spediti durante una sessione, mediato su tutte le sessioni. Tale metrica è essenziale per misurare l'efficienza energetica del protocollo: minore il valore, maggiore l'efficienza.

Numero medio di round per sessione n_{round} . Anch'esso è importante per la misura dell'efficienza energetica.

Probabilità media di fallimento di sessione p_{fail} . Con questa metrica si vuole misurare la probabilità che una sessione fallisca, ovvero che un nodo trasmettitore non riesca a completare i nodi riceventi entro il numero massimo di round L. La probabilità cercata è un'indicatore dell'efficienza della *policy* ottima.

Numero delle sessioni totali $N_{session}$. Esso rappresenta il numero totale di sessioni svolte nell'intera simulazione.

Funzione distribuzione di probabilità della copertura della rete. Con tale funzione si vuole misurare l'andamento della copertura di rete, all'aumentare delle sessioni.

5.3.5 Obiettivi

Gli obiettivi dell'analisi del protocollo progettato sono molteplici:

- valutare l'efficienza e l'affidabilità dello schema illustrato;
- valutare le prestazioni delle *policy* a ridondanza costante e quelle ottime in ambiente *multi hop*;
- confrontare le metriche r_{compl} , d_{proj} , n^{tx}_{pkt} , n_{round} , p_{fail} , della sorgente e quelle medie. In tal modo si vuole confrontare una sessione *broadcast* ad *hop* singolo ed una sessione *broadcast* ad *hop* multipli. La speranza è quella di trovare una forte correlazione tra i due insiemi: in questo in caso infatti sarebbe possibile predire l'andamento di un *broadcast multi hop* dai risultati del *broadcast* in ambiente ad *hop* singolo opportunamente scalati.

5.4 Il protocollo CRBCast

Come termine di paragone per il protocollo progettato si è scelto il protocollo *CRBCast*, che implementa utilizza anch'esso i codici a fontana, come spiegato nella sezione 4.4.1. Sono state inoltre apportate alcune modifiche alle due fasi allo scopo di migliorarne l'efficienza.

5.4.1 L'ottimizzazione della probabilità di inoltro

Il primo passo per l'implementazione del protocollo è stato l'ottimizzazione della probabilità d'inoltro p_{fwd}. Si ricorda come la Fase I attui un *broadcast* probabilistico dei pacchetti correttamente ricevuti e codificati dalla sorgente: è necessario quindi trovare la p_{fwd} ottima che consenta di massimizzare la diffusione del *broadcast* e di minimizzare allo stesso tempo le trasmissioni ridondanti. Poichè ogni nodo inoltra i pacchetti ricevuti con probabilità p_{fwd}, la probabilità che esso ritrasmetta informazione ridondante aumenta al crescere della densità della rete. Per riuscire nello scopo si sono misurate due quantità:

- la percentuale di nodi incompleti al termine della *Fase I*, metrica *decrescente* al crescere di p_{fwd};
- il numero totale di pacchetti codificati trasmessi da tutti i nodi che hanno partecipato alla *Fase I*, metrica *crescente* al crescere di p_{fwd}.

Per ottenere il valore ottimo in funzione di p_{fwd} , sono state moltiplicate le due metriche. Utilizzando i valori di λ definiti in Figura 5.2 sono state ottenute le curve di Figura 5.5 . I valori ottimi ottenuti ed utilizzati nelle simulazioni sono:

λ	p_{fwd}^{opt}
2.5	0.75
5	0.73
10	0.44
20	0.22

Figura 5.4: Valori ottimi di p_{fwd} al variare di λ.

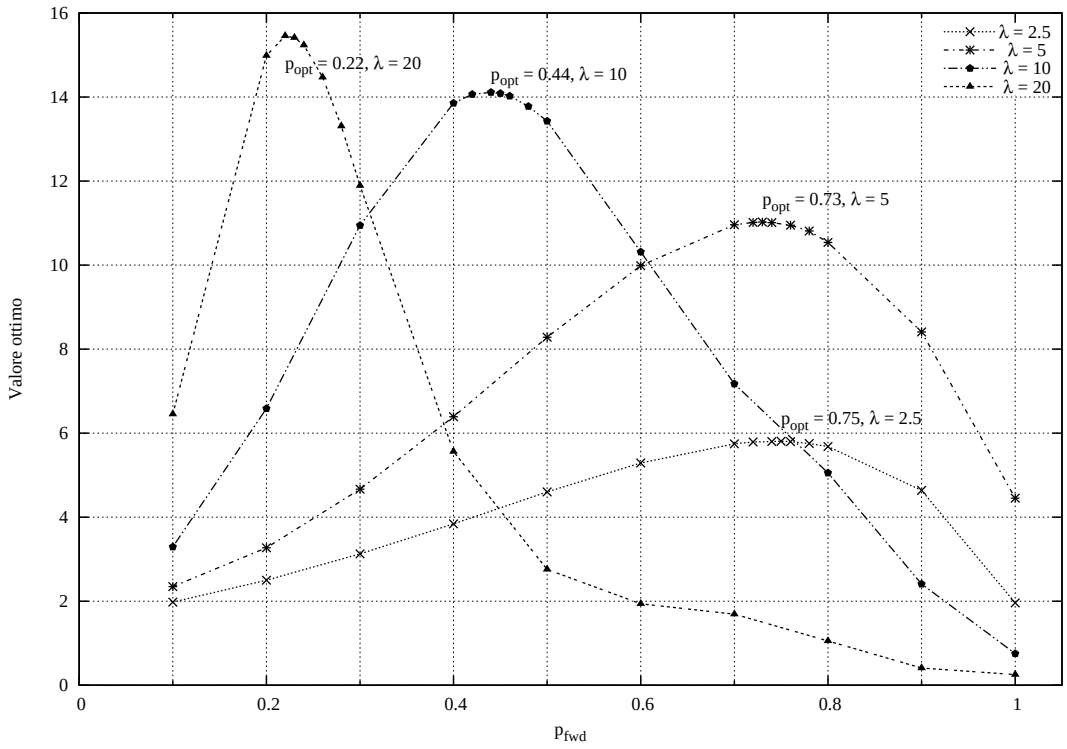

Figura 5.5: Ottimizzazione della p_{fwd} per la Fase I del protocollo *CRBCast*.

Dalla Figura 5.5 si nota che in topologie a bassa densità, come nel caso di $\lambda = 2.5$, 5 è necessario utilizzare una p_{fwd} elevata per consentire una buona diffusione del *broadcast*. Al contrario, nelle reti ad alta densità è sufficiente che una piccola parte dei nodi operi la ritrasmissione, è sufficiente cioè una p_{fwd} contenuta (caso $\lambda = 10$, 20).

5.4.2 La Fase I modificata

La Fase I del protocollo originale non sfrutta a dovere la diversità fornita dai codici a fontana. In particolare, il *broadcast* degli *stessi e soli* pacchetti codificati dalla sorgente è una forte limitazione: si consideri il caso in cui tali pacchetti codificati non consentano la corretta decodifica a causa della dipendenza lineare di alcuni di essi. Anche se fossero ricevuti correttamente dall'intera rete, la Fase II partirebbe nella situazione in cui la sola sorgente è l'unico nodo completo. Appare evidente come sia necessario introdurre dei *nuovi* pacchetti codificati anche nella Fase I per ottimizzarne l'efficacia.
A questo scopo si propongono le seguenti modifiche:

- Sono abilitati all'inoltro probabilistico i soli nodi completi. Tali nodi provvedono alla decodifica ed alla ricodifica di *nuovi* pacchetti. Come nel protocollo originale un nodo può procedere all'inoltro probabilistico una ed una sola volta.

- Se un nodo incompleto diviene completo in un qualsiasi istante temporale della Fase I esso ha diritto a partecipare alla Fase I modificata.

5.4.3 La Fase II modificata

La *policy* utilizzata dalla Fase II di *CRBCast* non sembra essere efficiente nè dal punto di vista del numero di pacchetti codificati spediti nè dal punto di vista energetico. La regola aurea della Fase II è il numero $\{n_p\gamma - N_{ack}^{min}\}$[6], tuttavia tale regola non è efficace in quanto non tiene conto nè della *classe d'errore* a cui il ricevitore i-esimo appartiene, nè dei tentativi di recupero precedenti.
Per ovviare a queste mancanze si propone la Fase II modificata, in cui si utilizzano le *policy* ottime trattate nelle sezioni precedenti.

5.4.4 Metriche osservate

Sono state considerate e raccolte le statistiche:

Numero di sessioni totale $N_{sessions}$ **.** Rappresenta il numero totale di sessioni svolte durante l'intera simulazione.

[6]dove $n_p\gamma$ è il numero di pacchetti codificati per blocco di informazione e N_{ack}^{min} è il minimo minimo tra il numero di pacchetti correttamente ricevuti dai suoi ricevitori.

Numero totale di sessioni della Fase I $N_{sessions}^{PhI}$ **.** Indica il numero di sessioni compiute durante la Fase I del protocollo. Non si considerano le sessioni sprecate.

Numero totale di sessioni sprecate della fase I N_{wst}^{PhI} **.** Indica il numero di sessioni della fase I inutili, ovvero nei casi in cui:

- non vi siano ricevitori nell'area di copertura;
- tutti i ricevitori all'interno dell'area di copertura siano già completi.

Numero totale di sessioni della Fase II $N_{sessions}^{PhII}$ **.** Rappresenta il numero totale di sessioni instaurate nella Fase II del protocollo.

Numero totale di messaggi sprecati $\boldsymbol{ADV}$ N_{wst}^{ADV} **.** Indica il numero di messaggi di ADV trasmessi inutilmente, ovvero nei casi in cui:

- non vi siano ricevitori nell'area di copertura;
- tutti i ricevitori all'interno dell'area di copertura siano già completi.

Numero medio di pacchetti codificati trasmessi nella Fase I $n_{pkt}^{tx}(PhI)$ **.** Indica il numero medio di pacchetti codificati trasmessi nella Fase I. Esso non comprende i pacchetti trasmessi nelle sessioni sprecate.

Numero medio di pacchetti codificati trasmessi nella Fase II $n_{pkt}^{tx}(PhII)$ **.** Indicatore dell'efficienza energetica della Fase II utilizzata.

Densità di probabilità della copertura della rete nella Fase I. Utile per stimare l'efficacia della Fase I utilizzata. Un maggiore valore di tale metrica indica un minore dispendio di risorse nella Fase II.

Densità di probabilità del numero di round di trasmissione in Fase II. Indicatore della bontà della *policy* utilizzata in Fase II. Un minore valore di tale metrica indica un minore dispendio di round di trasmissione per il completamento dell'intera rete.

Funzione distribuzione di probabilità della copertura di rete in Fase II Utile nel confronto del protocollo *CRBCast* con quello proposto in questa sede.

5.4.5 Obiettivi

Nel seguito sono state analizzate e confrontate sia tra loro sia con il protocollo *OFBP* le prestazioni delle seguenti varianti:

Id	Fase I	Fase II
1	CRBCast	CRBCast
2	CRBCast	modificata
3	modificata	CRBCast
4	modificata	modificata

Figura 5.6: Possibili combinazioni del protocollo *CRBCast*

Capitolo 6

La simulazione

6.1 L'ambiente simulativo *NS2*

NS è un simulatore orientato agli oggetti, scritto in C++, con un interprete OTcl come interfaccia utente; il simulatore presenta una gerarchia di Classi in C++ ed un'altra nell'interprete OTcl: esse sono strettamente correlate; dalla prospettiva utente infatti c'è una corrispondenza uno-a-uno tra una classe appartenente alla seconda ed una compilata nella prima. La radice di tale gerarchia è la classe TclObject. Gli utenti sono in grado di creare nuove simulazioni e quindi nuovi oggetti tramite l'interprete e sono abbinati con i rispettivi oggetti compilati in C++.
NS usa due linguaggi perchè il simulatore ha due obiettivi principali. In primo luogo simulazioni dettagliate di protocolli richiedono un linguaggio di progammazione che possa manipolare efficientemente *byte*, *packet header*, ed implementare algoritmi che operano su grandi moli di dati. Per questi compiti la velocità di esecuzione è essenziale. D'altra parte nella simulazioni è spesso necessario esplorare lo spazio dei parametri in maniera rapida ed efficiente. NS soddisfa entrambe le esigenze con due linguaggi, C++ e OTcl. C++ è rapido nell'esecuzione ma è più lento nella ricompilazione ed è per questo adatto alla programmazione degli elementi simulativi, mentre OTcl è lento nell'esecuzione ma è semplice e rapido per l'interazione con l'utente e la configurazione della simulazione.

6.1.1 Schema di funzionamento

NS si basa su un simulatore ad eventi. Un evento, in genere, è formato da un tempo d'esecuzione ed una funzione incaricata di gestire l'evento stesso. Lo *scheduler* ha il compito di selezionare l'evento più recente, eseguirlo fino al completamento, e iterare il ciclo eseguendo l'evento successivo. L'unità di tempo utilizzata è il secondo. Allo stato attuale lo schedulatore è *single-threaded*: uno ed uno solo evento può essere eseguito ad un determinato istante. Se più eventi hanno lo stesso tempo d'esecuzione verranno eseguiti nell'ordine di arrivo.
Su questo semplice ma efficace sistema è possibile costruire un'infinità di classi di oggetti aventi i compiti più svariati ed in grado di coprire qualsiasi esigenza simulativa: dai collegamenti wireless, a quelli cablati, dalle modulazioni più esotiche ai protocolli *IP*, *TCP*, ai protocolli di routing a qualsiasi protocollo o sistema appartenente allo *stack ISO-OSI*.
Gli oggetti fondamentali del simulatore sono i *nodi*. Grazie a degli oggetti *classifier* è possibile assegnare ed identificare i vari oggetti che definiscono il funzionamento e lo scopo del nodo stesso. Tra questi è di fondamentale importanza l'oggetto *agente*, che ha il compito di trattare i *pacchetti*, gli elementi di scambio informazioni tra i vari agenti collocati sui nodi di tutta la rete. Agli oggetti *connettori* è lasciato il compito di collegare i vari nodi tra loro.

6.2 *NS2-MIRACLE*

MIRACLE[1] è una libreria per ns2 sviluppata e progettata per permettere la comunicazione tra più strati dello standard *ISO-OSI* ed una progettazione *multi-strato* flessibile ed efficiente in ns2. Inoltre, la libreria contiene un insieme di classi e metodi per la progettazione e comunicazione *intra-strato* ed *intra-protocollo*. Ad esempio tali primitive permettono lo scambio di qualsiasi tipo di messaggio / struttura / comando tra i moduli o protocolli.

[1] Multi-InteRfAce Cross-Layer Extension

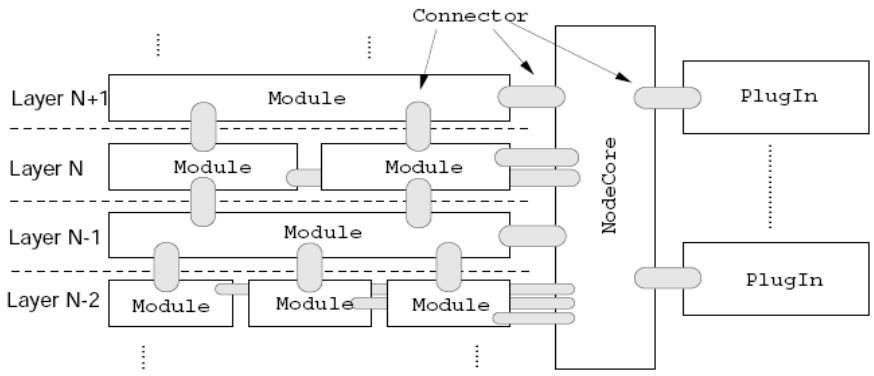

Figura 6.1: Architettura della libreria *Miracle*.

6.2.1 La classe *Module*

Uno dei pilastri fondamentali della struttura è rappresentato dalla classe *Module*. Module è una classe progettata per contenere qualsiasi protocollo o modulo si è soliti creare per simulare il corrispondente *stack ISO-OSI*. A tale scopo esso è provvisto di un canale di comunicazione tra moduli di strati adiacenti con lo stesso nome del riferimento *OSI*: il *Service Access Point* (*SAP*). Grazie alla flessibilità della liberia, è possibile definire un numero qualsiasi di moduli e di strati e connettere ciascuno di essi tramite *SAP*. Il *SAP* è utilizzato inoltre per il tracciamento delle attività simulative. Il canale è stato invece progettato come modulo *ad-hoc*, poichè esso deve essere in comune tra più moduli e deve poter memorizzare dati di tutti gli stessi. Tali funzionalità sono state create grazie alla classe *Channel SAP* (*ChSAP*). Per quanto concerne la comunicazione *cross-layer* si è introdotto un altro canale di comunicazione chiamato *Cross-Layer SAP* (*ClSAP*), che ha il compito di recapitare tali messaggi e le cui funzioni sono descritte in dettaglio nella sezione seguente.

6.2.2 Le classi *Cross-Layer Message* e *Node-Core*

Un ulteriore novità introdotta da *MIRACLE* è la comunicazione tra moduli tramite *Messaggi Cross Layer* definiti nella classe CLMessage. Tali mesaggi sono scambiati tramite il *Cross-Layer SAP* (*ClSAP*) e poichè tutti i moduli ne sono equipaggiati e sono interconnessi tramite essi la comunicazione *Cross-Layer* è presto fatta. La struttura che è incaricata di consegnare tali messaggi al giusto *ClSAP* è la classe *NodeCore*. Estendendo la classe *ClMessage*, è possibile ottene-

re qualsiasi messaggio *Cross-Layer* richiesto. Per migliorare la versatilità di tale struttura si sono introdotti due tipi di comunicazione: asincrona e sincrona. Nel primo caso non si richiede alcuna risposta diretta e, se richiesto, essa può essere inviata tramite *ClMessage* in un secondo momento. In modalità sincrona invece è obbligatorio che la risposta avvenga nella stessa istanza del messaggio ricevuto. Tali metodi danno al programmatore la possibilità di avere variabili in comune tra più moduli.

6.2.3 La classe *Plugin*

La classe *PlugIn* è un'altra novità introdotta da *MIRACLE* per dare flessibilità alle strutture dati e per modellare nuovi tipi di architetture. *PlugIn* infatti è la classe padre di *Module* ed è progettata per dare la possibilità di collegare qualsiasi modo direttamente agli oggetti *NodeCore*. In *PlugIn* sono definiti solo le comunicazioni e le funzionalità *cross-layer* di *Module* permettendo in questo modo di raggruppare tutte le caratteristiche di difficile progettazione e le eventuali caratteristiche di intelligenza *extra-modulo* utili alle funzionalità *cross-layer*.

6.3 Il canale sottomarino di *MIRACLE*

Come descritto nella sezione precedente, il canale in *MIRACLE* è una classe che ha il compito di connettere le varie entità della rete, i *nodi*, e , nello specifico, di connettere gli *strati fisici*. L'unità d'informazione è il *pacchetto*: all'arrivo di uno di essi al canale è demandato l'inoltro dello stesso secondo le regole di propagazione specifiche.
Nel caso sottomarino, la propagazione è quasi-sferica, il pacchetto viene quindi inoltrato ad ogni strato fisico della rete con il ritardo di propagazione appropriato, secondo la trattazione vista nel Capitolo 2. Ogni modulo appartenente allo strato fisico riceverà il pacchetto e calcolerà l'*SNR* corrispondente secondo l'equazione (5.2), dove tutti gli elementi presenti in tale formula sono calcolati come descritto nel Capitolo 2 e *forniti dal canale stesso*. Se tale *SNR* è sufficiente alla corretta ricezione del pacchetto, esso è inoltrato agli strati superiori.

6.4 I moduli *MIRACLE* sviluppati

Lo sviluppo di applicazioni in *MIRACLE* è concettualmente intuitivo una volta appresa l'architettura di base. La classe *Module* è infatti il mattone essenziale di tutte le applicazioni sviluppate in questa sede.

6.4.1 Il modulo dei codici a fontana

La prima applicazione progettata è stata il modulo dei codici a fontana, la base cioè del protocollo qui presentato e del protocollo *CRBCast*: esso ha il compito di instaurare e controllare il corretto svolgimento di una sessione di trasmissione descritta nella Sezione 5.3.1 . Esso è posto sopra lo strato *MAC* nello *stack ISO-OSI* di riferimento.
Per il dialogo con i moduli esso si appoggia a due canali:

- il canale dati, dove vengono spediti e ricevuti i pacchetti codificati;
- il canale di controllo, dove vengono spediti e ricevuti i messaggi di controllo. La peculiarità di quest'ultimo canale è quello di essere privo di errori. In questo modo è possibile simulare l'assenza di errore sul *feedback*.

Il funzionamento del modulo è quello di una macchina a stati finiti: tramite l'utilizzo di *timer* essa è in grado di pianificare gli *eventi*[2] necessari al controllo della condizione per la permanenza nello stato corrente o il passaggio allo stato successivo. Gli stati possibili possono essere schematizzati nei seguenti:

1. **Instaurazione della sessione**. In questa fase il trasmettitore provvede ad instaurare la sessione per mezzo dei messaggi di controllo *SYNC* e dei messaggi di risposta SYNC ACK. Completata la lista dei nodi ricevitori esso provvede al calcolo delle distanze relative ed alla loro catalogazione in *classi d'errore*. Se tale procedura è andata a buon fine è possibile passare alla fase (2). In caso contrario si ripete la procedura.

2. **Trasmissione dei pacchetti codificati**. In questa fase il trasmettitore procede al calcolo del numero dei pacchetti da trasmettere. Tale calcolo si basa sulla *policy* utilizzata e sul round corrente:

[2]intesi nell'accezione di *ns*, come descritto nella Sezione 6.1 .

- *Policy a ridondanza costante, primo round*: il numero cercato $N^{(1)}_{pkts_{tx}} = K + \xi$, dove ξ è la ridondanza costante.
- *Policy a ridondanza costante, round* $i > 1$: sia i il round corrente con $i > 1$, allora $N^{(i)}_{pkts_{tx}} = N^{(i-1)}_{pkts_{tx}} + \xi$.
- *Policy ottima, round i-simo*: sia i il round corrente, allora $N^{(i)}_{pkts_{tx}} = N^{(i-1)}_{pkts_{tx}} + \eta(i)$ dove $\eta(i)$ il valore corrente della policy ottima calcolata in Sezione 5.3.2 .

Al termine della trasmissione dei pacchetti dati, è inviato un messaggio di controllo *DECODE* per informare i nodi riceventi del termine della trasmissione e per ordinare la decodifica e l'invio di un messaggio di controllo *STATS*, contente il rango residuo e il numero di pacchetti codificati correttamente ricevuti. Si fa notare come questo sia un espediente simulativo: in un implementazione più realistica sarebbe necessario programmare un opportuno meccanismo di segnalazione della fine del blocco codificato, che tuttavia non è argomento di tesi.

3. **Controllo dei messaggi *STATS***. Il nodo trasmettitore provvede alla ricezione dei pacchetti *STATS*, in caso di errata o mancata ricezione esso provvede ad una nuova richiesta specificando l'identificativo del nodo mancante, setta un *timer* lungo un *RTT*, allo scadere del quale si ritorna all'inizio del presente stato (3). Nel caso in cui il controllo degli *STATS* sia avvenuto con successo esso procede al calcolo del massimo rango residuo e del minimo numero di pacchetti codificati correttamente ricevuto nella stessa classe e successivamente prende rispettivamente il massimo ed il minimo tra questi ultimi.
   Se il massimo rango residuo è zero, tutti i nodi sono completi e si può procedere alla trasmissione di un nuovo blocco d'informazione codificata (in caso ci sia) e quindi a (2) dopo aver informato i nodi ricevitori tramite un messaggio di *NEW BLOCK*. Se invece tale rango è maggiore di zero ciò significa che qualche ricevitore non è stato in grado di decodificare correttamente. Se il round corrente i minore o uguale a L si può procedere al recupero dei nodi in questione ritornando a (3), tenendo conto dell'informazioni necessarie all'utilizzo della *policy* ottima (se prevista): massimo rango residuo, minimo numero di pacchetti correttamente ricevuti e classe

d'errore relativa.
Se $i > L$ si considera il blocco corrente fallito, si informano i ricevitori tramite un messaggio *NEW BLOCK* e si procede alla trasmissione di un blocco d'informazione successiva (se presente) e quindi a (2).

4. **Raccolta delle metriche**. Al termine della simulazione si raccolgono le metriche d'interesse.

6.4.2 Il modulo di controllo del protocollo *multihop*

Tale modulo ha il compito di simulare il protocollo in Sezione 5.3.1, e di coordinare le sessioni di inoltro. Insiemi fondamentali per il controllo sono l'insieme dei nodi appartenenti all'*hop* corrente $\mathcal{I}_{currhop}$, quello dei nodi appartenenti all'*hop* successivo $\mathcal{I}_{nexthop}$, l'insieme dei nodi completi $\mathcal{I}_{compl}$ e quello dei nodi incompleti $\mathcal{I}_{incompl}$. Il funzionamento del modulo può essere schematizzato in:

0. **Inizializzazione dell'algoritmo**. All'inizio del protocollo la sorgente s è l'unico nodo che ha l'informazione originale. Pertanto si pone s in $\mathcal{I}_{currhop}$ come passo iniziale.

1. **Calcolo degli insiemi di copertura**. Per ciascun nodo $t \in \mathcal{I}_{currhop}$ si calcolano le distanze relative $d(t,u)$, $u \in \mathcal{I}_{incompl}$. Per ogni u si calcola la $d(t,u)_{min} = \min_t \{d(t,u)\}$; se $d(t,u)_{min} < d_{tx}$ allora $u \in \mathcal{R}_t$ ovvero u entrerà a far parte dei ricevitori di t. Si passa così a (3).

2. **Avvio delle sessioni**. Successivamente per ciascun nodo $t \in \mathcal{I}_{currhop}$ che abbia $\mathcal{R}_t \neq \emptyset$ si avviano in sequenza le sessioni di trasmissioni descritte in 6.4.1. Al termine di ognuna di esse:

   (a) si raccolgono le metriche della sessione corrente;

   (b) si inseriscono gli eventuali nuovi nodi completi c in $\mathcal{I}_{nexthop}$;

   (c) si passa alla sessione successiva;

   La sequenzialità delle sessioni è essenziale per l'assenza di collisioni, ipotesi importante per l'idealità della simulazione stessa. Al termine delle sessioni di $\mathcal{I}_{currhop}$, si pone $\mathcal{I}_{nexthop} = \mathcal{I}_{currhop}$. Se l'informazione originale ha raggiunto tutta la rete si procede con l'invio di un nuovo blocco e si torna a (0). Altrimenti si torna ad (1).

3. **Raccolta delle metriche**. Al termine della simulazione si raccolgono le metriche d'interesse.

6.4.3 Il modulo di controllo *CRBCast*

Il modulo in questione ha il compito di simulare il protocollo *CRBCast* descritto in 4.4.1 e di coordinarne le sessioni d'inoltro. Anche in questo caso è necessario tenere conto di $\mathcal{I}_{currhop}$, $\mathcal{I}_{nexthop}$, $\mathcal{I}_{compl}$, $\mathcal{I}_{incompl}$.

La Fase I originale

La Fase I originale può essere schematizzata in:

0. **Inizializzazione dell'algoritmo**. All'inizio del protocollo la sorgente s è l'unico nodo che possiede l'informazione originale. Pertanto si pone s in $\mathcal{I}_{currhop}$ come passo iniziale e si procede all'avvio della sua sessione. Si pongono tutti i nodi ricevitori in $\mathcal{I}_{nexthop}$ e si pone al termine della sessione $\mathcal{I}_{nexthop} = \mathcal{I}_{currhop}$.

1. **Calcolo degli insiemi di copertura**. Per ciascun nodo $t \in \mathcal{I}_{currhop}$ che abbia ricevuto correttamente almeno un pacchetto di quelli codificati dalla sorgente si calcolano le distanze relative $d(t,u)$, $u \in \mathcal{I}_{incompl}$: se $d(t,u)_{min} < d_{tx}$ allora $u \in \mathcal{R}_t$ ovvero u entrerà a far parte dei ricevitori di t. Si fa notare come sia probabile l'appartenenza di qualche u a più insiemi $\mathcal{R}_t$. Si passa così a (3).

2. **Avvio delle sessioni**. Successivamente per ciascun nodo $t \in \mathcal{I}_{currhop}$ si testa la probabilità di inoltro p_{fwd} . Se la probabilità p ottenuta è $p \leq p_{fwd}$ e se $\mathcal{R}_t \neq \emptyset$ si *avvia la sessione d'inoltro dei pacchetti codificati dalla sorgente correttamente ricevuti* . Al termine di quest'ultima:

   (a) si raccolgono le metriche della sessione corrente;

   (b) si inseriscono i nodi ricevitori c che abbiano ricevuto correttamente almeno un pacchetto in $\mathcal{I}_{nexthop}$;

   (c) si passa alla sessione successiva;

   Se invece $p \leq p_{fwd}$ ma $\mathcal{R}_t = \emptyset$ si considera sessione sprecata. se $p > p_{fwd}$ non si procede alla fase d'inoltro e si considera il nodo come se avesse

già proceduto all'inoltro. La sequenzialità delle sessioni è essenziale per l'assenza di collisioni, ipotesi importante per l'idealità della simulazione stessa. Al termine delle sessioni di $\mathcal{I}_{currhop}$, si pone $\mathcal{I}_{nexthop} = \mathcal{I}_{currhop}$. Se tutti i nodi che abbiano ricevuto almeno un pacchetto codificato della sorgente hanno portato a termine la sessione d'inoltro si passa alla *Fase II*. Altrimenti si torna ad (1).

La Fase I modificata

La Fase I modificata differisce nel punto (2) della sottosezione precedente:

2. **Avvio delle sessioni**. Successivamente per ciascun nodo $t \in \mathcal{I}_{currhop}$ si testa la probabilità di inoltro p_{fwd} . Se la probabilità p ottenuta è $p \leq p_{fwd}$ e se $\mathcal{R}_t \neq \emptyset$ si avvia la sessione di trasmissione di nuovi pacchetti codificati come descritto in 6.4.1. Al termine di quest'ultima:

   (a) si raccolgono le metriche della sessione corrente;

   (b) si inseriscono i nodi ricevitori c completi in $\mathcal{I}_{nexthop}$;

   (c) si passa alla sessione successiva;

   Se invece $p \leq p_{fwd}$ ma $\mathcal{R}_t = \emptyset$ si considera sessione sprecata. Se $p > p_{fwd}$ non si procede alla fase d'inoltro e si considera il nodo come se avesse già proceduto all'inoltro. La sequenzialità delle sessioni è essenziale per l'assenza di collisioni, ipotesi importante per l'idealità della simulazione stessa. Al termine delle sessioni di $\mathcal{I}_{currhop}$, si pone $\mathcal{I}_{nexthop} = \mathcal{I}_{currhop}$. Se tutti i nodi completi della rete hanno provveduto alla trasmissione una ed una sola si passa alla *Fase II*. Altrimenti si torna ad (1).

La Fase II originale

Si consideri la lista ordinata dei nodi completi della rete $\mathcal{L}_{compl}$. La Fase II originale può essere a sua volta schematizzata nel modo seguente:

1. **Calcolo degli insiemi di copertura**. Si estrae (e quindi si elimina) il primo nodo completo c da $\mathcal{L}_{compl}$, per esso si calcolano le distanze relative di tutti i nodi incompleti $d(c, u)$, $\forall u \in \mathcal{I}_{incompl}$; se $d(c, u) < d_{tx}$ Si inserisce u in $\mathcal{R}_c$.

2. **Avvio della sessione di recupero**. Se $\mathcal{R}_c \neq \emptyset$ si avvia la sessione del nodo c corrente :

   (a) si raccoglie la statistica del numero massimo di pacchetti codificati correttamente ricevuti da ognuno dei ricevitori;

   (b) si calcola il minimo tra le statistiche precedenti N_{ack}^{min} e si inviano $n_p\gamma - N_{ack}^{min}$ nuovi pacchetti codificati;

   (c) si raccolgono le statistiche d'interesse.

   (d) si aggiungono in coda a $\mathcal{L}_{compl}$ gli eventuali nuovi nodi completi.

   Nel caso si abbia $\mathcal{R}_c = \emptyset$ si considera la sessione come sessione sprecata. Se esistono ancora dei nodi incompleti si torna a (1). Se al contrario tutti i nodi sono completi, si raccolgono le statistiche d'interesse e si procede alla Fase I di un nuovo blocco d'informazione (se presente).

La Fase II modificata

La fase II modificata differisce dalla Fase II originale nel punto (2):

2. **Avvio della sessione di recupero**. Se $\mathcal{R}_c \neq \emptyset$:

   (a) si avvia la sessione del nodo c corrente con la *policy* ottima come descritto in Sezione 6.4.1. Si fa notare come in questo caso la sessione possa durare fino a L round contro uno della Fase II originale.

   (b) Al termine della sessione si raccolgono le statistiche d'interesse;

   (c) si aggiungono in coda a $\mathcal{L}_{compl}$ gli eventuali nuovi nodi completi.

   Nel caso si abbia $\mathcal{R}_c = \emptyset$ si considera la sessione come sessione sprecata. Se esistono ancora dei nodi incompleti si torna a (1). Se al contrario tutti i nodi sono completi, si raccolgono le statistiche d'interesse e si procede alla Fase I di un nuovo blocco d'informazione (se presente).

Capitolo 7

Risultati

Si presentano ora i risultati ottenuti dalle simulazioni in ambiente *Ns2-MIRACLE*. In Sezione 7.1 sono illustrati i risultati del protocollo *OFBP*, in Sezione 7.2 invece sono presentati i risultati del protocollo *CRBCast*; il confronto finale è in Sezione 7.3. Ciascuna simulazione si basa sulla trasmissione *broadcast* di 1000 blocchi d'informazione. Per raggiungere un buon margine di confidenza statistica, ogni punto dei grafici seguenti equivale alla media di circa 400 simulazioni effettuate su topologie di rete differenti, ovvero su realizzazioni separate di un processo di Poisson d'area, di parametro λ fissato. Uno studio degli intervalli di confidenza ha permesso di rilevare l'importanza di una media effettuata su un numero di topologie maggiore, piuttosto che su un numero maggiore di blocchi di pacchetti codificati. Si è quindi deciso di aumentare il numero di topologie simulate fino a raggiungere ristretti intervalli di confidenza al 95%.

7.1 I codici a fontana in ambiente *multi hop*

Di seguito sono riportati i risultati delle simulazioni del protocollo *OFBP*. Si passerà successivamente al protocollo *CRBCast* e si concluderà con il confronto tra i due. Per ciascuna delle metriche osservate si prenderanno in esame i casi di $d_{tx} = 2500, 5000, 7500\ m$ con *policy* a ridondanza costante; si confronterà altresì il caso $d_{tx} = 5000m$ della categoria precedente con il caso a *policy* ottima.

7.1.1 L'efficienza del protocollo proposto

Si osservano in primo luogo le metriche essenziale a verificare la bontà del protocollo: *probabilità di fallimento del broadcast*, *efficienza di trasmissione*, *ritardo medio per sessione* e *ritardo medio per hop*.

La probabilità di fallire il *broadcast*

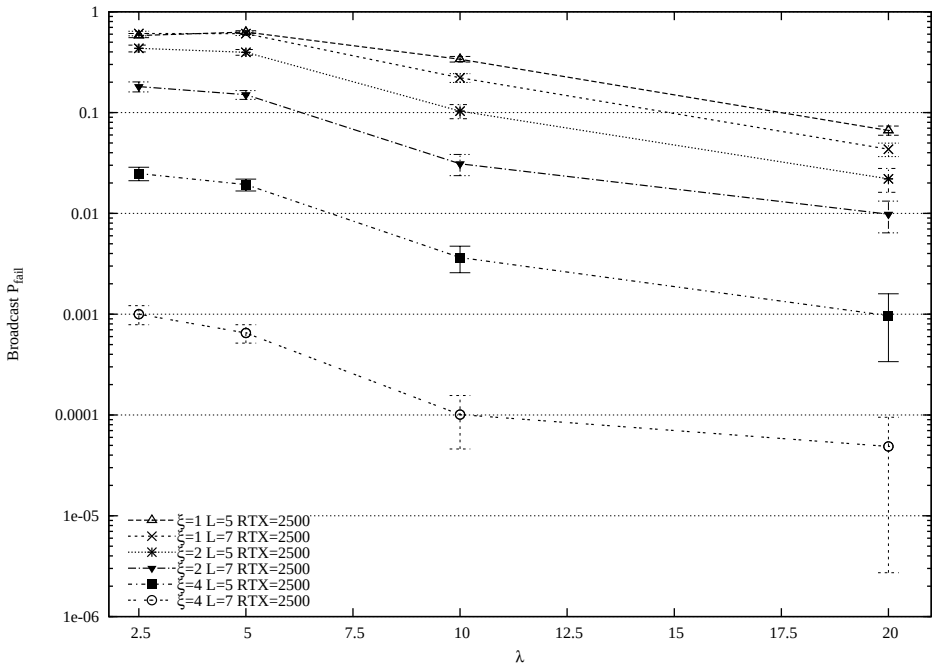

Figura 7.1: La probabilità di fallire il *broadcast* per $d_{tx} = 2500\ m$.

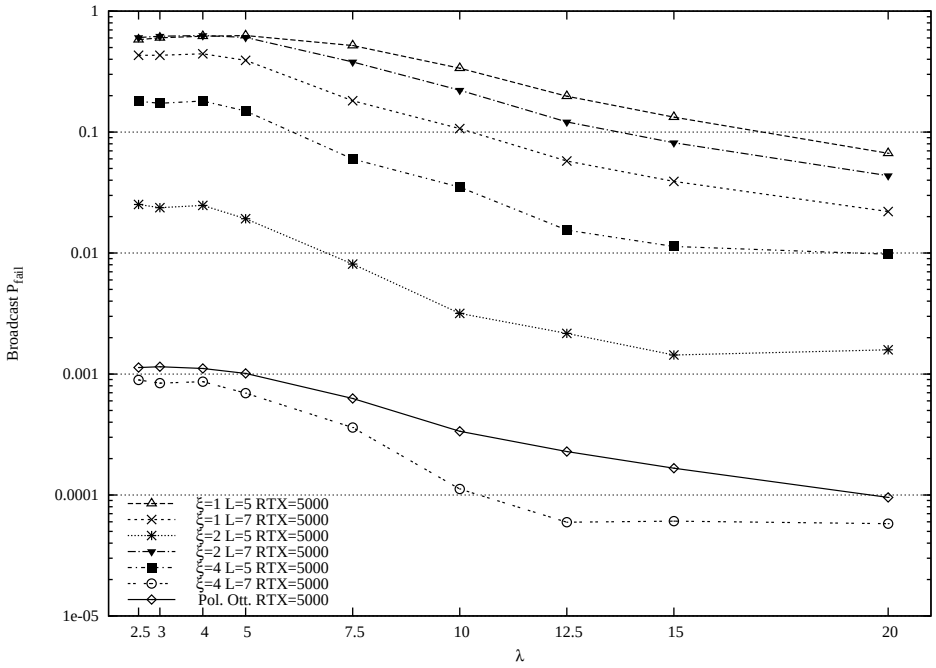

Figura 7.2: La probabilità di fallire il *broadcast* per $d_{tx} = 5000\ m$.

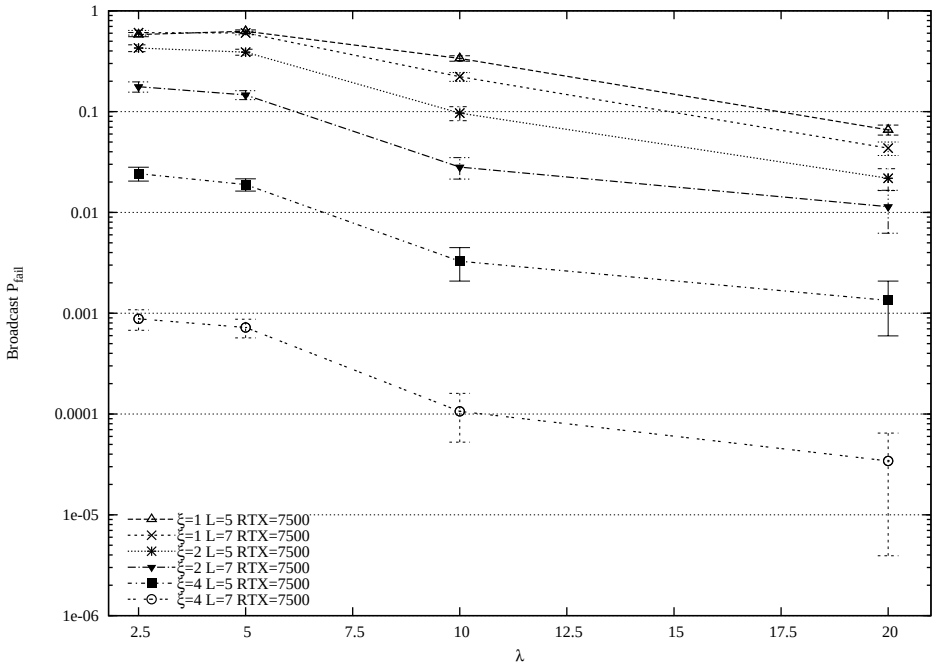

Figura 7.3: La probabilità di fallire il *broadcast* per $d_{tx} = 7500\ m$.

Si osserva innanzitutto che siamo di fronte ad un comportamento simile tra le tre figure proposte. Vi è soltanto un cambiamento di scala tra i risultati: tale comportamento è da imputarsi alla potenza in trasmissione P_{tx}, che è stata adattata a ciascuno delle d_{tx} per ottenere una p_{err} sul pacchetto di 0.25. **La similitudine dei risultati sarà un elemento costante di tutta l'analisi del protocollo proposto, pertanto nelle sezioni successive sono riportate le figure relative al caso $d_{tx} = 5000$, mentre le rimanenti sono catalogate in Appendice A**. Si consideri allora la Figura 7.2; si notano due fattori mitiganti della probabilità di fallimento:

- **l'aumento della ridondanza**. All'aumentare della ridondanza (ξ, L) trasmessa aumenta ovviamente la probabilità di ottenere una decodifica corretta;
- **l'aumento di λ**. All'aumentare della densità aumenta la probabilità che un nodo incompleto possa essere recuperato dai suoi vicini completi, numero che è proporzionale a λ.

Si osserva infine la $p_{broadcast}$ della *policy* ottima. Essa ha prestazioni migliori rispetto alle configurazioni di ξ per $L = 5$. Si noti infatti che la curva che ha prestazioni superiori alla *policy* ottima per $L = 5$ è in realtà ottenuta in condizioni diverse $(L = 7)$.

L'efficienza di trasmissione

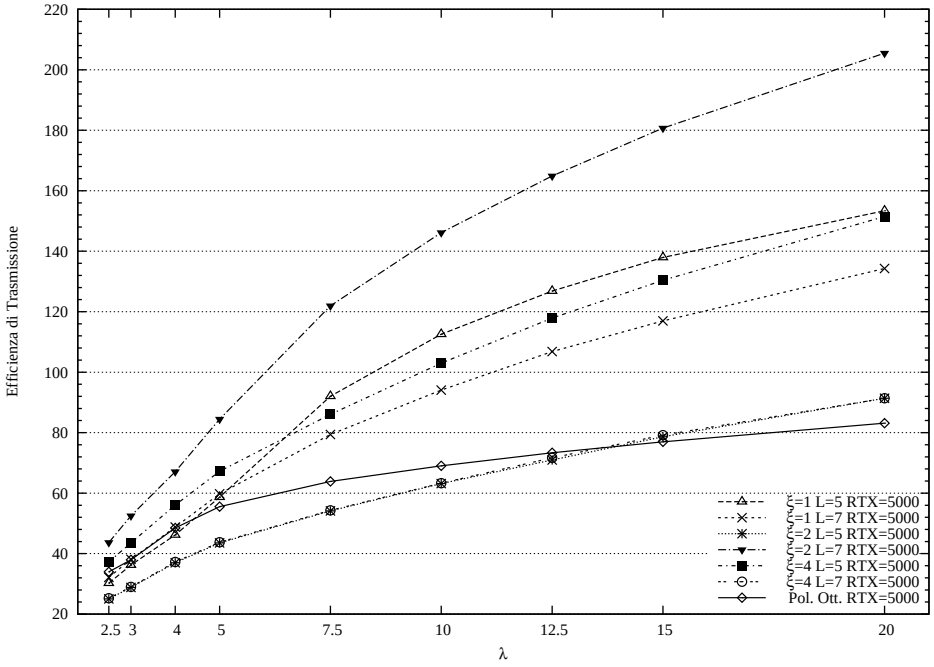

Figura 7.4: L'efficienza di trasmissione per $d_{tx} = 5000\ m$.

L'efficienza di trasmissione è il rapporto tra il numero di pacchetti codificati spediti per disseminare correttamente l'informazione in tutta la rete ed il numero di pacchetti d'informazione originali. Minore è tale rapporto, maggiore è l'efficienza. Si consideri la Figura 7.4: si nota subito come per $\xi = 1, 2$ il passaggio di L da 5 a 7 comporti un maggior dispendio di energia senza vantaggi significativi. La ridondanza infatti è troppo poca e tale aumento comporta una minima diminuzione della $p_{broadcast}$ (si veda Figura 7.2) contro un numero eccessivo di pacchetti codificati trasmessi.
Al contrario, per $\xi = 4$ si nota come le curve relative a $L = 5, 7$ siano sovrapposte: ciò sta ad indicare come la policy stabilita sia in grado di raggiungere e anche superare la trasmissione del numero di pacchetti codificati richiesto in media per ottenere una decodifica con successo.
La *policy* ottima non è legata ad un quantitativo fissato di ridondanza incrementale per round e dunque ottiene, come previsto, le prestazioni migliori.

Il ritardo medio di una sessione

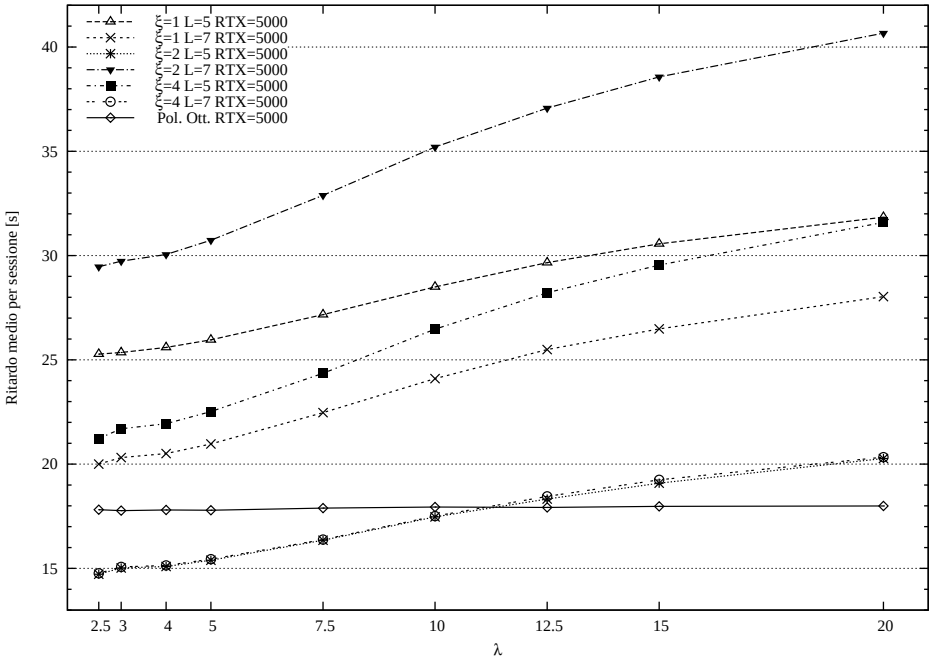

Figura 7.5: Il ritardo medio di una sessione per $d_{tx} = 5000$.

Si consideri la Figura 7.5. Si osserva in primo luogo che per $\xi = 1, 2$ il ritardo aumenta all'aumentare di λ ed è maggiore per L maggiore. Ciò è dovuto ancora una volta alle prestazioni limitate delle policy a ridondanza costante bassa: esse sono costrette molto spesso ad utilizzare tutti gli L round aggiungendo al ritardo di trasmissione ben L RTT[1]. In questi casi il numero di pacchetti trasmessi è con probabilità molto alta $K + (L+1)\xi$ e tale valore è inferiore al numero di pacchetti codificati trasmessi con $\xi = 4$; nonostante ciò il ritardo è ben più elevato e la probabilità di successo più bassa.
Si noti successivamente come il ritardo sia quasi lineare con λ: tale comportamento è tipico delle *policy* a ridondanza costante per round.
La *policy* ottima è indipendente da λ: ciò è forte segnale della bontà della stessa; essa è in grado, cioè, di completare i propri nodi ricevitori nello stesso tempo indipendentemente dal numero. Si osservi infine come essa impieghi un tempo maggiore del caso $\xi = 4$ per $\lambda = 2.5$, 5: l'efficienza in trasmissione è ottenuta quindi al prezzo di un maggior ritardo per terminare la sessione.

[1]Round Trip Time

Il numero medio di *hop*

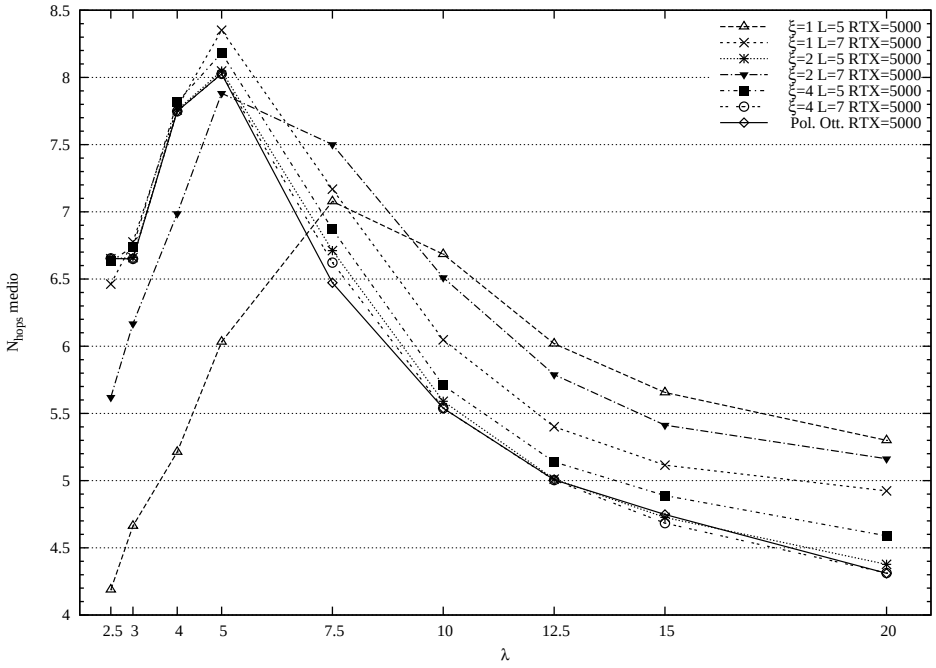

Figura 7.6: Il numero medio di *hop* per $d_{tx} = 5000$.

Si osserva in primo luogo l'andamento comune a tutte le curve eccetto quella per $\xi = 2$, $L = 5$: esse presentano un massimo in $\lambda = 5$. La spiegazione è da individuare nella densità della topologia stessa: per $\lambda = 2.5$ il numero di nodi è limitato, di conseguenza in numero di *hop* da coprire per completare l'intera rete è minore rispetto a $\lambda = 5$, dove la rete ha ancora una distribuzione abbastanza sparsa ma ha il doppio del numero dei nodi da raggiungere. Di conseguenza il numero di *hop* aumenta. Al crescere della densità ($\lambda > 5$) il numero di hop diminuisce quasi linearmente, poichè aumenta il grado di parallelismo tra le trasmissioni. Si consideri poi la curva per $\xi = 1$, $L = 5$: essa presenta dei valori minori delle altre per $\lambda = 2.5, 5$ per il semplice fatto che la probabilità di errore è tale da permettere il completamento di topologie a basso numero di hop. Per densità elevate la curva $\xi = 1$ si assesta in media come previsto su valori massimi. Si osservi infine come la curva appartenente alla policy ottima sia in grado di connettere con il minor numero di *hop* tutte le casistiche.

Il ritardo medio di un blocco codificato

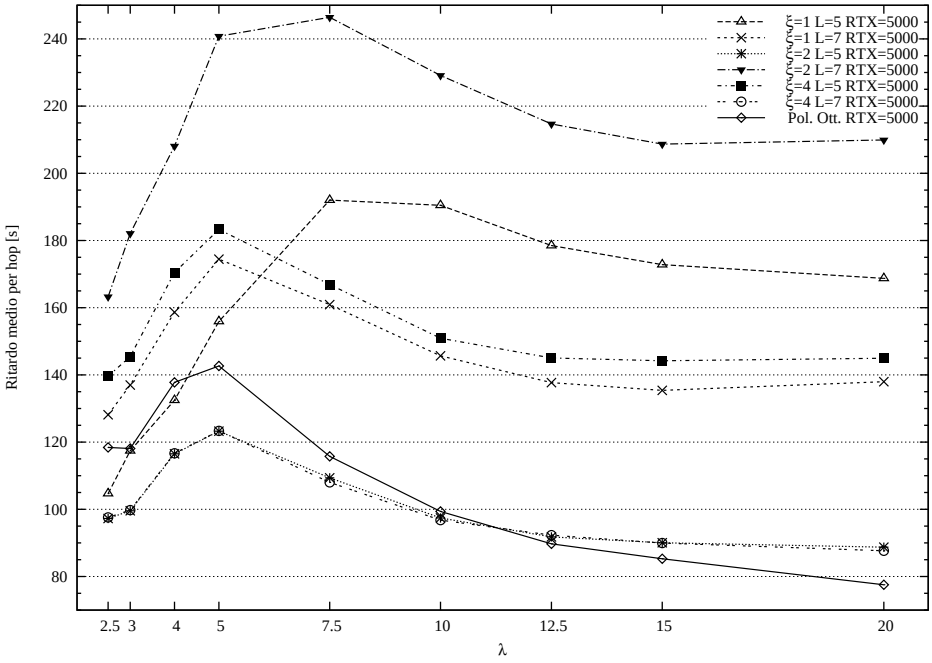

Figura 7.7: Il ritardo medio di blocco per $d_{tx} = 5000$.

Il ritardo medio di trasmissione di un blocco è strettamente dipendente dal numero di *hop* e dal numero medio di pacchetti codificati trasmessi in una sessione. Per valori di λ bassi, tali per cui il numero di *hop* non abbia il valore atteso, i ritardi subiscono un andamento simile a quello delle curve in Figura 7.6 . Nelle topologia ad alta densità è invece il numero di pacchetti codificati spediti ad essere determinante. Si nota infatti che sebbene il numero di *hop* cali rapidamente da $\lambda = 5$ a $\lambda = 20$ esso è controbilanciato dall'aumento del numero di pacchetti codificati trasmessi e del numero di round di sessione; il risultato è un andamento più morbido delle curve come si nota in Figura 7.7 . Come detto in precedenza, la *policy* ottima è ottimizzata per inviare il minor numero di pacchetti codificati in cinque round di trasmissione e non per minimizzare il numero di round di trasmissione. Per questo motivo il suo ritardo è lievemente maggiore del caso $\xi = 4$. Tuttavia all'aumentare di λ tale differenza diminuisce fino al cambiamento di tendenza per $\lambda = 20$.

7.1.2 Il confronto tra *single hop* e *multi hop*

Il numero medio di pacchetti codificati trasmessi per sessione

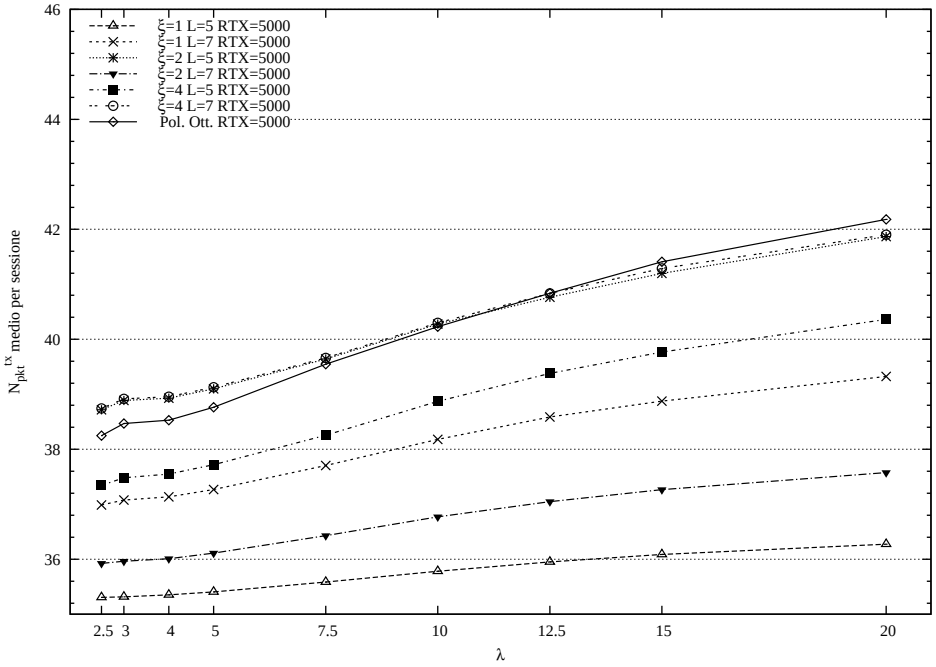

Figura 7.8: Il numero medio di pacchetti codificati trasmessi per sessione, $d_{tx} = 5000$.

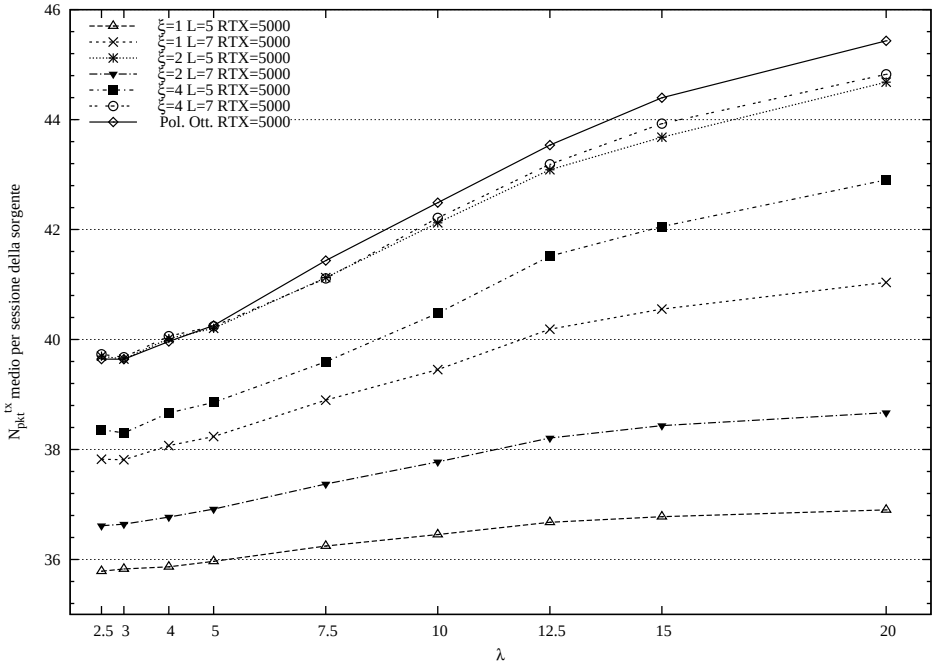

Figura 7.9: Il numero medio di pacchetti cod. trasmessi dalla sorgente per sessione, $d_{tx} = 5000$

Si nota come entrambe le figure abbiano un andamento quasi direttamente proporzionale a λ, a ξ ed a L. All'aumentare dei nodi presenti in copertura è più alta la probabilità che uno di essi non riesca a decodificare correttamente, e quindi sia maggiore il numero di pacchetti codificati richiesti a tale scopo. Tale numero poi è limitato superiormente da ξ e da L: solo la *policy* ottima e i casi $\xi = 4$, $L = 5, 7$ sono in grado di trasmetterne un numero sufficiente al completamento dei nodi ricevitori.
Se l'andamento generale è simile, si riscontra una lieve differenza tra le due metriche: la metrica single-hop richiede un quantitativo maggiore di pacchetti. Questo comportamento è dovuto al fatto che in un caso *single-hop* il trasmettitore deve riuscire a correggere *tutti* i nodi presenti nel raggio di copertura d_{tx} poichè *tutti* i vicini presenti sono incompleti. Nel caso *multi-hop* invece è necessario un numero minore, in quanto all'aumentare della densità aumenta anche la probabilità di trovare dei nodi *parzialmente incompleti* nella propria area di copertura.

Il numero medio di round per sessione

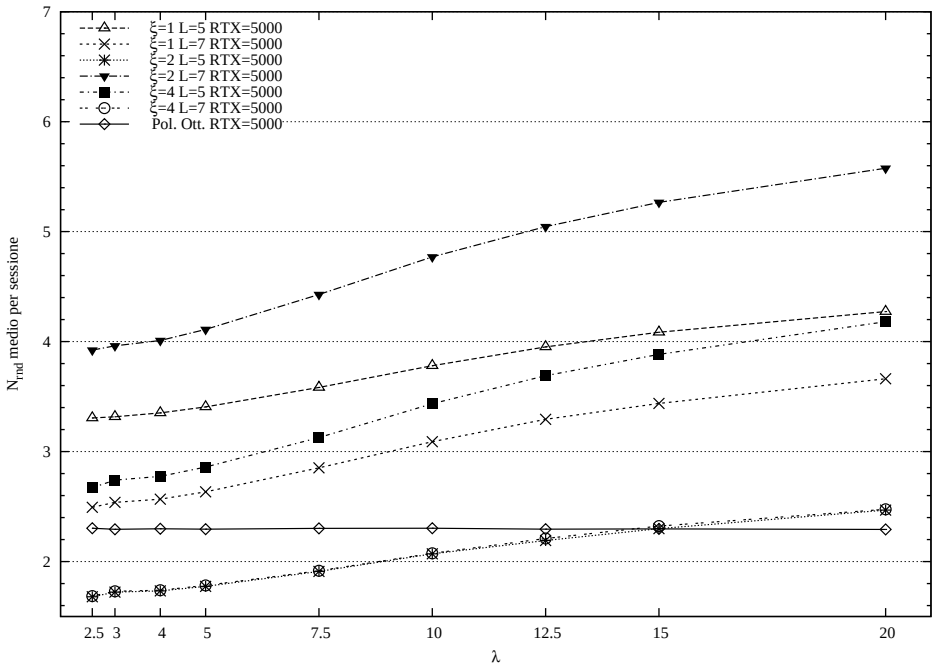

Figura 7.10: Il numero medio di round per sessione $d_{tx} = 5000$.

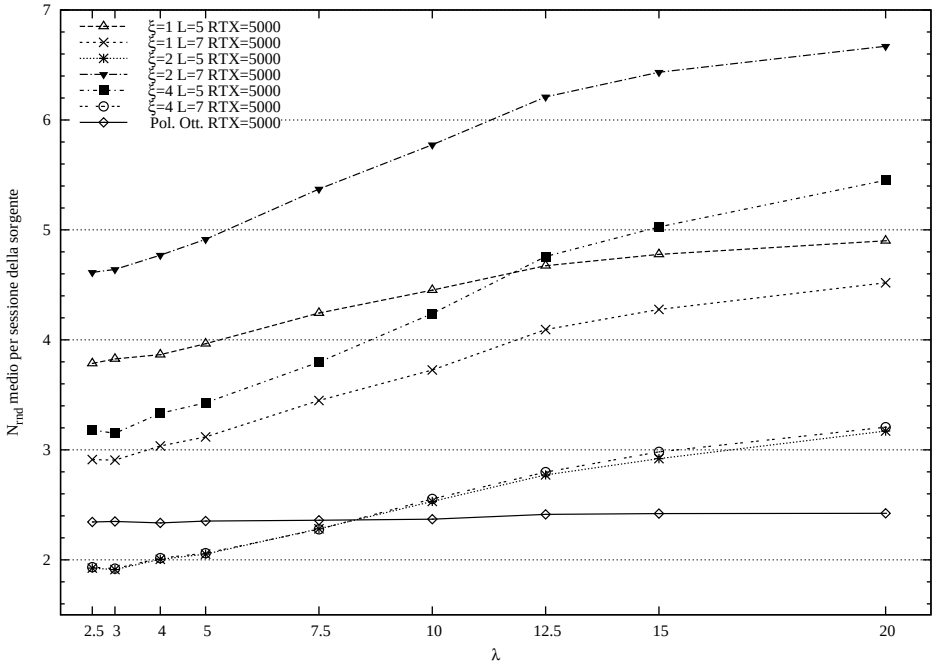

Figura 7.11: Il numero medio di round per sessione di sorgente $d_{tx} = 5000$.

Anche il numero medio di round per sessione ha una famiglia di curve con le medesime caratteristiche di quelle descritte nella sottosezione precedente. È presente una proporzionalità quasi diretta con λ , ξ , L. La dinamica *single-hop* inoltre presenta dei valori più elevati rispetto a quella *multi-hop*, per lo stesso motivo descritto in precedenza.
Se si osserva infine il confronto per $d_{tx} = 5000$ si nota la totale uguaglianza delle due curve relative alle *policy* ottime. Tale evento non deve sorprendere, ma deve essere letto come un'ulteriore conferma della *prudenza* della stessa. Si sarebbe portati a pensare infatti che in ambiente *multi-hop*, avendo un numero di nodi parzialmente incompleti direttamente proporzionale a λ il numero di round dovrebbe diminuire. Esso è costante $\approx 2,2$ perchè tale è il numero di round che la *policy* impiega ad adattarsi e risolvere la particolare topologia che si trova ad affrontare.
La probabilità d'errore sul pacchetto e la probabilità d'invio di pacchetti codificati linearmente dipendenti con quelli già ricevuti sono tra i fattori che determinano tale comportamento.

La probabilità media di fallire una sessione

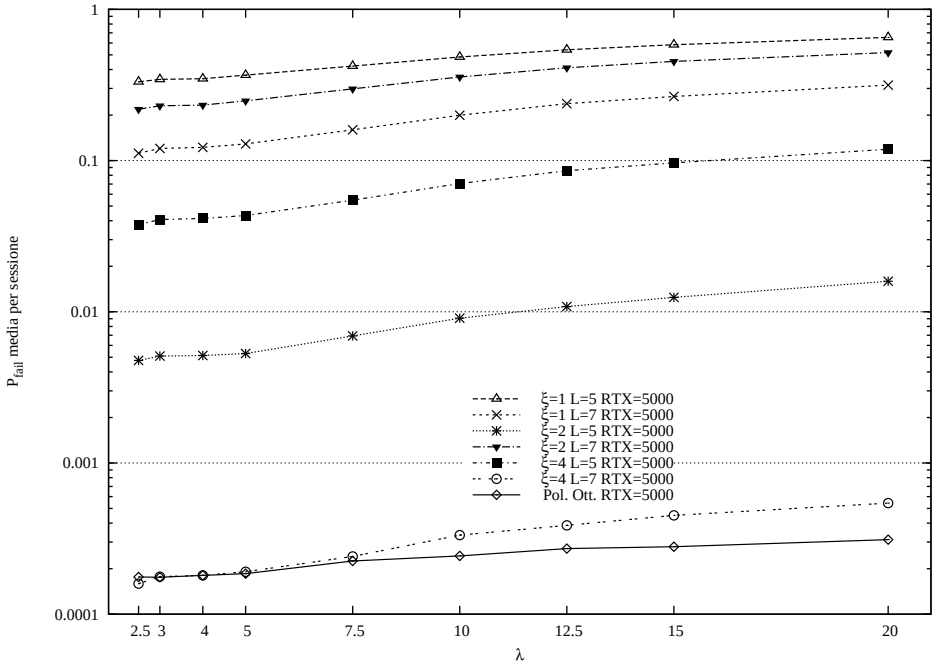

Figura 7.12: La probabilità media di fallire una sessione, $d_{tx} = 5000\ m$.

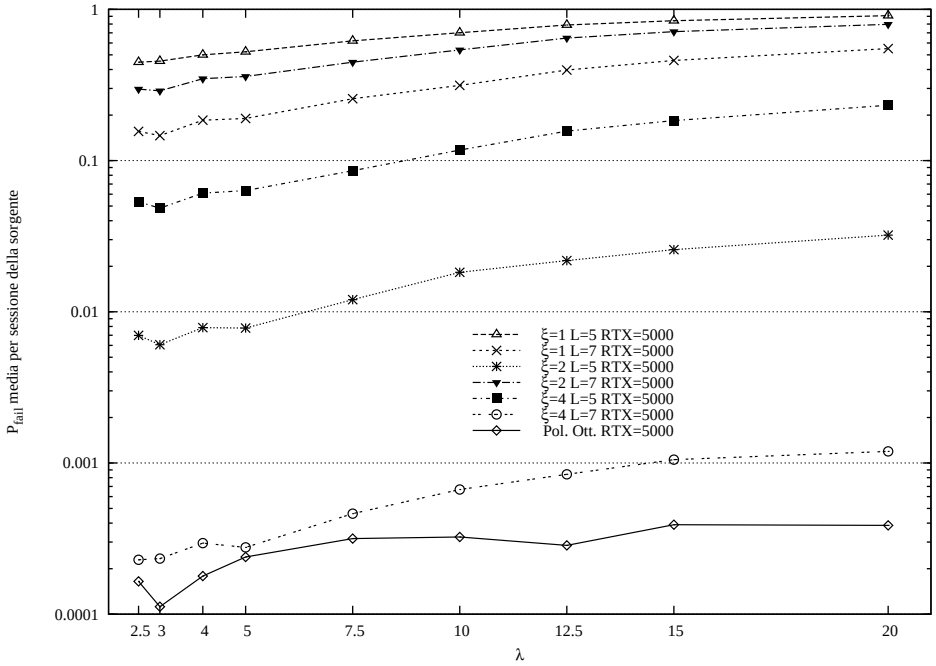

Figura 7.13: La probabilità media di fallire una sessione della sorgente, $d_{tx} = 5000\ m$.

Anche per tale metrica valgono le considerazioni dette in precedenza, riguardo alla diretta proporzionalità ed alla differenza tra i due ambienti considerati. Tale differenza è minima per valori di $\lambda \leq 5$ mentre è più marcata altrimenti.
Si fa notare come la *policy* ottima ottenga la probabilità di fallimento più bassa, minore addirittura del caso $\xi = 4$, $L = 7$, e come essa sia pressochè costante. Tutto ciò è un ulteriore conferma della bontà della stessa.

La percentuale media di vicini completi in una sessione

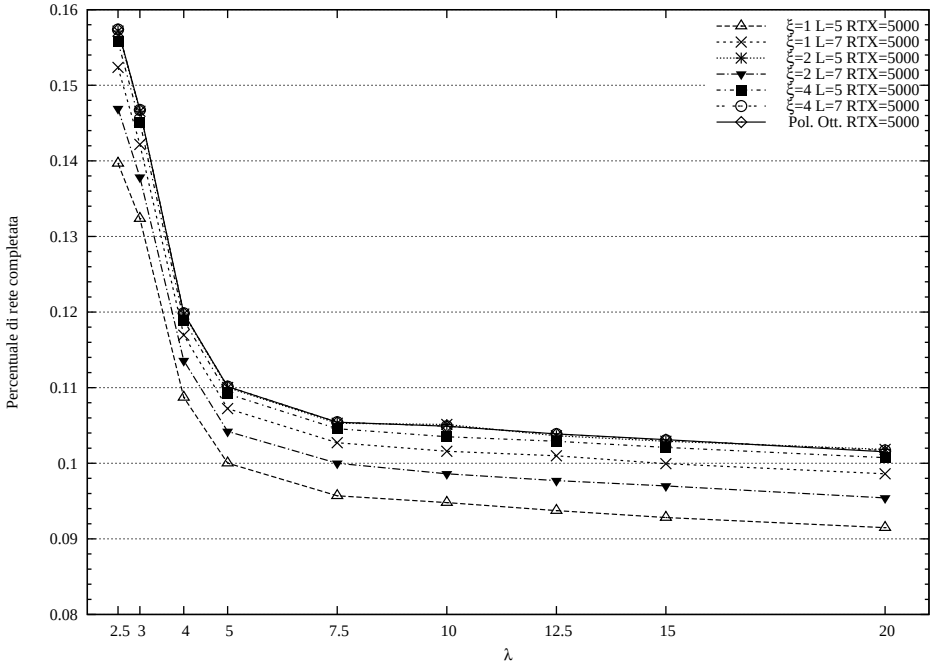

Figura 7.14: La percentuale media di vicini completi dopo una sessione, $d_{tx} = 5000\ m$.

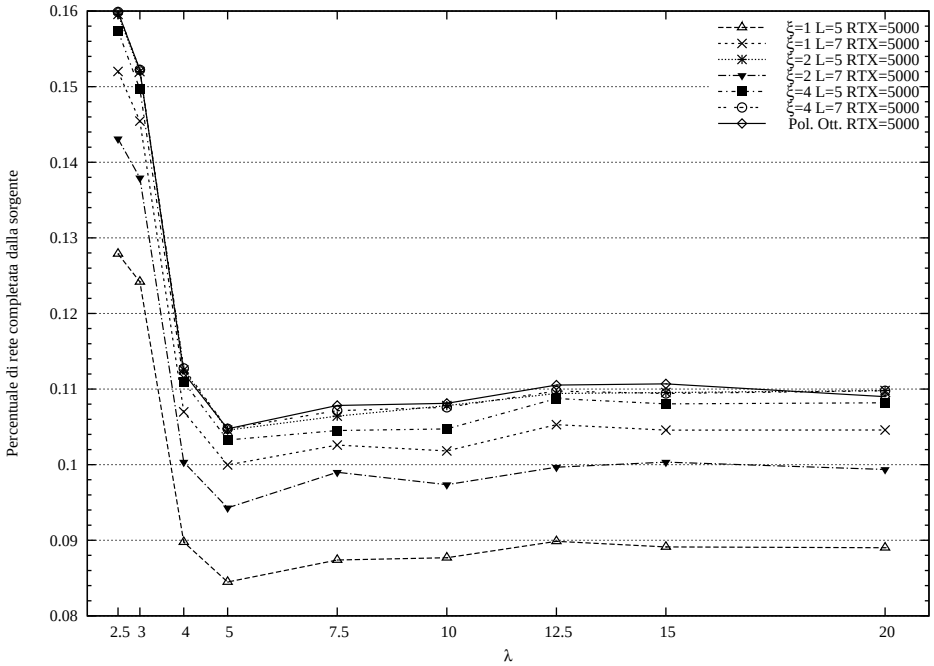

Figura 7.15: La percentuale media di vicini completi della sorgente dopo una sessione, $d_{tx} = 5000\ m$.

Con tale metrica si vuole definire il numero medio di vicini completi presenti nell'area del cerchio di raggio d_{tx}, anche detta area di copertura. Si nota immediatamente come esso sia strettamente dipendente dalla massima ridondanza trasmissibile, e quindi da ξ e da L. Maggiori i due parametri, maggiore la probabilità di completare un nodo.
Il comportamento *single-hop* è semplice: i nodi completi nell'area di copertura sono i *soli* che esso stesso ha provveduto a portare a buon fine. Nel caso *multi-hop* invece potrebbe capitare il caso in cui essi fossero già presenti perchè completati da altri nodi in precedenza.
Se la percentuale della sorgente si stabilizza (e quindi in numero assoluto aumenta) si nota invece come quella media sia decrescente (il numero assoluto cresce più lentamente): è la somma cioè di due eventi contrastanti:

1. all'aumentare della densità aumenta la probabilità che i nodi vicini completino i nodi parzialmente incompleti di precedenti sessioni. Tale evento porta ad una crescita della metrica considerata.

2. All'aumentare della densità la dinamica della diffusione dell'informazione codificata diventa più imprevedibile: si possono formare *bolle* temporanee di nodi parzialmente o totalmente incompleti; la loro connessione alla rete di noti già completa è molto probabile che dipenda da un insieme ristretto di nodi sul bordo dell'area di copertura. Nell'istante in cui uno di quest'ultimi venga completato si ottiene una catena di sessioni di trasmissione successive fino al completamento della bolla. In questa fase i nodi trasmettitori non possono contare sull'aiuto di altri nodi completi, in quanto se così fosse la bolla non sarebbe nemmeno esistita: le statistiche registate sono quindi minori della media. L'evento appena descritto ha frequenza maggiore con all'aumentare della densità λ e porta ad una diminuzione della metrica considerata.

3. All'aumentare della densità aumenta la probabilità di avere nodi in ricezione in prossimità del bordo dell'area di copertura. Tale evento porta ad una diminuzione della metrica considerata.

La metrica è infine massima utilizzando la *policy* ottima.

L'avanzamento medio

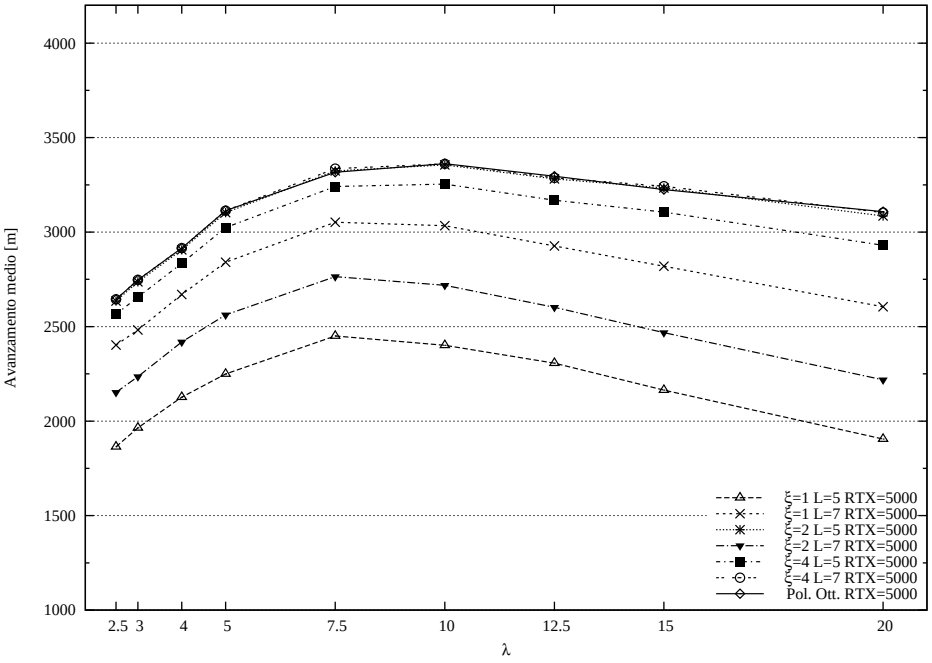

Figura 7.16: L'avanzamento medio, $d_{tx} = 5000\ m$.

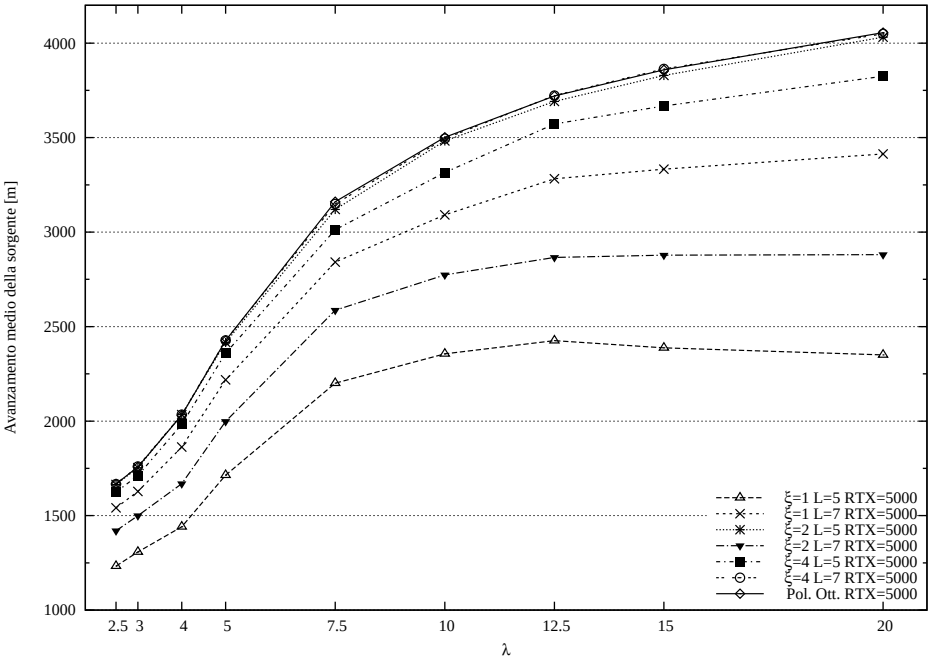

Figura 7.17: L'avanzamento medio di sorgente, $d_{tx} = 5000\ m$.

L'avanzamento medio, definito in Sezione 5.3.4, può essere considerato come una prova ulteriore della teoria esposta nella sottosezione precedente. Appurata infatti la proporzionalità diretta da ξ e L come nei casi precedenti, si nota come tale metrica sia crescente per il caso *single-hop* e una funzione concava nel caso *multi-hop* al variare di λ.
Si consideri il caso della *policy* ottima per $d_{tx} = 5000$, si nota come nel caso *single-hop* la crescita sia di circa $\Delta \approx +3250\ m$. Nel caso *multi-hop*:

- per $\lambda \leq 7.5$ si ha che l'effetto contrario già citato è assente. L'avanzamento medio infatti è maggiore del caso a *hop* singolo.

- per $\lambda \geq 10$ l'effetto negativo comincia ad essere presente, si nota infatti come l'avanzamento medio per $\lambda = 20$ sia inferiore rispetto alla sua controparte di $\Delta \approx 1000\ m$.

Il raggio di completezza medio

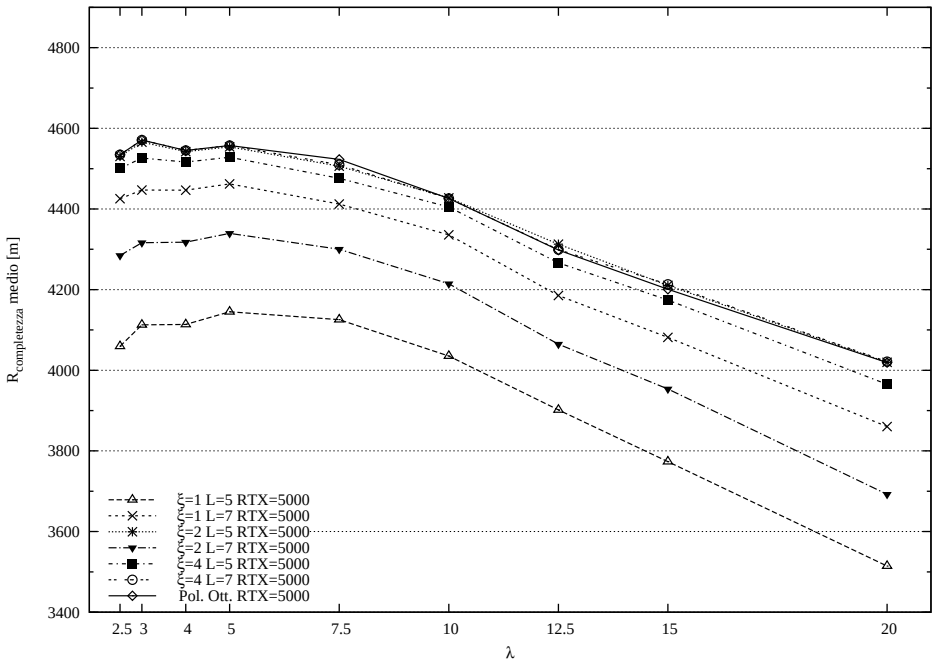

Figura 7.18: Il raggio di completezza medio, $d_{tx} = 5000\ m$.

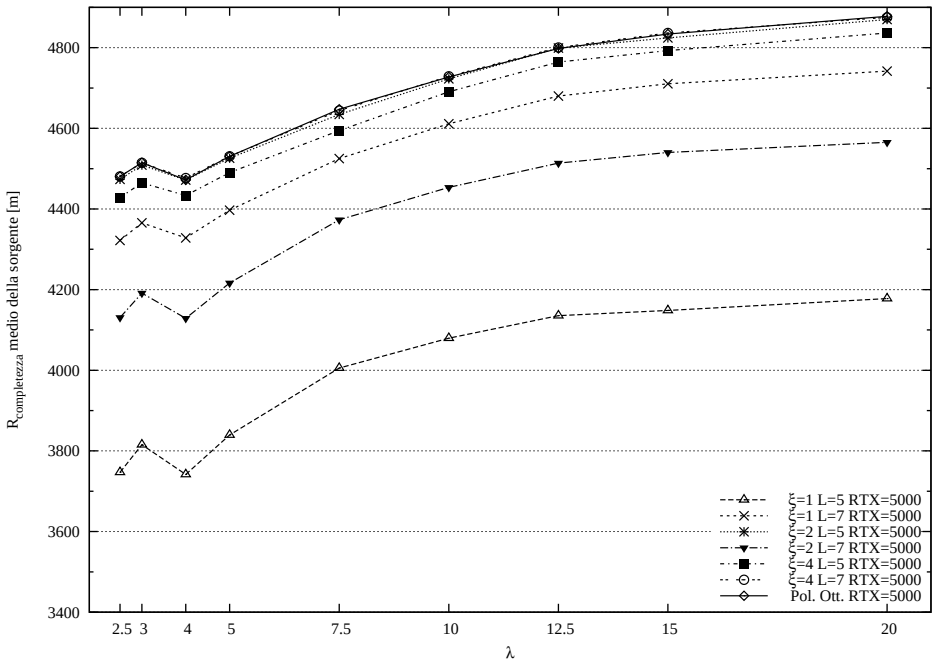

Figura 7.19: Il raggio di completezza della sorgente medio, $d_{tx} = 5000\ m$.

Il raggio di completezza, anch'esso definito in Sezione 5.3.4, è l'ultima conferma della teoria esposta in Sezione 7.1.2. Si consideri anche in questo caso la *policy* ottima per $d_{tx} = 5000$:

- nel caso *single-hop* il raggio di completezza è crescente e la differenza tra il valore massimo ed il valore minimo di ogni curva è $\Delta \approx 500\ m$;
- nel caso *multi-hop* il raggio di completezza ha valori simili a quelli della controparte *single-hop* per $\lambda \leq 5$. Per $\lambda \geq 10$ l'effetto negativo della teoria proposta ha sempre più peso. Poichè in questo caso si considera un'area invece che una distanza orientata come nel caso precedente, l'effetto sarà più mitigato. La differenza tra il valore massimo e quello mininimo infatti è c.ca $\Delta \approx 400\ m$.

Considerazioni Finali

Il confronto tra *single-hop* e *multi-hop* ha dato esiti parzialmente positivi: in reti a bassa densità di nodi infatti i due comportamenti possono essere ritenuti approssimabili e la predizione dell'ambiente *multi-hop* da quello *single-hop* è possibile: se si vuole coprire ad esempio una distanza lineare dal centro della rete al suo bordo, per $\lambda \leq 5$, la si può suddividere in un numero di avanzamenti medi *single-hop* ottenendo un quoziente q. In questo caso le metriche *multi-hop* d'interesse saranno multiple di quelle *single-hop* moltiplicate per q.
Per $\lambda \geq 7.5$ invece, a causa dell'effetto dovuto alla teoria esposta in Sezione 7.1.2, si commette un'errore dell'ordine del 5%.

7.2 Il protocollo *CRBCast*

Si prosegue la trattazione con l'analisi del protocollo *CRBCast*, con particolare attenzione alle modifiche proposte ed al confronto con il protocollo *OFBP*. I risultati seguenti sono tutti e soli quelli relativi al *range* di trasmissione $d_{tx} = 5000\ m$, data la proporzionalità diretta degli stessi da d_{tx}. Si è optato inoltre per un valore di $n_p\gamma = 40$ pacchetti codificati trasmessi. Se si osserva infatti la Figura 7.8, si nota come tale valore sia sub-ottimo ma ragionevole per l'utilizzo in topologie di ogni densità. I valori di p_{fwd} utilizzati sono quelli esposti in Figura 5.5 .

7.2.1 Il numero medio di pacchetti codificati trasmessi

Nella Fase I originale è noto come i nodi che abbiano ricevuto correttamente almeno un pacchetto codificato del set della sorgente possano tentare il *broadcast* probabilistico, mentre nella Fase I modificata i soli nodi completi possono tentare l'inoltro probabilistico tramite l'invio di $n_p\gamma$ nuovi pacchetti codificati. Si ha così che la curva appartenente alla Fase I modificata sia costante a $n_p\gamma = 40$ come atteso, mentre la curva della Fase I originale oscilla in un intorno di $n_p\gamma \simeq 38.2$ in base alla distanza più o meno grande ed alla densità di nodi più o meno maggiore dei nodi. La figura successiva mostra tutte le combinazioni possibili delle due Fasi: in particolare si nota come la *policy* utilizzata nella Fase II originale sia molto aggressiva e poco efficiente, in entrambi i casi di Fase I originale e modificata:

- nel primo caso si ha una maggiore diffusione di pacchetti codificati e quindi un numero maggiore di nodi parzialmente incompleti. Il numero di nuovi pacchetti codificati è minore.

- nel secondo caso invece si ha una diffusione meno capillare dell'informazione, anche se essa sfrutta la diversità messa a disposizione dai codici a fontana. In questa modalità i nodi o sono completi (o quasi) o sono totalmente incompleti. Tale scena34rio ben si adatta all'andamento costante della curva in figura.

Si nota al contrario come l'utilizzo della *policy* ottima migliori nettamente l'efficienza. Anche qui la curva relativa alla Fase I modificata è maggiore di quella originale per i motivi sopra descritti.

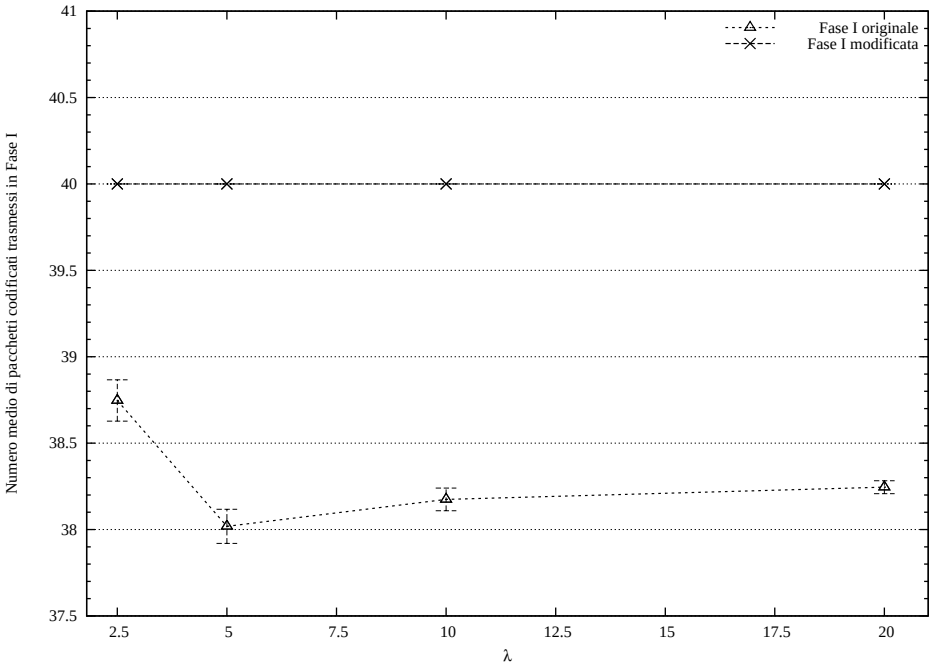

Figura 7.20: N_{tx} delle Fasi I considerate.

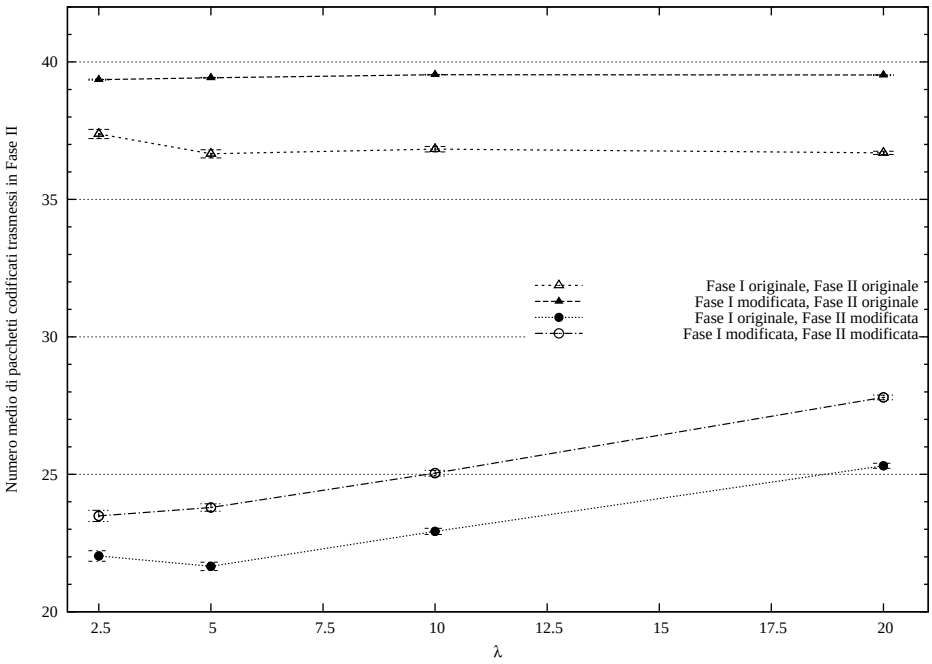

Figura 7.21: N_{tx} delle Fasi II considerate.

7.2.2 La probabilità media di fallire un round di trasmissione

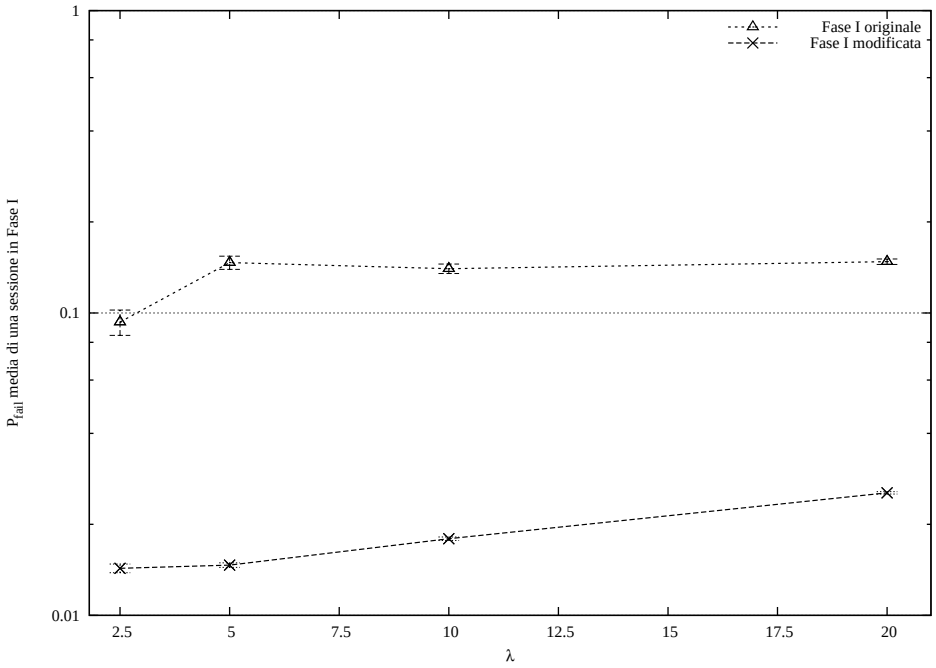

Figura 7.22: asd

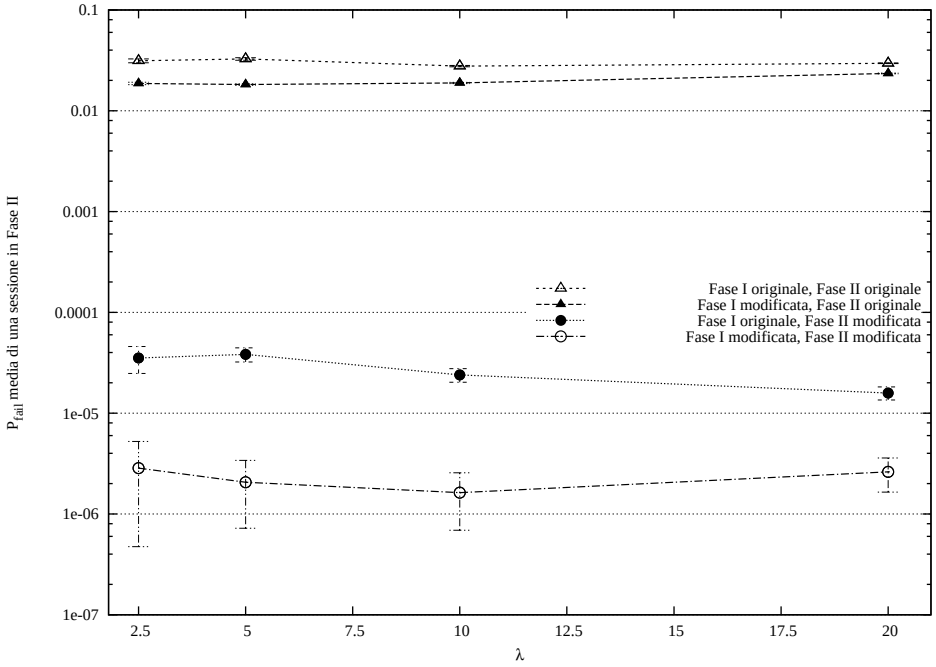

Figura 7.23: asd

Tale probabilità mostra l'importanza di introdurre modifiche protocollari su *CRB-Cast*. Si prenda ad esempio la Fase I: l'invio dello *stesso* set di pacchetti codificati a tutta la rete non è una buona scelta e la probabilità di fallire maggiore di circa un ordine di grandezza ne è testimone. La possibilità di inviare *nuovi* pacchetti codificati deve essere sempre sfruttata. La seconda figura invece mostra ancora una volta l'efficienza della *policy* ottima presentata in questa sede. Nei casi della della Fase II modificata la curva rappresenta la probabilità di fallire un'*intera sessione* di recupero: si noti ancora una volta come la curva relativa alla combinazione (Fase I modificata, Fase II modificata) sia l'opzione migliore per la metrica considerata, migliore rispetto alla combinazione (Fase I originale , Fase II modificata) di quasi un ordine di grandezza.

7.2.3 Le sessioni sprecate

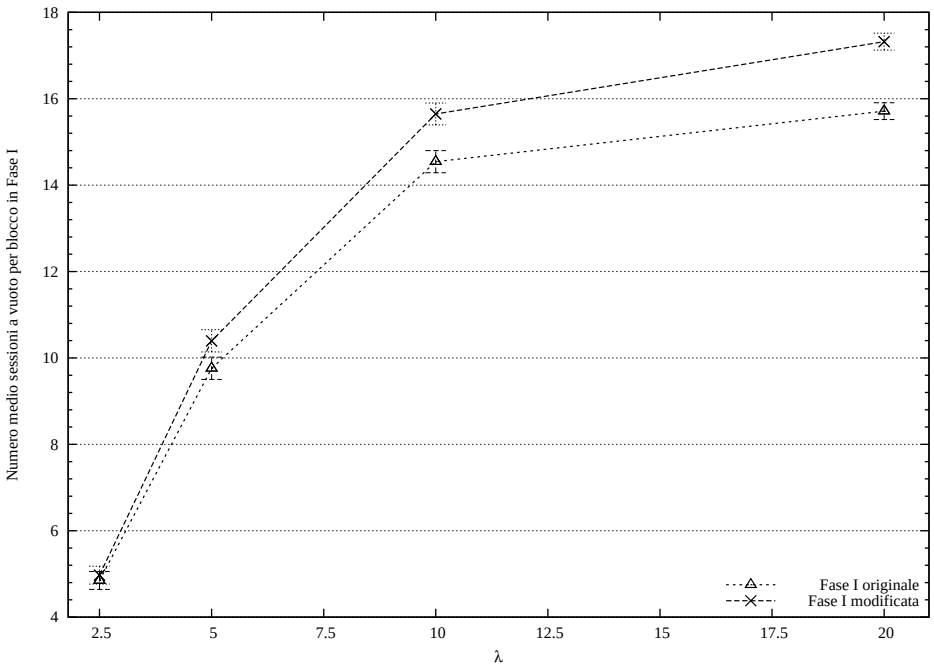

Figura 7.24: Le sessione andate a vuoto durante la Fase I

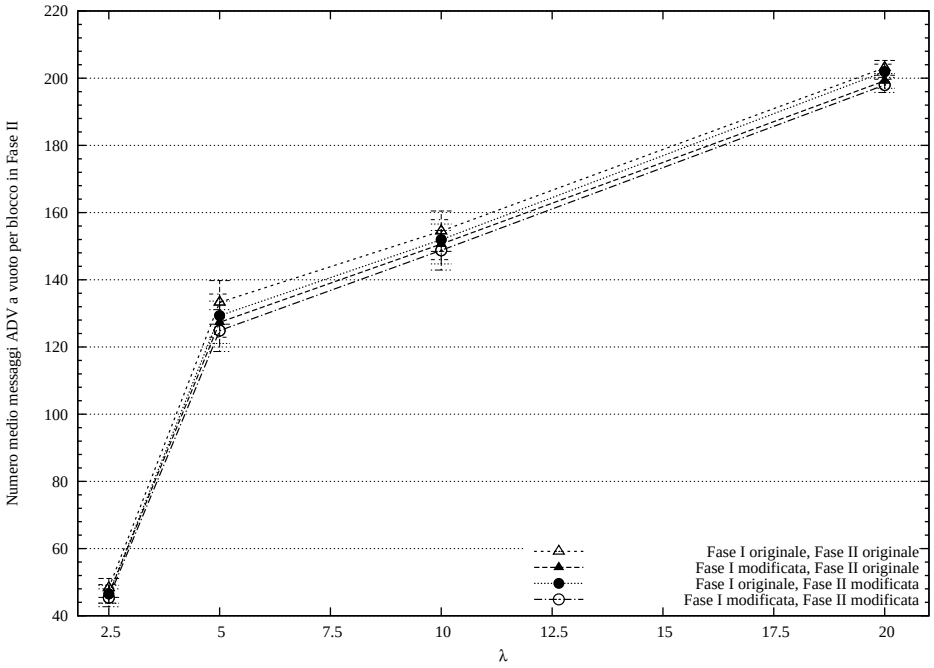

Figura 7.25: I messaggi di ADV andati a vuoto idurante la Fase II

La coppia di figure mostra l'efficienza del protocollo nelle due Fasi: nella prima, il numero medio di sessioni andate a vuoto[2] è maggiore nella modifica proposta. Ciò è da imputarsi semplicemente al numero medio di pacchetti trasmessi che come visto in precedenza è maggiore nella Fase I modificata. Una maggiore efficenza nella distribuzione di pacchetti codificati linearmente indipendenti viene controbilanciato quindi da una minore efficienza in trasmissione. La diretta proporzionalità delle metriche con λ è visibile in entrambe le figure.
I messaggi di ADV andati a vuoto sono praticamente coincidenti per tutte e quatto le casistiche considerate, tale metrica infatti dipende fortemente dalla particolare topologia ed aumenta in maniera lineare per $\lambda \geq 5$.

[2]Si ricorda che una sessione è sprecata quando non vi sono ricevitori presenti nell'area di copertura o sono già completi.

7.2.4 Le efficienze di tramissione

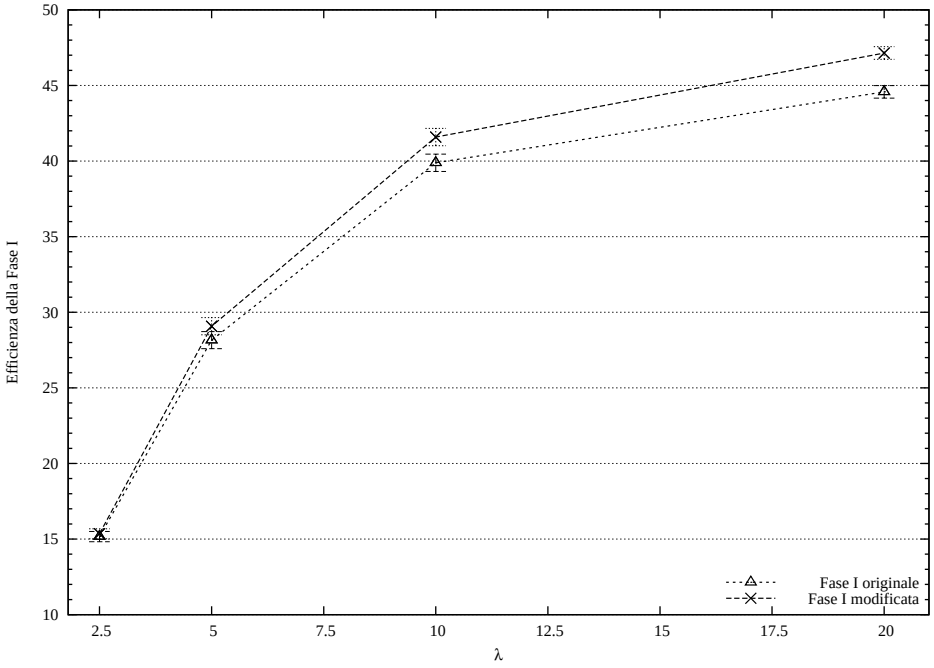

Figura 7.26: L'efficienza di trasmissione della Fase I.

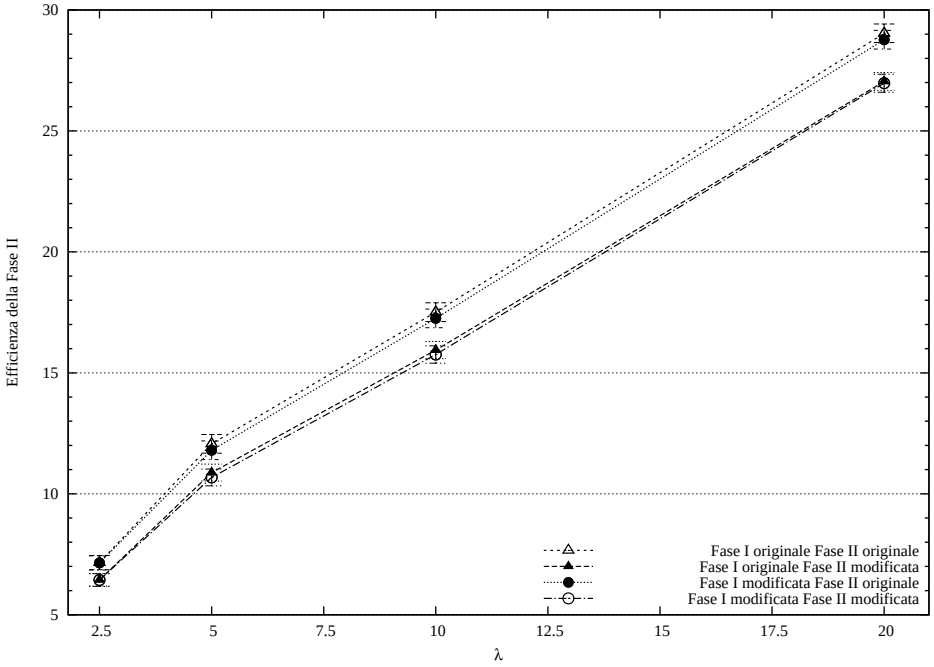

Figura 7.27: L'efficienza di trasmissione della Fase II.

Si consideri l'efficienza di trasmissione ovvero il numero di pacchetti codificati necessari per completare l'intera rete per pacchetto d'informazione. La fase I originale e quella modificata sono simili per $\lambda \leq 5$, mentre la prima si distacca leggermente per $\lambda \geq 10$: il motivo è quello descritto in precedenza: si ha uno spreco di un numero maggiore di pacchetti codificati nelle sessioni andate a vuoto della fase modificata. Nella figura relativa alla Fase II si distinguono due famiglie: quella delle curve con la Fase II originale e quelle delle curve aventi Fase II modificata. La seconda è ovviamente più efficiente, anche se il divario non è così netto come si sarebbe potuto attendere.
Si noti infine come le curve della Fase I tendano a saturare al crescere di λ, mentre al contrario quelle della Fase II crescano linearmente con essa.

7.2.5 I ritardi

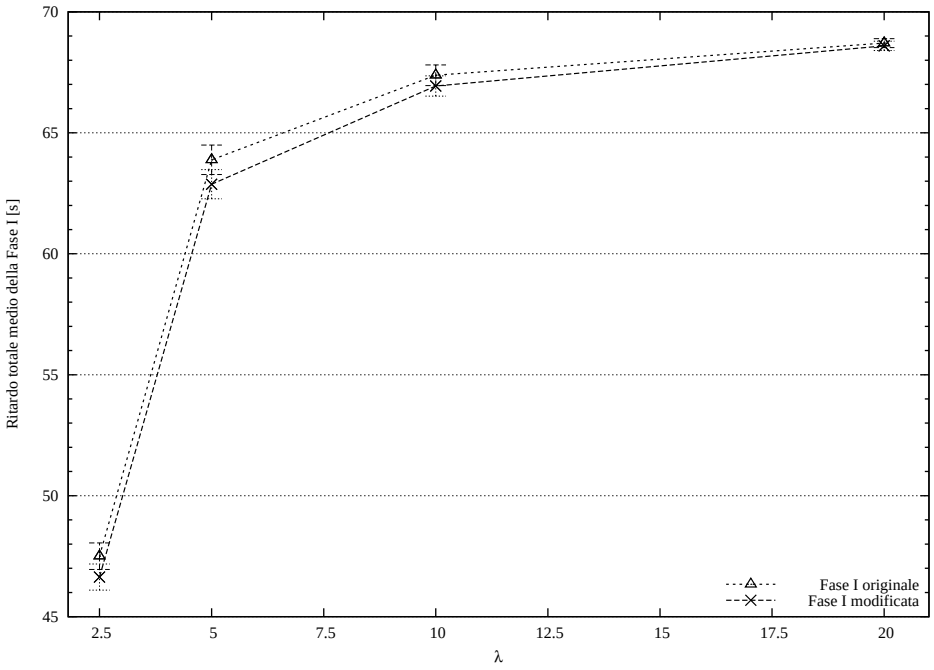

Figura 7.28: Il ritardo totale della Fase I.

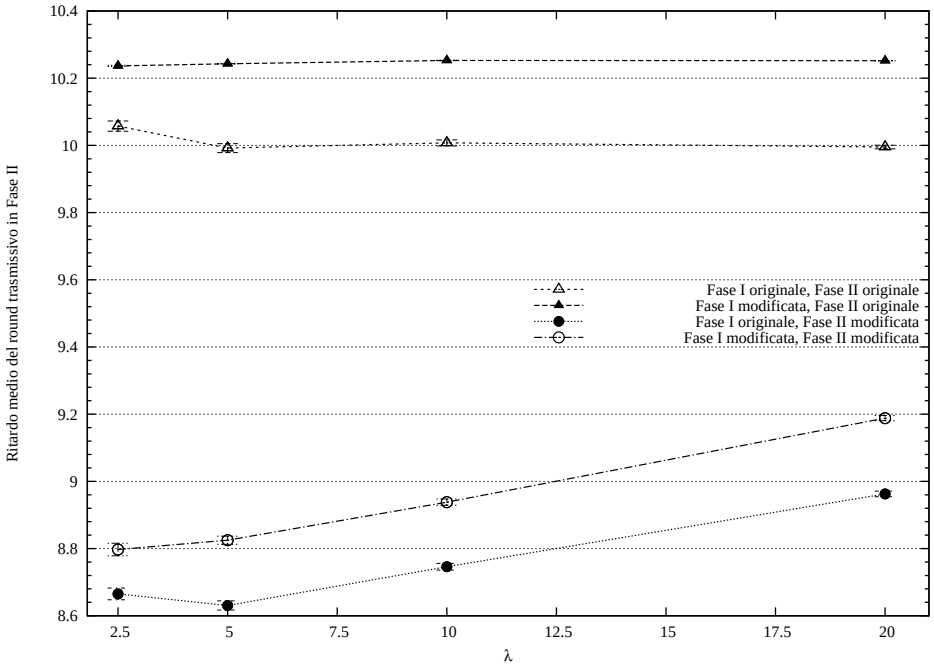

Figura 7.29: Il ritardo di un round di trasmissione della Fase II.

La prima figura mostra il ritardo totale per blocco d'informazione della Fase I; si noti come esso tenda a saturare all'aumentare di λ: la ragione è da ricercarsi nell'aumento del grado di parallelizzazione delle trasmissioni tipica delle reti ad alta densità. Si noti infatti come si passi dai ≈ 45 s per $\lambda = 2.5$ ai quasi 65 s per $\lambda = 5$, mentre tra $\lambda = 5$ e $\lambda = 20$ il divario sia di soli ≈ 5 s. Si noti infine come per $\lambda \geq 10$ tale ritardo sia già da solo comparabile al ritardo complessivo del protocollo proposto (Figura 7.7).
Nella seconda figura si mostrano i ritardi medi di un singolo round di trasmissione della Fase II. Si nota chiaramente come il ritardo di una sessione della Fase II modificata impieghi meno tempo di quella originale: ciò è dovuto all'efficienza della stessa, al minor numero cioè di pacchetti codificati trasmessi. La *policy* originale al contrario presenta una curva piatta in quanto in media trasmette lo stesso quantitativo di pacchetti codificati indipendentemente da λ. Si nota come le curve delle Fasi II relative alla Fase I originale siano inferiori rispetto alle controparti: ciò è da imputare ancora una volta al numero minore di pacchetti trasmessi.

7.2.6 La copertura della rete

La Fase I

In Figura 7.30 sono presenti le densità di probabilità della percentuale di nodi coperti al termine della Fase I per ogni valore di λ. Si riscontra subito una conferma della corretta ottimizzazione della probabilità di p_{fwd}: la copertura media tende ad essere molto elevata per tutti i valori di λ. Per ciascuna coppia di curve inoltre si nota come la Fase I modificata abbia una maggiore probabilità di ottenere una copertura più ampia: sfruttare la diversità nei pacchetti codificati trasmessi è quindi essenziale per ottenere una maggiore copertura. È interessante inoltre come le curve tendano ad appiattirsi all'aumentare di λ, tale effetto è dovuto al maggior numero di disposizioni possibili dei nodi nella rete stessa ed all'aumento conseguente della varianza della percentuale di rete completata.

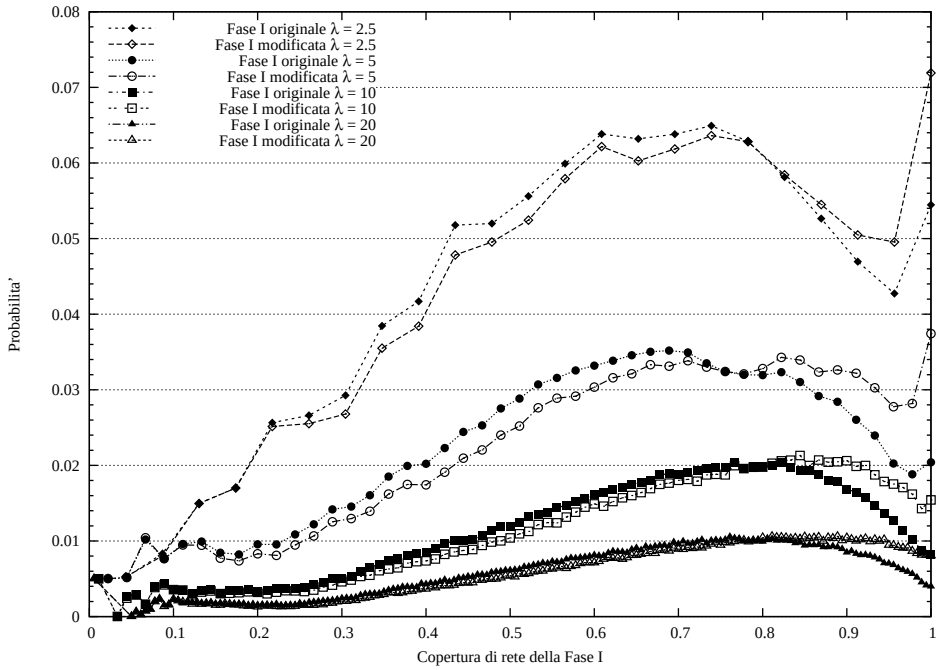

Figura 7.30: La *pdf* della copertura di rete al termine della Fase I.

La Fase II

Si confrontano ora la densità di probabilità del numero di round e la funzione di distribuzione di probabilità della percentuale di copertura di rete della Fase

II. Tale confronto è presentato separatamente per ciascun λ per mettere una più chiara lettura delle immagini stesse.

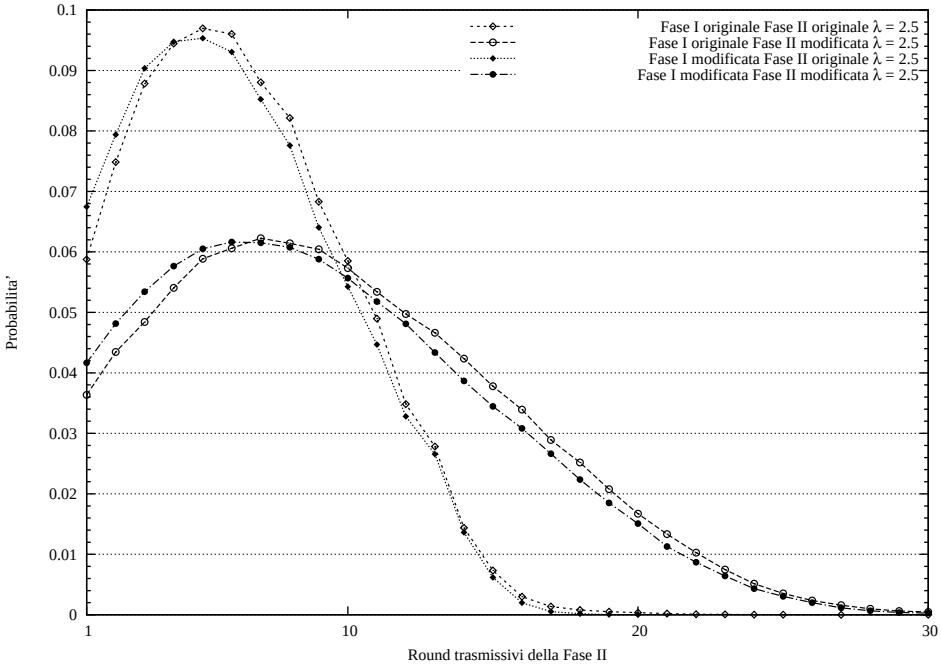

Figura 7.31: La densità di probabilità del numero di round, $\lambda = 2.5$.

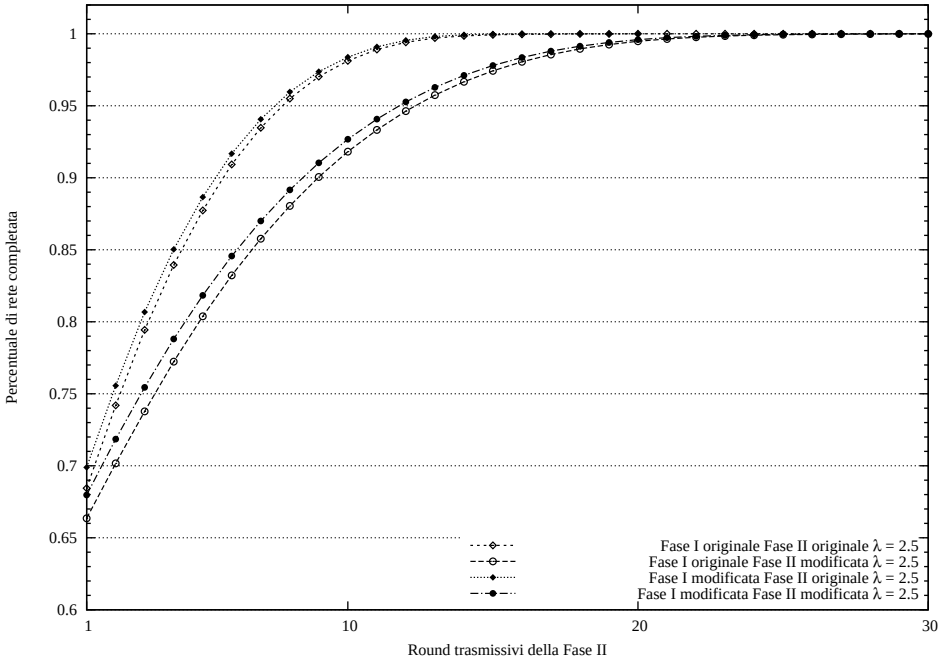

Figura 7.32: La *cdf* della percentuale di copertura di rete, $\lambda = 2.5$.

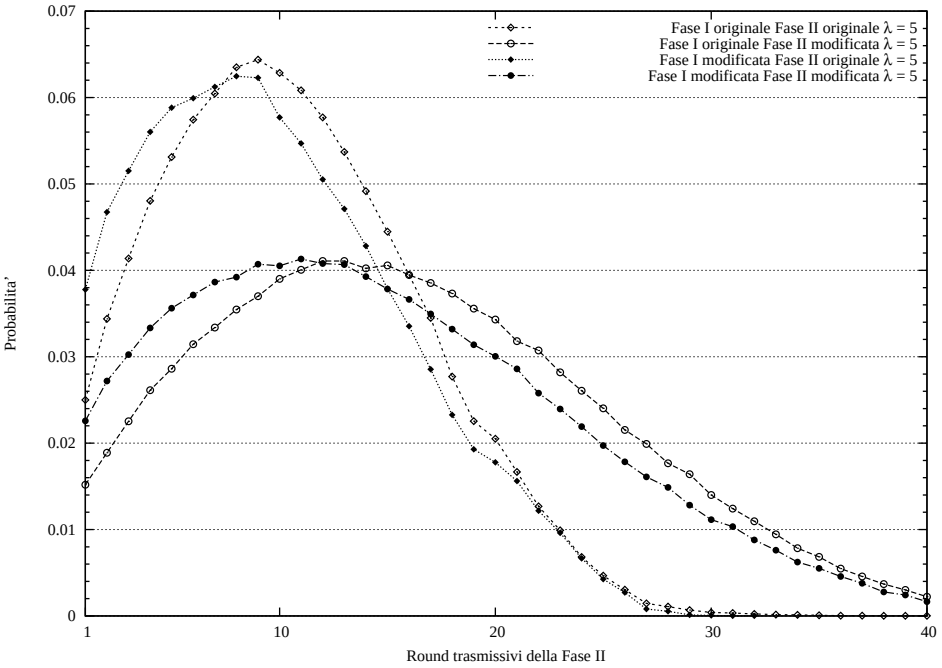

Figura 7.33: La densità di probabilità del numero di round, $\lambda = 5$.

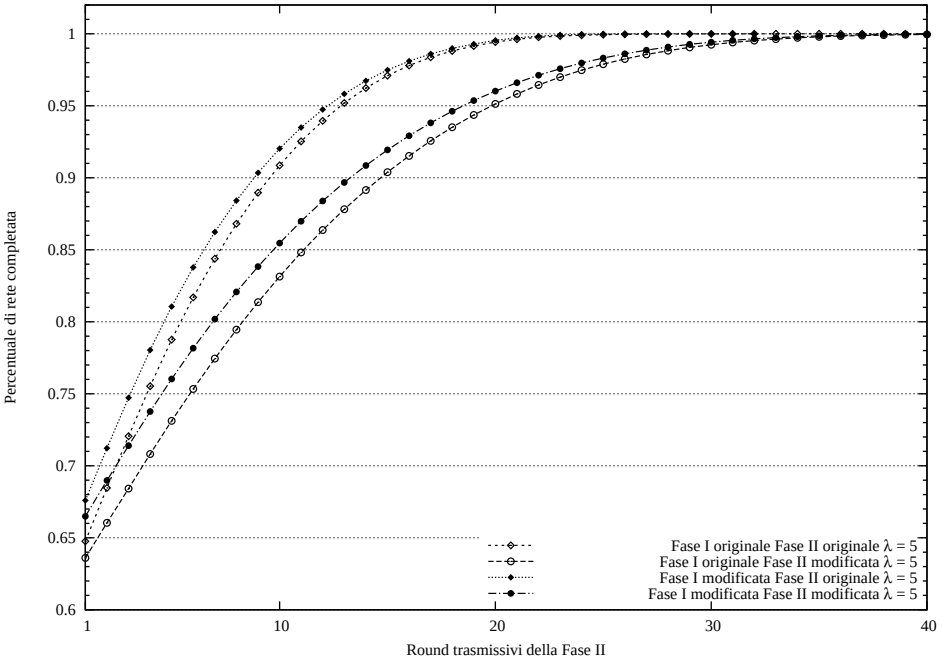

Figura 7.34: La *cdf* della percentuale di copertura di rete, $\lambda = 5$.

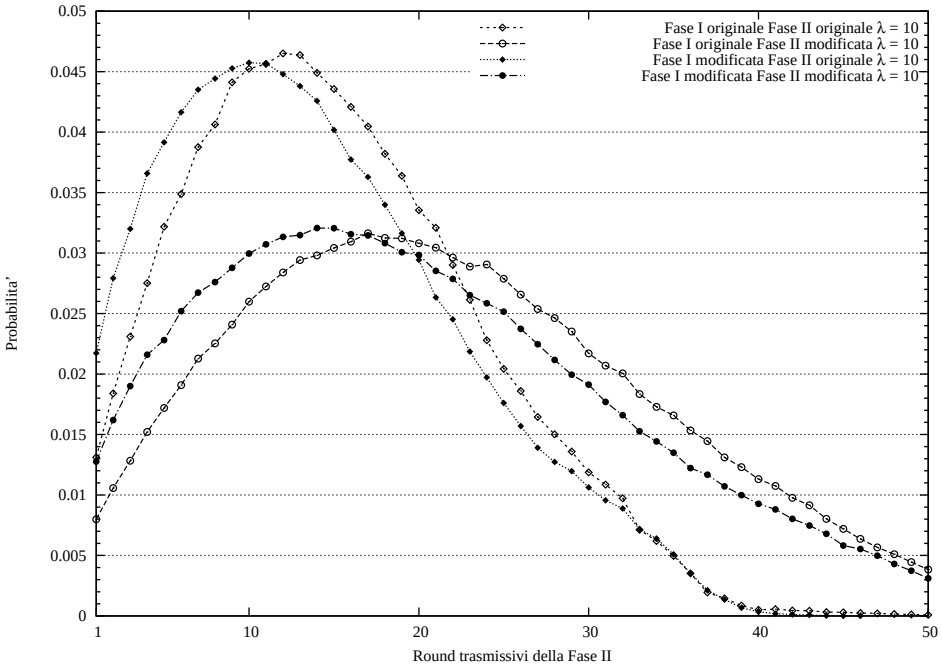

Figura 7.35: La densità di probabilità del numero di round, $\lambda = 10$.

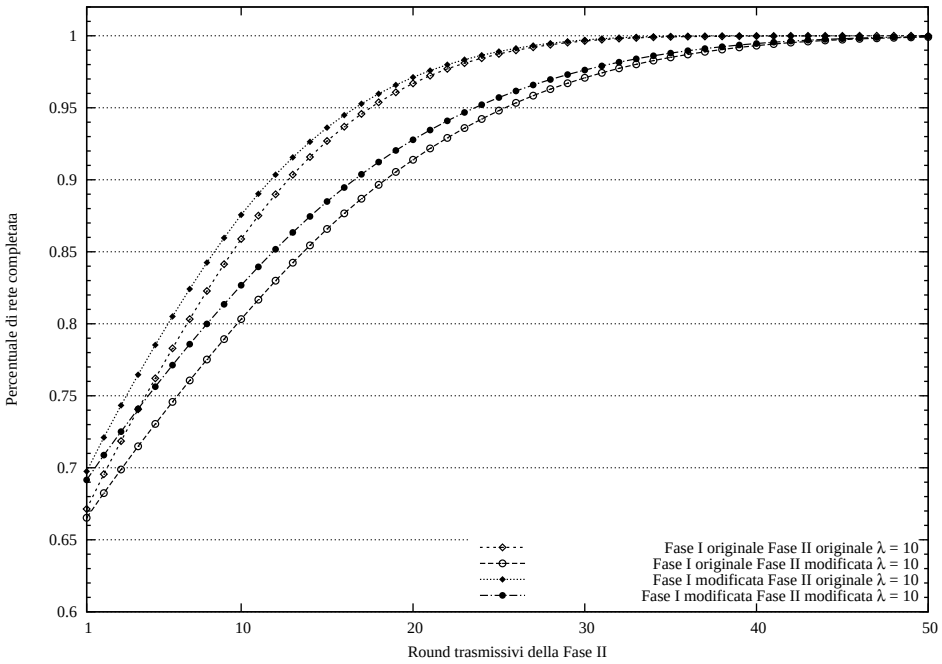

Figura 7.36: La *cdf* della percentuale di copertura di rete, $\lambda = 10$.

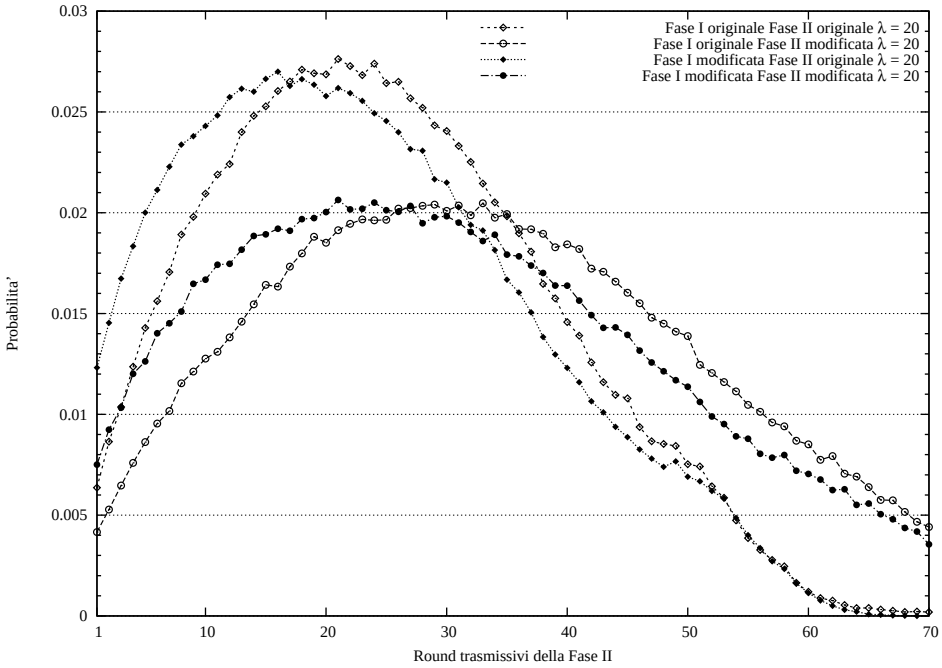

Figura 7.37: La densità di probabilità del numero di round, $\lambda = 20$.

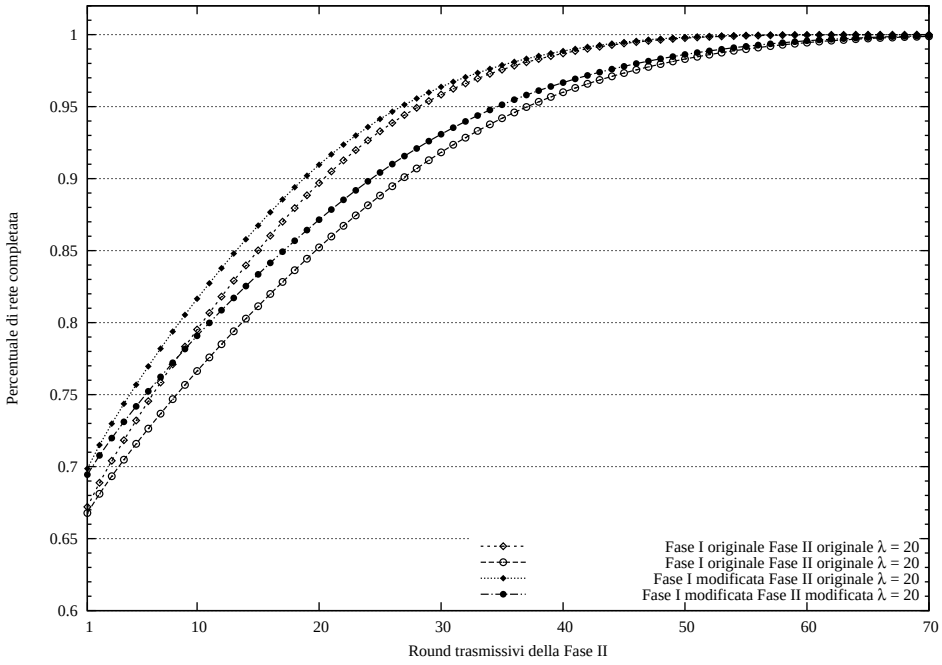

Figura 7.38: La *cdf* della percentuale di copertura di rete, $\lambda = 20$.

Si consideri la famiglia di curve delle densità di probabilità della frequenza di apparizione del numero di round. La caratteristica comune è la tendenza ad appiattirsi all'aumentare di λ: tale effetto è dovuto al maggior numero di disposizioni possibili dei nodi nella rete stessa ed all'aumento conseguente della varianza del numero di round. In tale famiglie di curve si possono individuare due ulteriori sotto famiglie: quelle in cui aventi Fase II originale e quelle aventi Fase II modificate. Le prime sono caratterizzate da un alto numero di pacchetti codificati trasmessi in fase di recupero, che si traduce in un minor numero di round di trasmissione; si nota inoltre come la Fase I modificata abbia una maggiore probabilità di terminare la fase di recupero in anticipo rispetto alla controparte originale. La seconda famiglia di curve è un ulteriore dimostrazione di quello che è stato più volte detto riguardo alla *policy* ottima. Essa è formulata per essere efficiente e per mandare quindi un numero minore di pacchetti codificati per round di trasmissione. Il numero di round per la convergenza di quest'ultima è più maggiore rispetto a quello della Fase II originale.
Si prenda in esame ora la famiglia di curva della funzione di distribuzione di probabilità della percentuale di rete coperta. La carattestica comune in questo caso è la tendenza ad aumentare il numero di round per completare l'intera rete all'aumentare di λ: si tratta di un risultato atteso, se si considera che l'aumento di λ si traduce in un aumento del numero di nodi presenti nella rete. Anche i questa famiglia si possono individuare due sottofamiglie di curve e precisamente le stesse definite in precedenza: il tempo di convergenza diverso tra la *policy* di recupero originale e quella ottima proposta è ben evidente. Vale la pena osservare come le modifiche proposte permettano di raggiungere più rapidamente la totale completezza della rete rispetto alle controparti originali.

7.3 Il confronto dei due protocolli

Si confrontano infine le metriche comuni ad i due protocolli, l'efficienza di trasmissione e l'andamento della percentuale di copertura di rete.

7.3.1 Il ritardo complessivo

Non è stato possibile confrontare la statistica del ritardo per le ragioni già esposte. Tuttavia se si confrontano i grafici in Figura 7.7 , in Figura 7.28 ed in Figura 7.29 è possibile ipotizzare una vittoria del protocollo proposto per $\lambda \geq 10$. La differenza tra il ritardo totale della Fase I e quello totale del protocollo è di 40 s e 20 s per $\lambda = 10$ e $\lambda = 20$. Considerato che una trasmissione della Fase II impiega in media ≈ 10 s si vede come bastino appena 4 o 2 sessioni nella Fase II per ottenere un ritardo peggiore rispetto a *OFBP*. Si riporta in figura tale confronto per completezza.

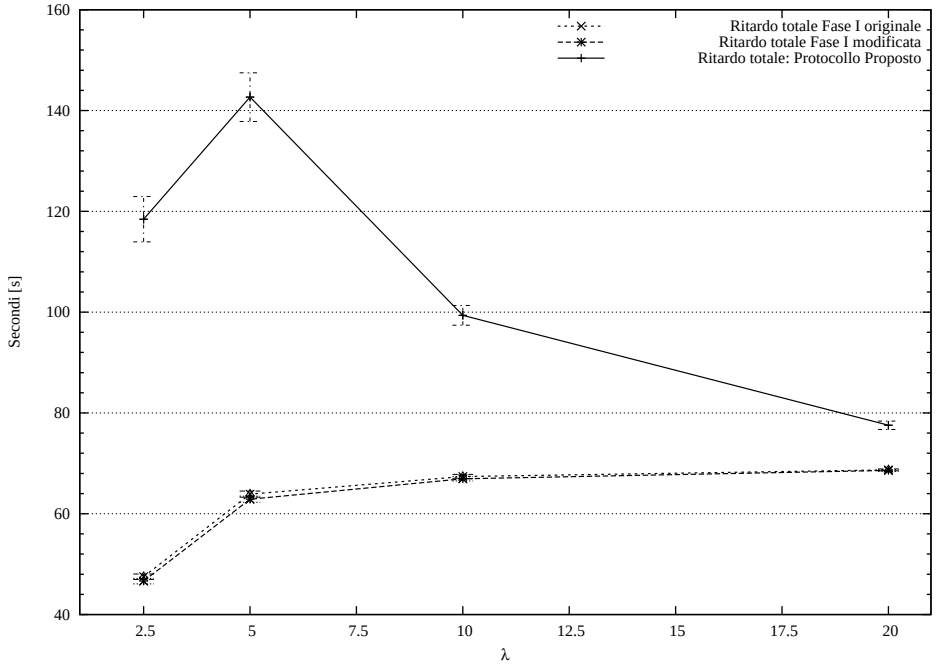

Figura 7.39: Il confronto tra i ritardi.

7.3.2 L'efficienza totale

L'obiettivo principale di un *broadcast* per reti di sensori deve essere l'efficienza di trasmissione, che si traduce ovviamente in efficienza energetica. La difficoltà d'accesso ai nodi rende questi due obiettivi essenziali per la sopravvivenza della rete stessa. Dal confronto appare evidente la netta supremazia del protocollo qui presentato in ogni situazione di rete. Si passa da $\lambda = 2.5$ dove il risparmio è pari ad $\approx 1/4$ a $\lambda \geq 5$ dove il risparmio energetico è circa della metà rispetto al protocollo *CRBCast*.

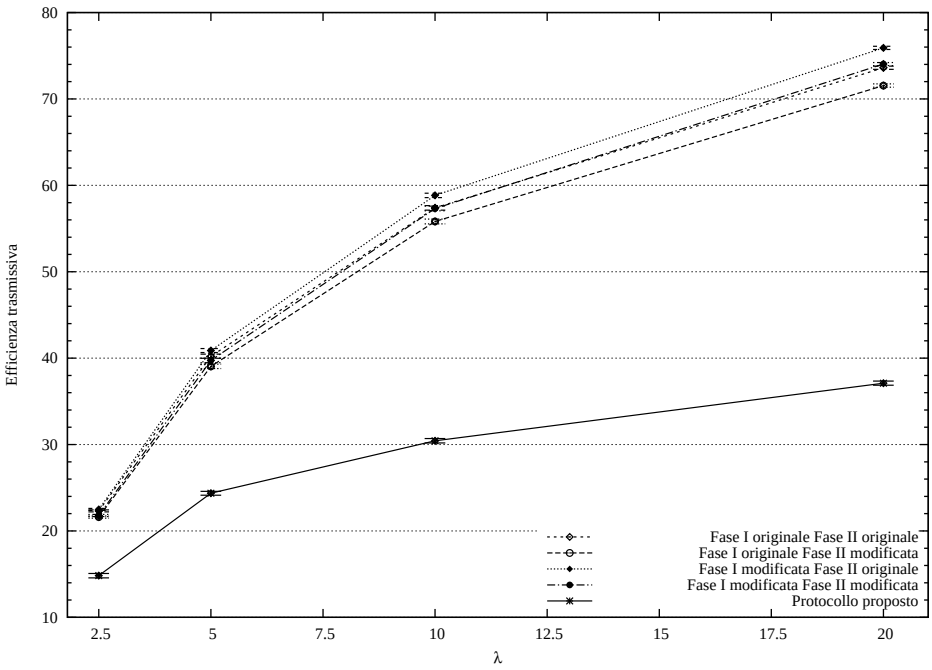

Figura 7.40: Confronto dell'efficienza di trasmissione.

7.3.3 La *CDF* della copertura di rete

Si procede al confronto finale tra le funzioni di distribuzione di probabilità della percentuale di rete coperta. Per una maggiore facilità di lettura sono state suddivise per valore di λ. Tale è confronto è fatto a parità di percentuale di rete completata all'inizio della Fase II.

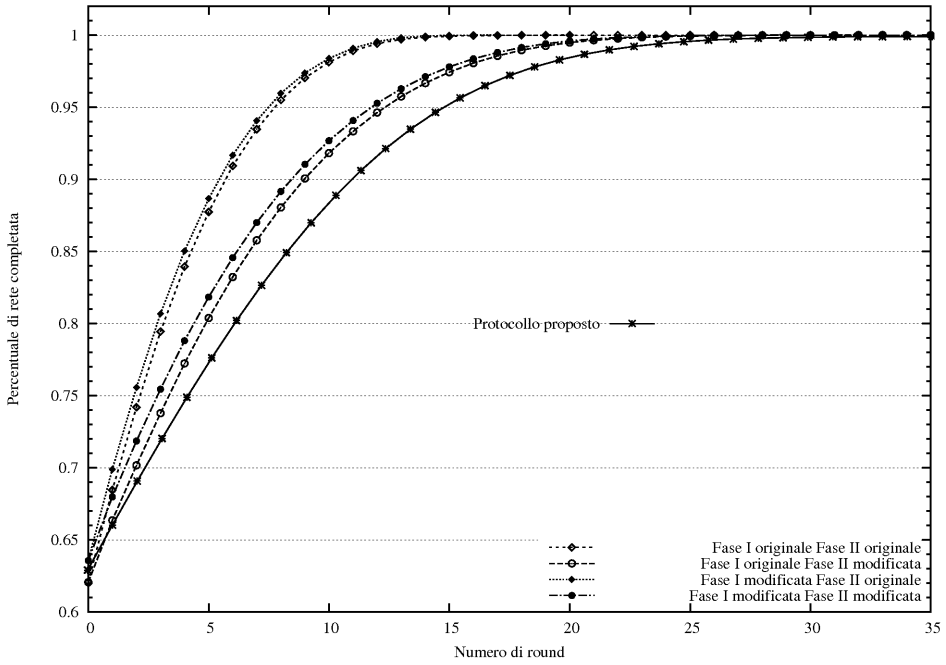

Figura 7.41: Il confronto delle *cdf* della percentuale di rete coperta, $\lambda = 2.5$.

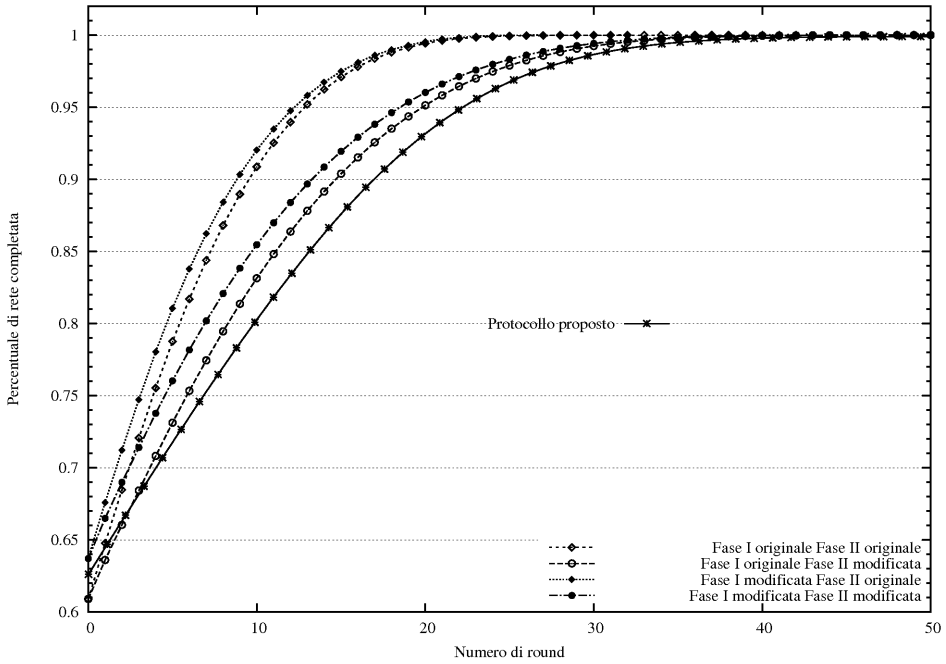

Figura 7.42: Il confronto delle *cdf* della percentuale di rete coperta, $\lambda = 2.5$.

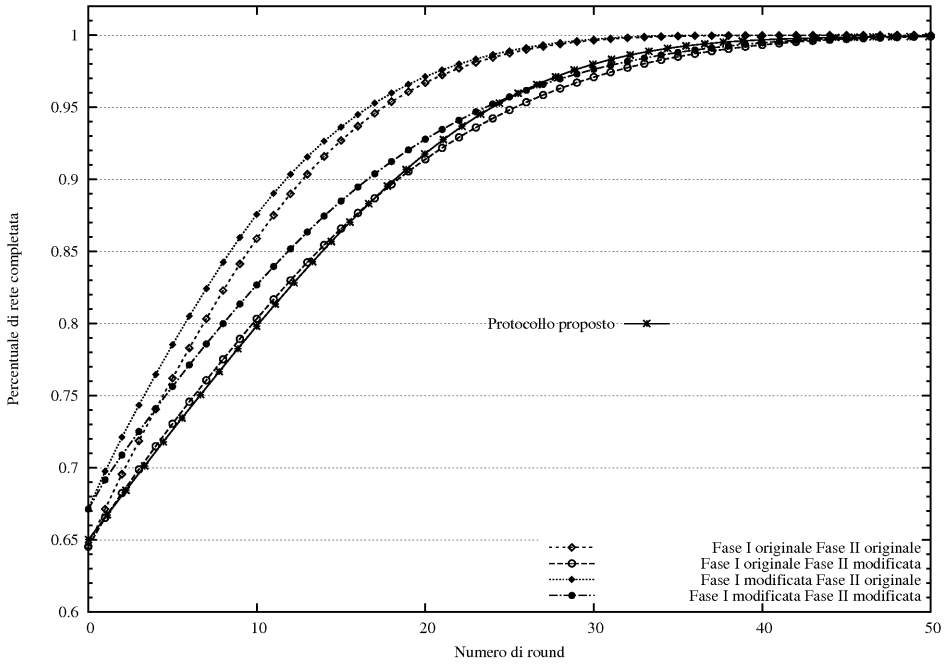

Figura 7.43: Il confronto delle *cdf* della percentuale di rete coperta, $\lambda = 2.5$.

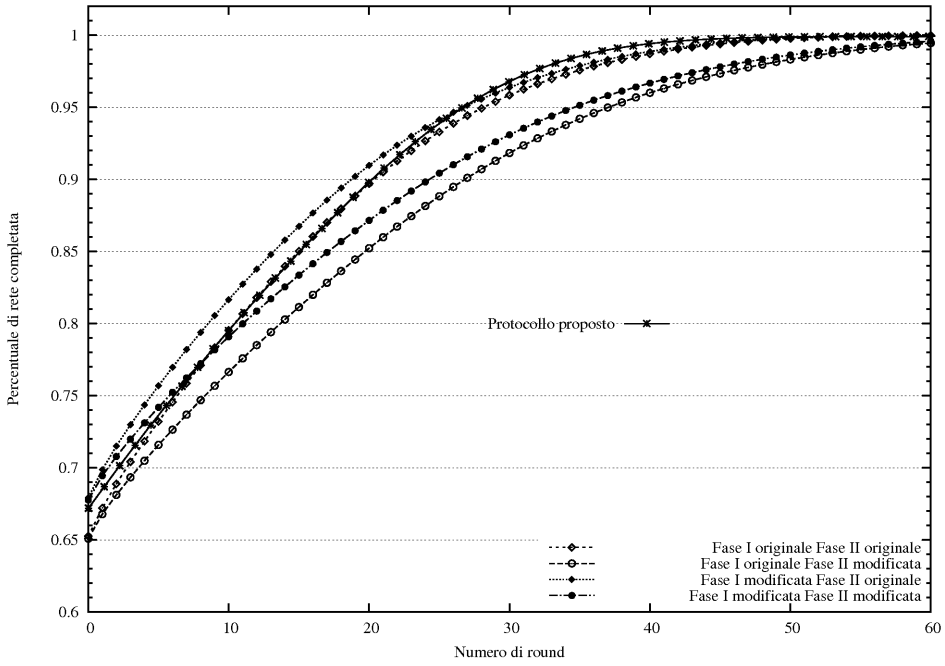

Figura 7.44: Il confronto delle *cdf* della percentuale di rete coperta, $\lambda = 2.5$.

Nelle topologie di rete sparse e moderatamente sparse ($\lambda = 2.5\ ,\ 5$), si nota come il numero di round richiesti per il completamento della rete del protocollo *OFBP* sia moderatamente maggiore anche delle configurazioni più lente del protocollo *CRBCast*; tuttavia non è possibile formulare alcuna ipotesi di ritardo in quanto i round trasmissivi del protocollo qui presentato sono molto spesso in parallelo, mentre il grado di parallelismo della Fase II non è stato possibile simularlo.
Per valori di $\lambda \geq 10$ il numero di round di trasmissione necessari ad *OFBP* tende a diminuire e per $\lambda = 20$ è addirittura il minimo.

Capitolo 8

Conclusione

In questa sede si è affrontato il problema del *broadcast* nelle reti acustiche sottomarine. Si è proposto un nuovo protocollo che si basa sui codici a fontana e se ne sono analizzate le prestazioni del protocollo stesso in scenari *single-hop* e *multi-hop*: se ne è dedotto che per densità di rete basse l'approssimazione tra le due è possibile, mentre per $\lambda \geq 7.5$ si commette un errore dell'ordine del 5%. Si è infine passati ad un confronto di *OFBP* con il protocollo già esistente *CRBCast* adattato al caso sottomarino: l'analisi ha prodotto un risultato nettamente favorevole al protocollo qui proposto, nei termini di ritardo ed efficienza energetica. Tale risultato incoraggia a proseguire lo sviluppo e il perfezionamento del protocollo stesso. *Si ricorda infatti che l'analisi proposta è un limite superiori alle prestazioni reali.* Il passo successivo è quello di rimuovere le ipotesi di idealità del canale di *feedback*, di osservarne le nuove prestazioni e di progettare l'integrazione dello stesso con uno strato *MAC* altrettanto efficiente ed affidabile.

Appendice A

Ulteriori risultati

A.1 I codici a fontana in ambiente *multi hop*

A.1.1 L'efficienza del protocollo proposto

L'efficienza di trasmissione

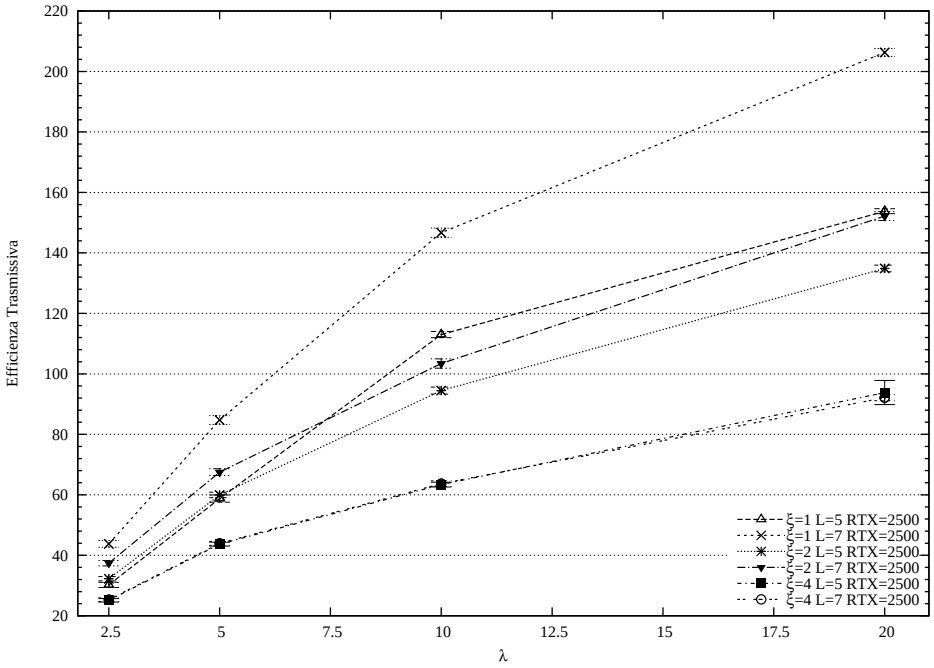

Figura A.1: L'efficienza di trasmissione per $d_{tx} = 2500\ m$.

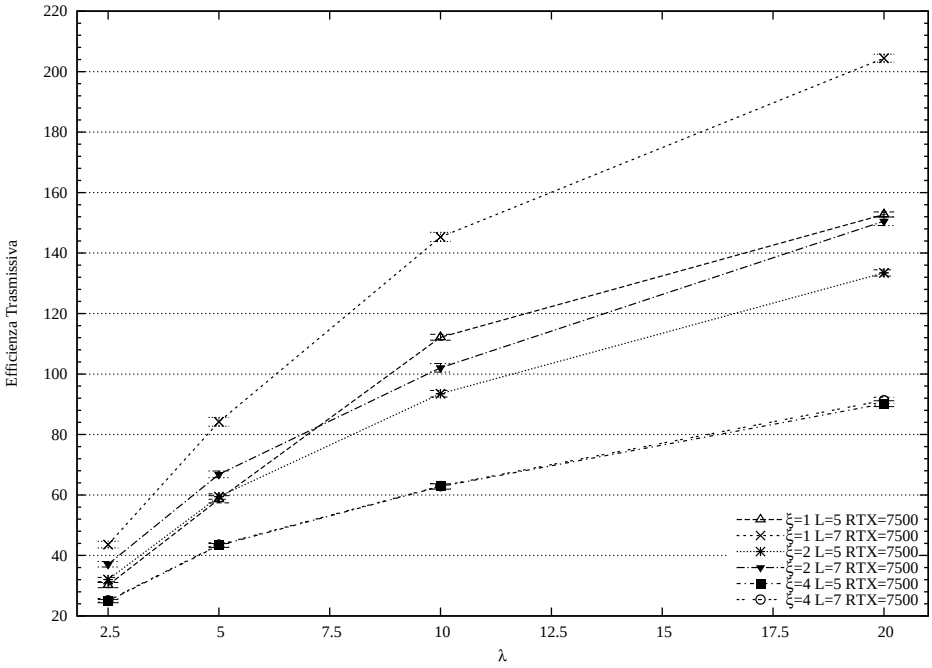

Figura A.2: L'efficienza di trasmissione per $d_{tx} = 7500\ m$.

Il ritardo medio di una sessione

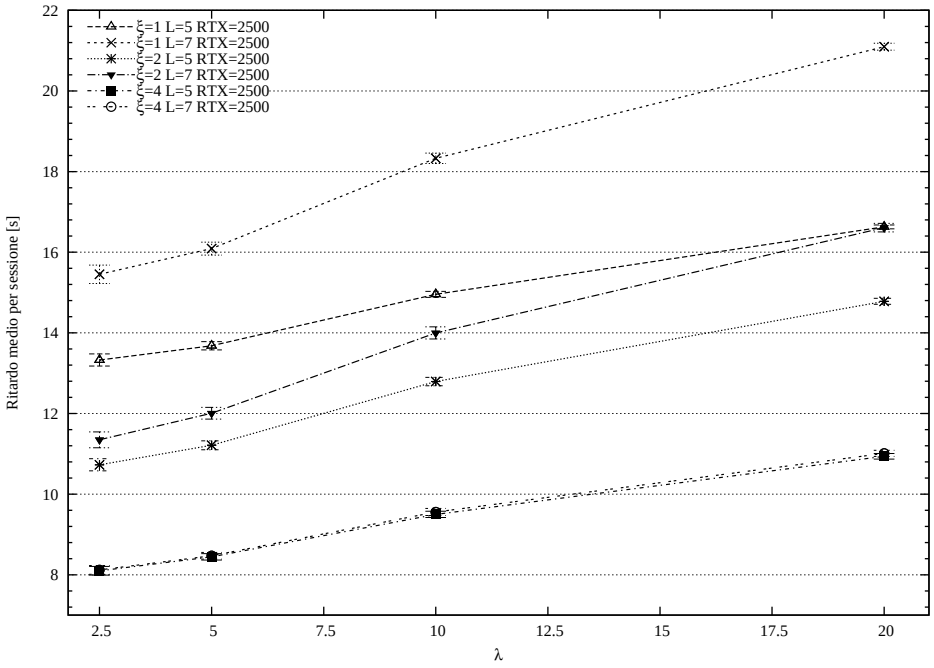

Figura A.3: Il ritardo medio di una sessione per $d_{tx} = 2500$.

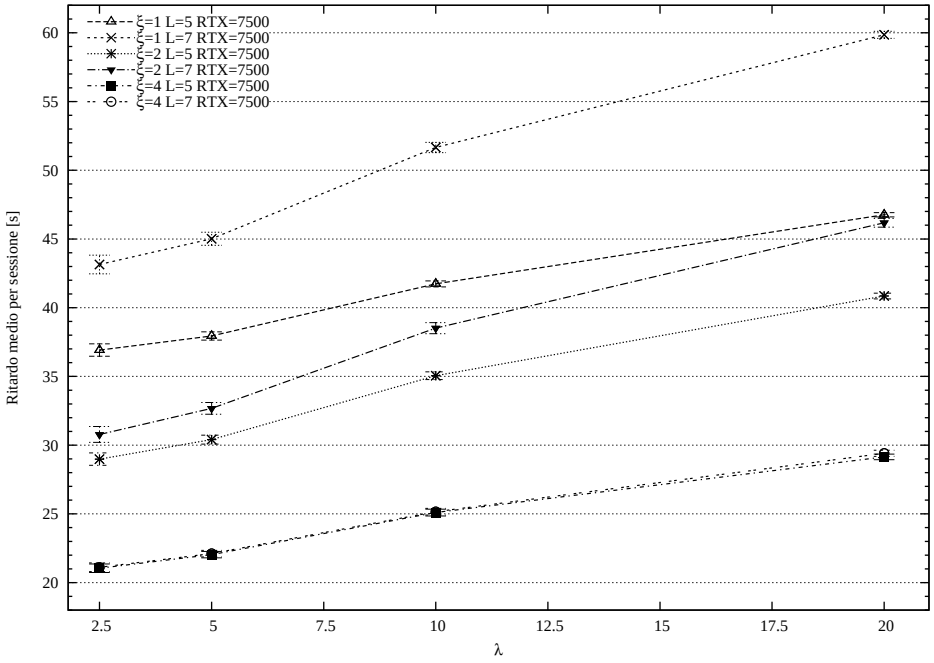

Figura A.4: Il ritardo medio di una sessione per $d_{tx} = 7500$.

Il numero medio di *hop*

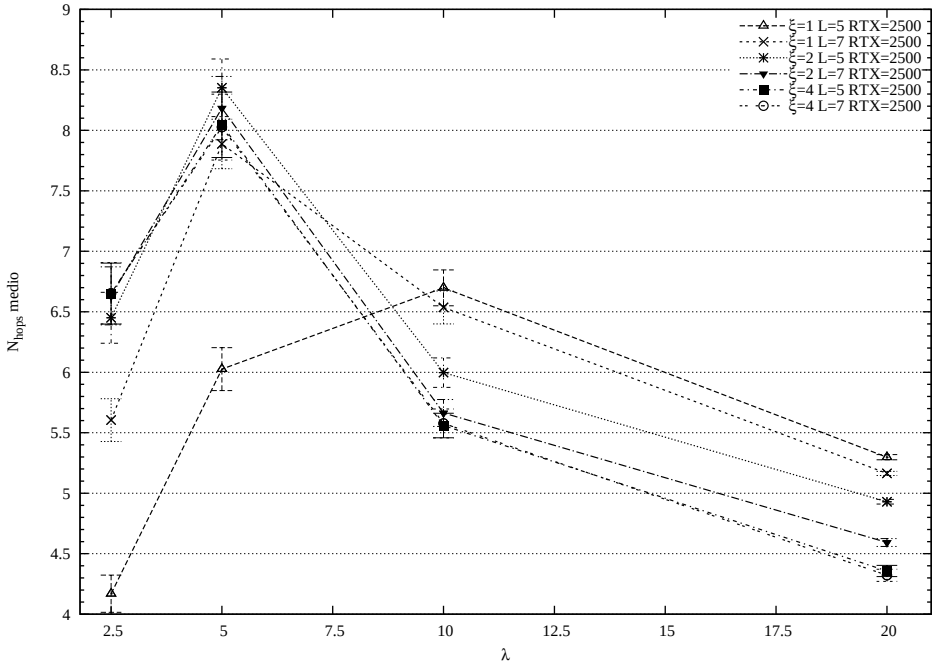

Figura A.5: Il numero medio di *hop* per $d_{tx} = 2500$.

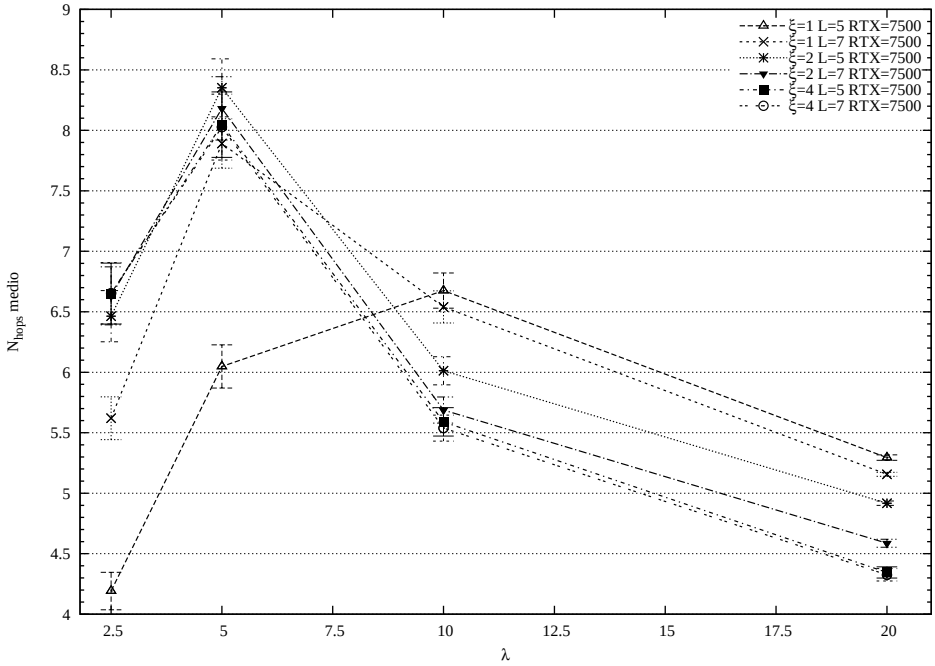

Figura A.6: Il numero medio di *hop* per $d_{tx} = 7500$.

Il ritardo medio di un blocco codificato

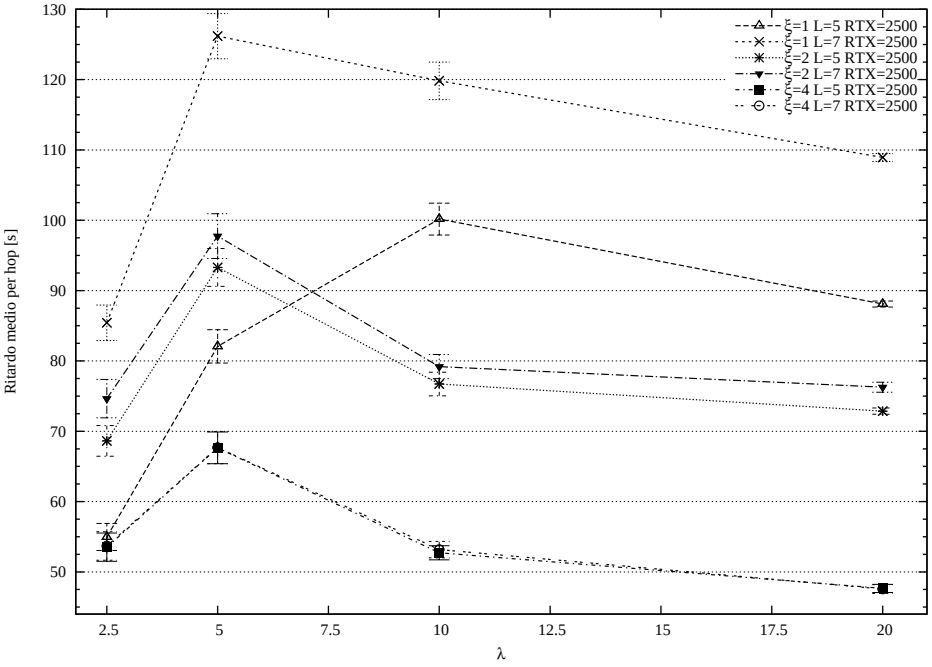

Figura A.7: Il ritardo medio di blocco per $d_{tx} = 2500$.

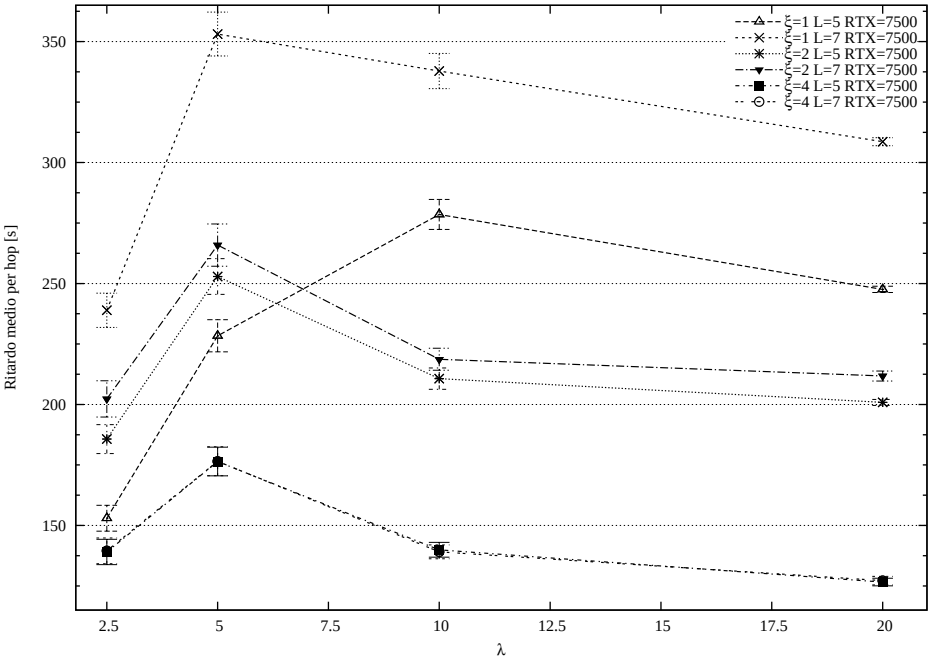

Figura A.8: Il ritardo medio di blocco per $d_{tx} = 7500$.

A.1.2 Il confronto tra *single hop* e *multi hop*

Il numero medio di pacchetti codificati trasmessi per sessione

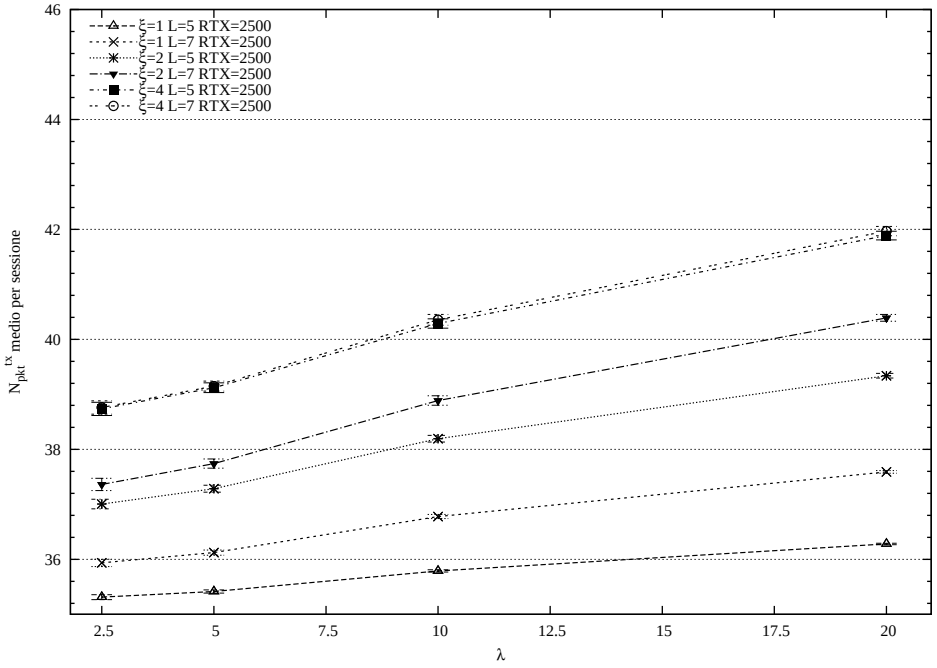

Figura A.9: Il numero medio di pacchetti codificati trasmessi per sessione, $d_{tx} = 2500$.

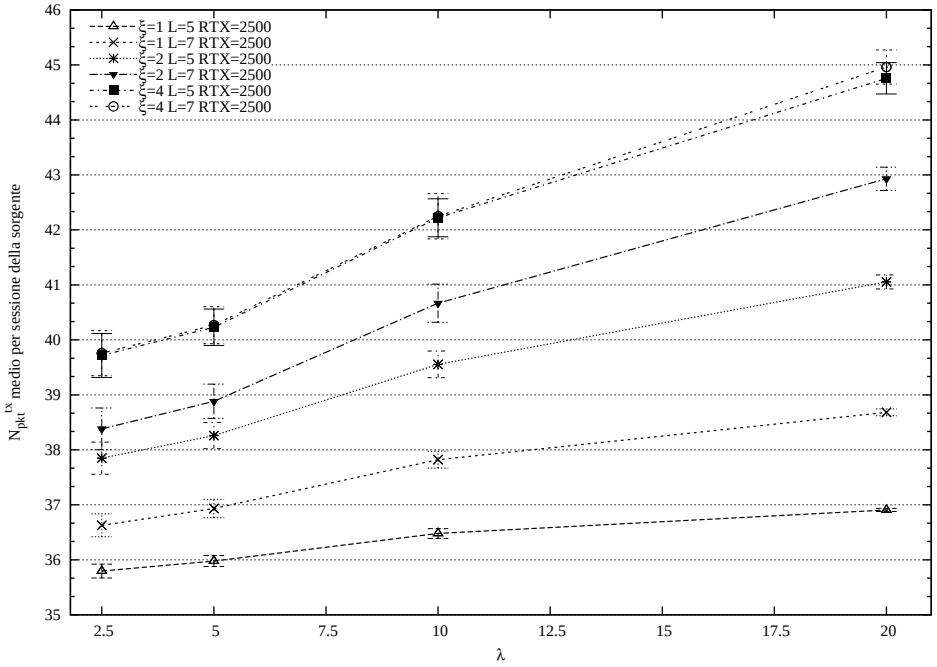

Figura A.10: Il numero medio di pacch. cod. trasmessi dalla sorg. per sessione, $d_{tx} = 2500$.

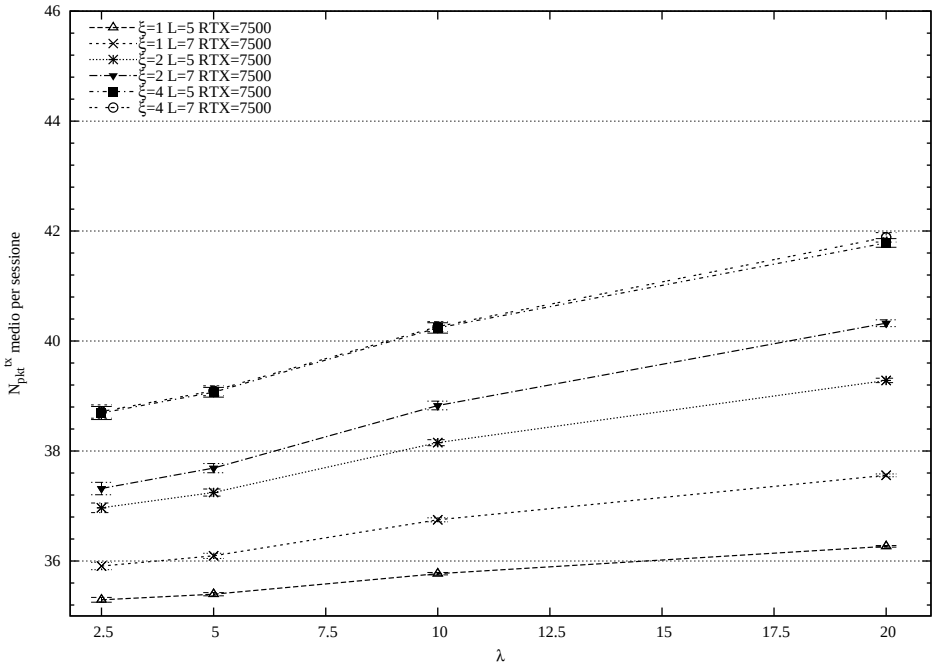

Figura A.11: Il numero medio di pacchetti codificati trasmessi per sessione, $d_{tx} = 7500$.

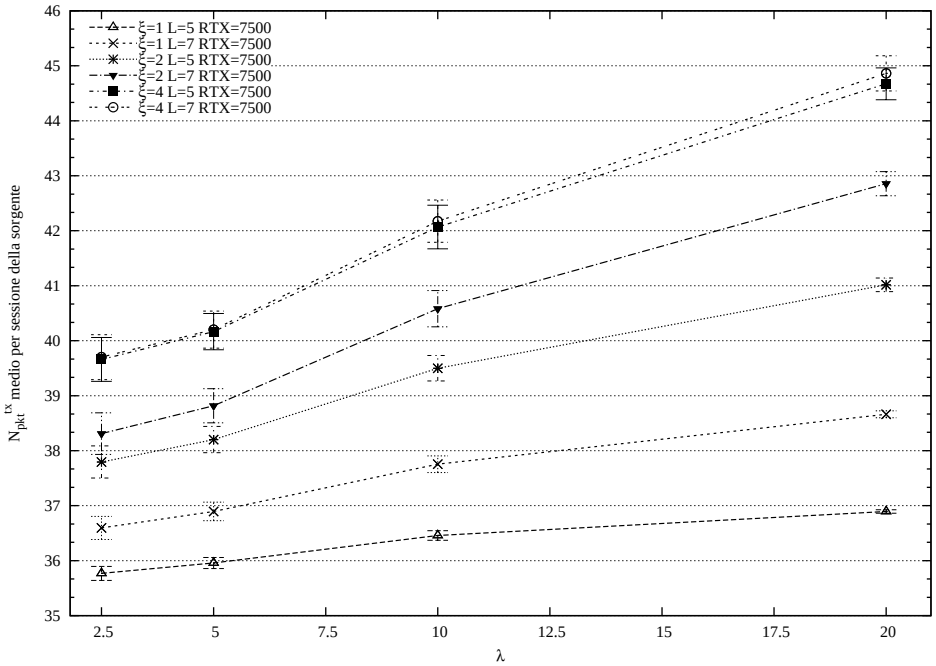

Figura A.12: Il numero medio di pacchetti codificati trasmessi dalla sorgente per sessione, $d_{tx} = 7500$

Il numero medio di round per sessione

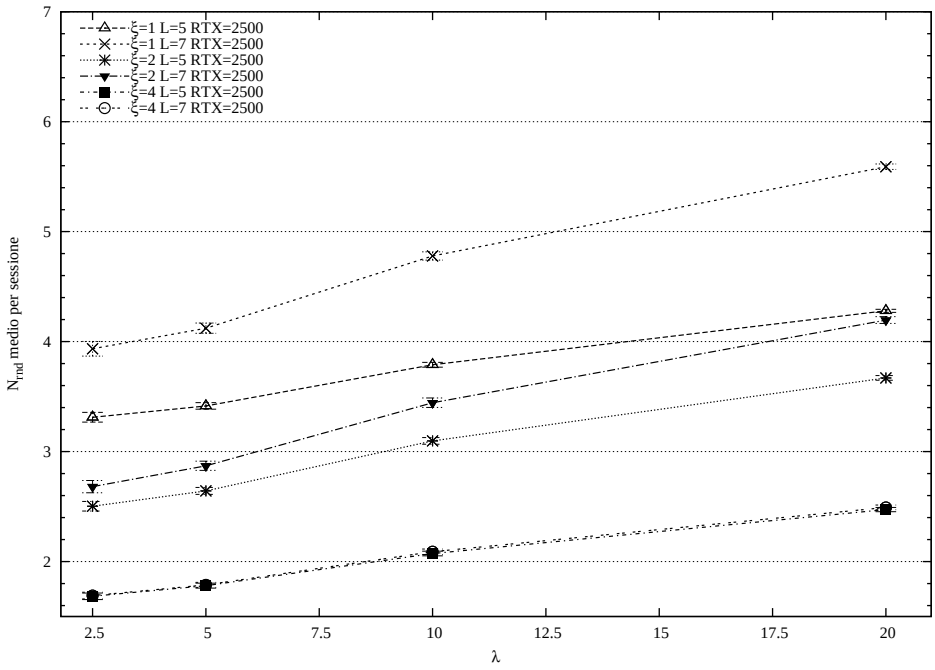

Figura A.13: Il numero medio di round per sessione $d_{tx} = 2500$.

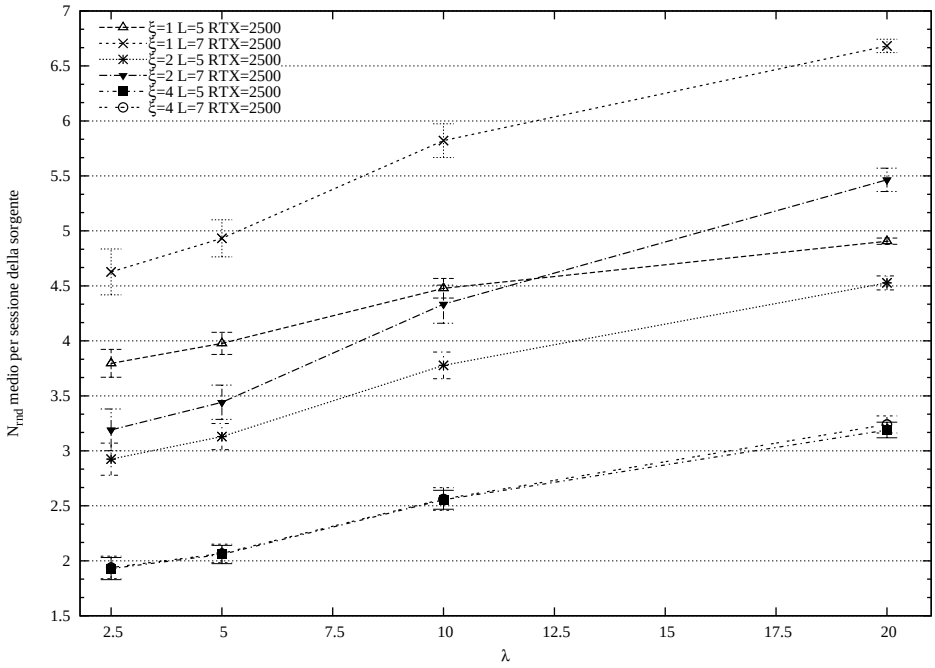

Figura A.14: Il numero medio di round per sessione di sorgente $d_{tx} = 2500$.

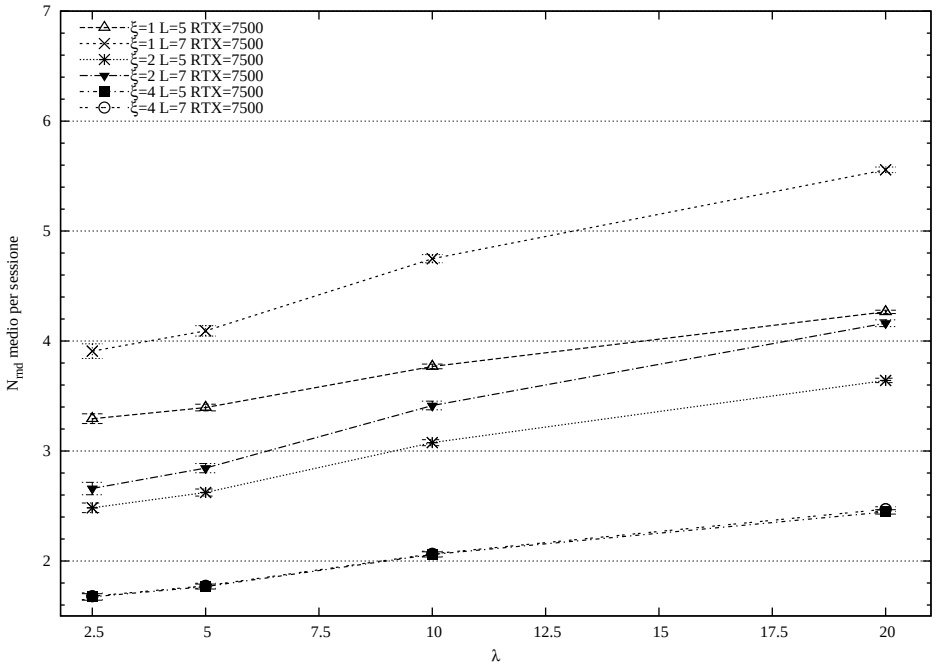

Figura A.15: Il numero medio di round per sessione $d_{tx} = 7500$.

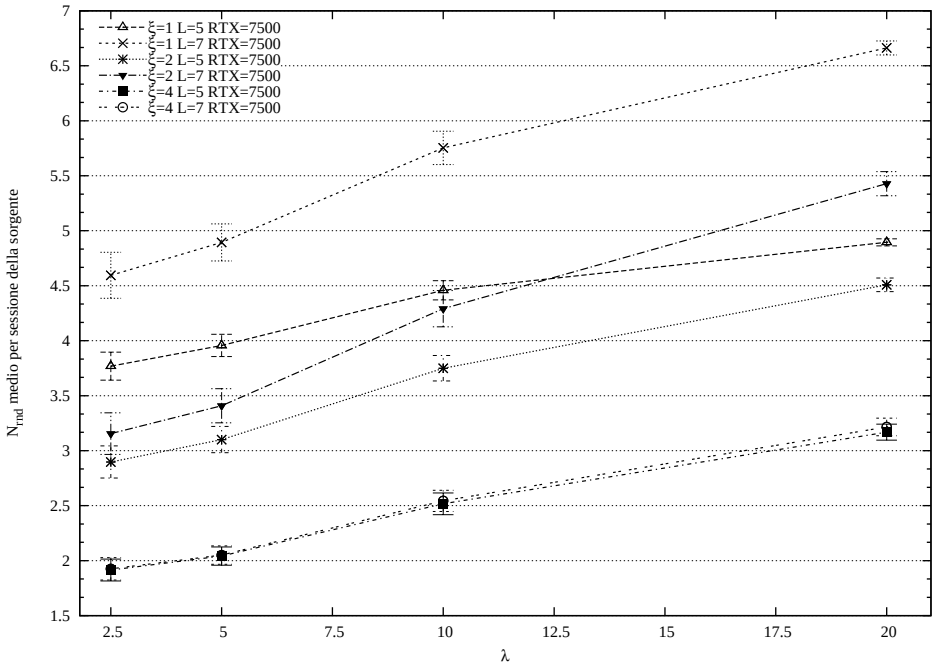

Figura A.16: Il numero medio di round per sessione di sorgente $d_{tx} = 7500$.

La probabilità media di fallire una sessione

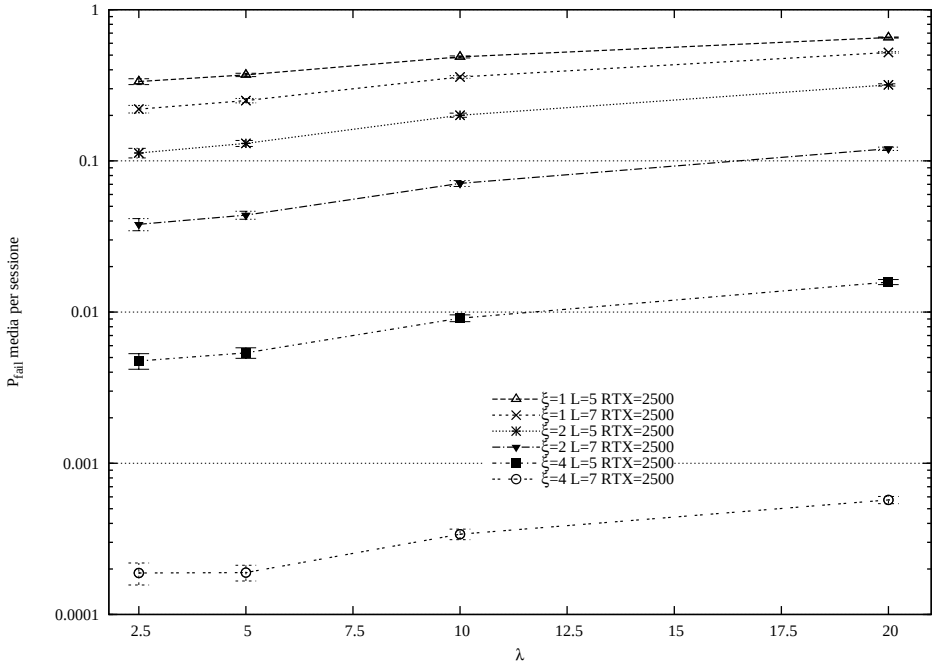

Figura A.17: La probabilità media di fallire una sessione, $d_{tx} = 2500\ m$.

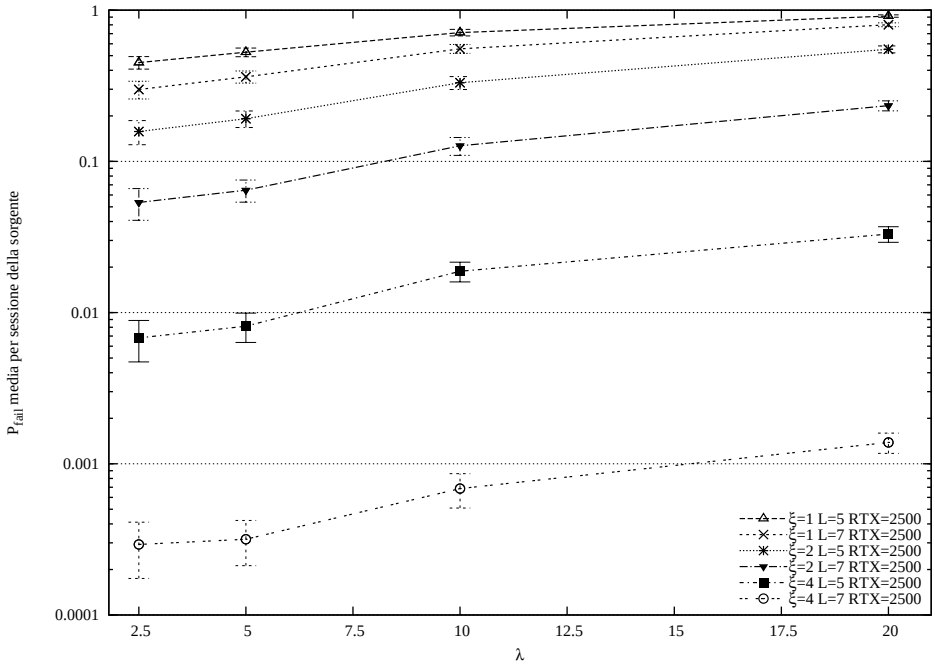

Figura A.18: La probabilità media di fallire una sessione della sorgente, $d_{tx} = 2500\ m$.

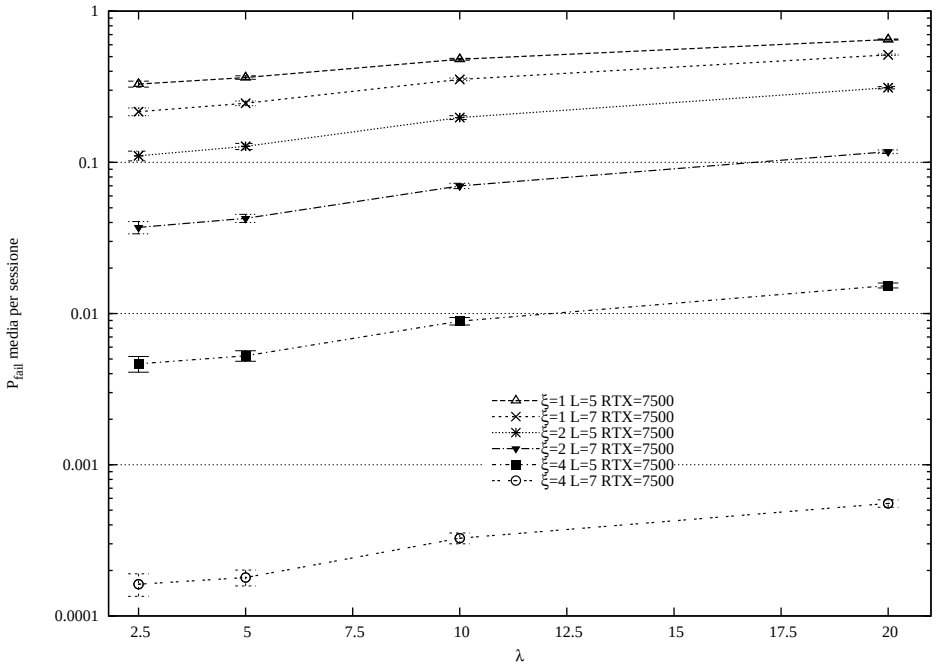

Figura A.19: La probabilità media di fallire una sessione, $d_{tx} = 7500\ m$.

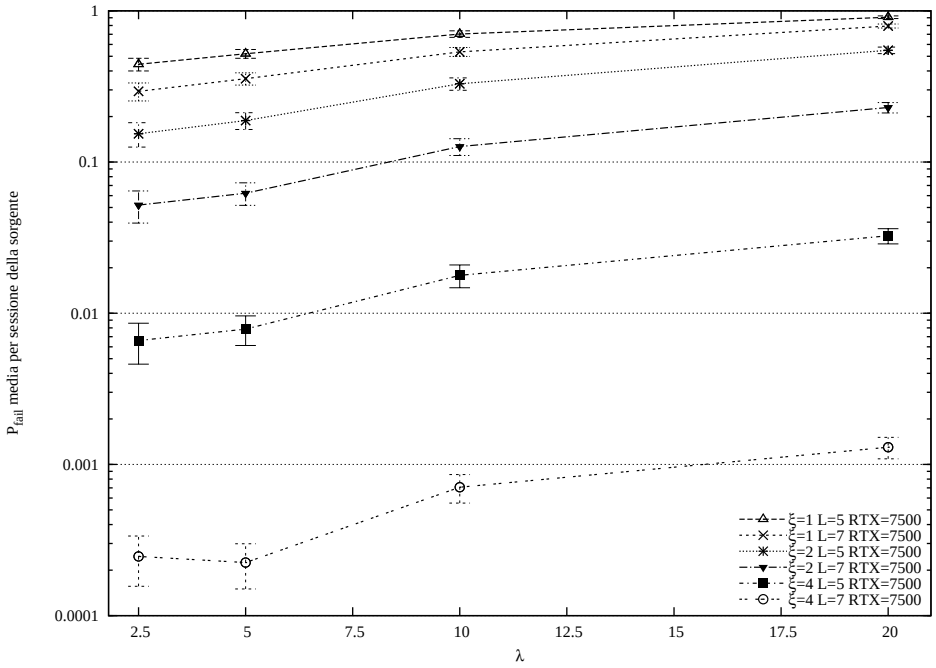

Figura A.20: La probabilità media di fallire una sessione della sorgente, $d_{tx} = 7500\ m$.

La percentuale media di vicini completi in una sessione

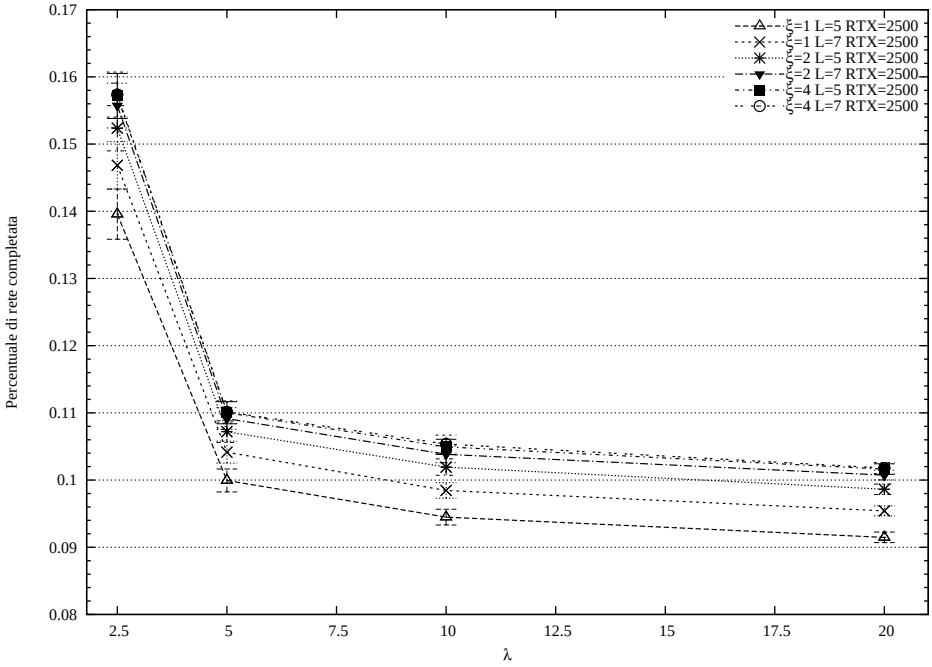

Figura A.21: La percentuale media di vicini completi dopo una sessione, $d_{tx} = 2500\ m$.

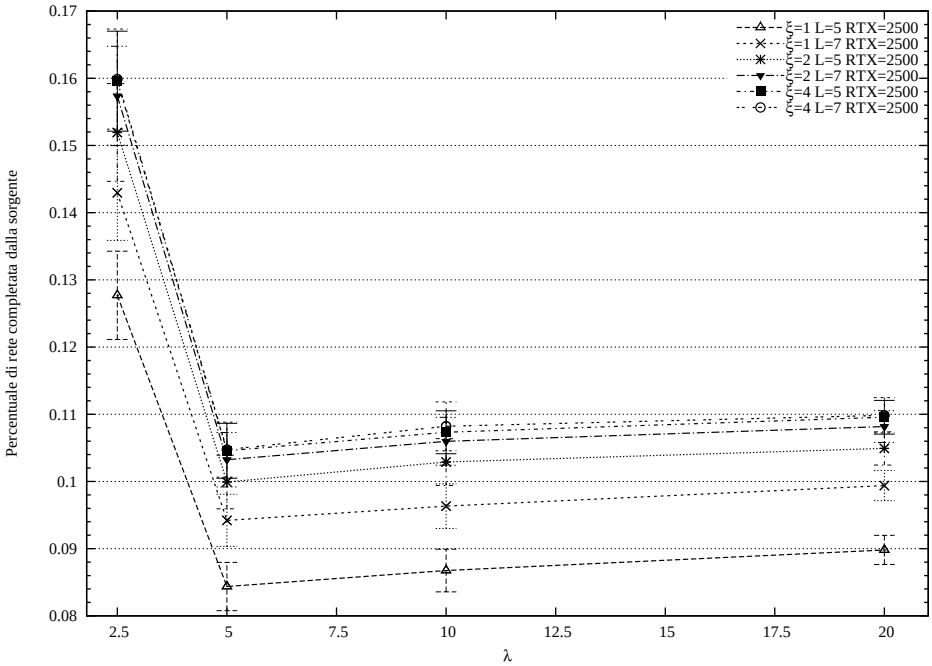

Figura A.22: La percentuale media di vicini completi della sorgente dopo una sessione, $d_{tx} = 2500\ m$.

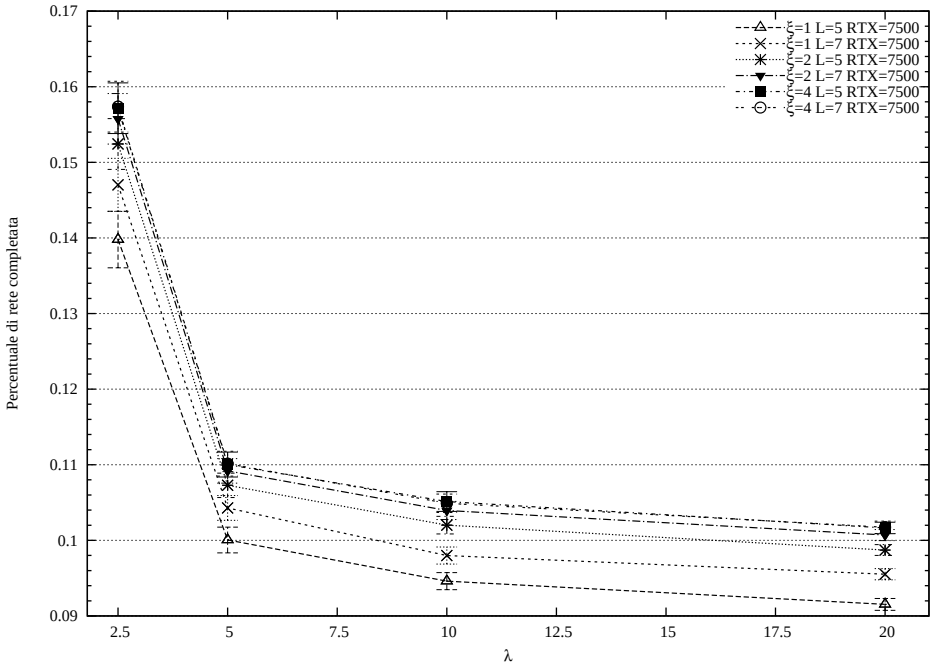

Figura A.23: La percentuale media di vicini completi dopo una sessione, $d_{tx} = 7500\ m$.

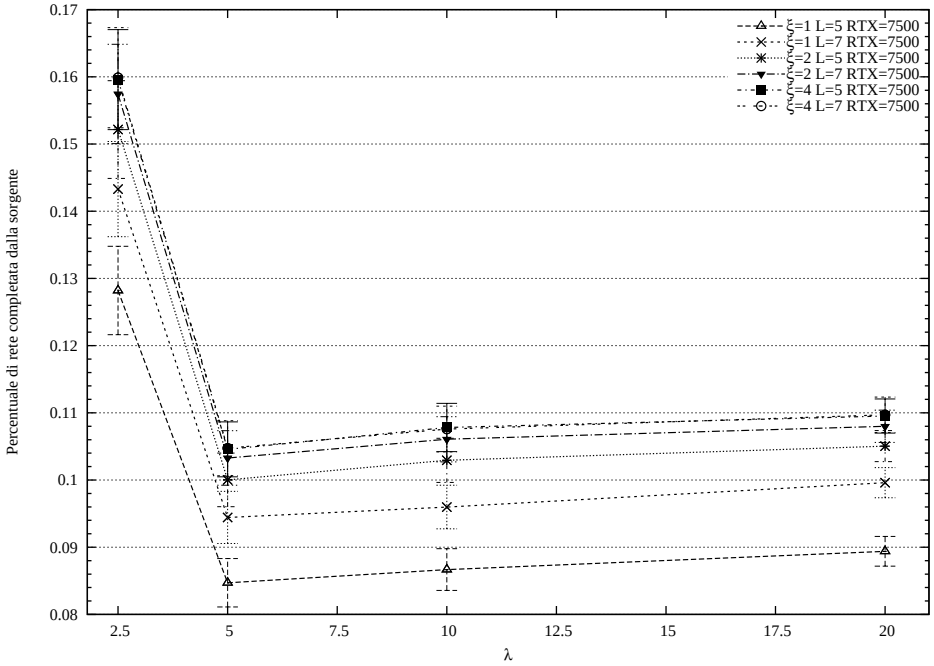

Figura A.24: La percentuale media di vicini completi della sorgente dopo una sessione, $d_{tx} = 7500\ m$.

L'avanzamento medio

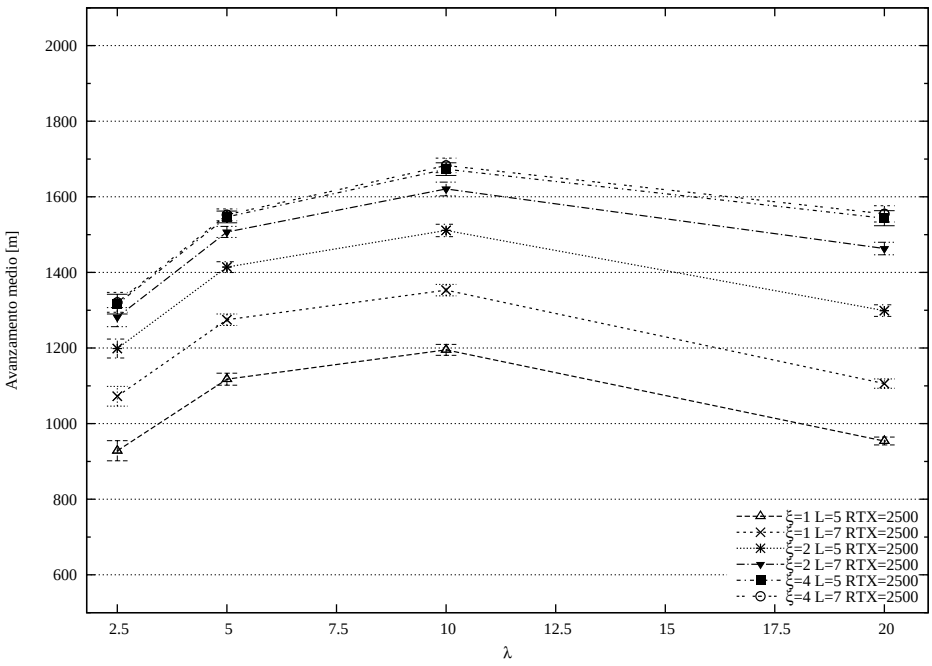

Figura A.25: L'avanzamento medio, $d_{tx} = 2500\ m$.

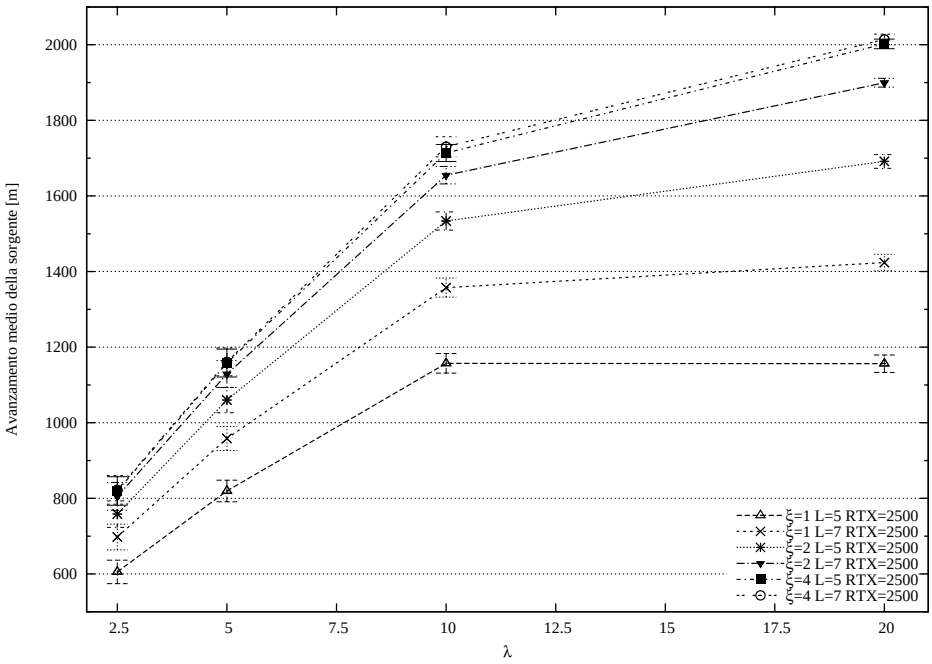

Figura A.26: L'avanzamento medio di sorgente, $d_{tx} = 2500\ m$.

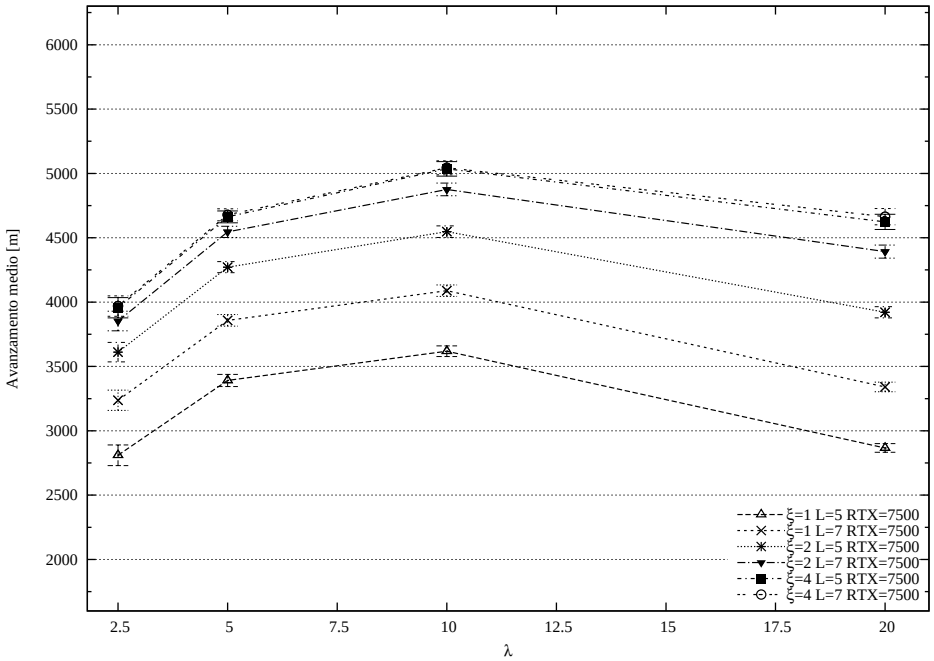

Figura A.27: L'avanzamento medio, $d_{tx} = 7500\ m$.

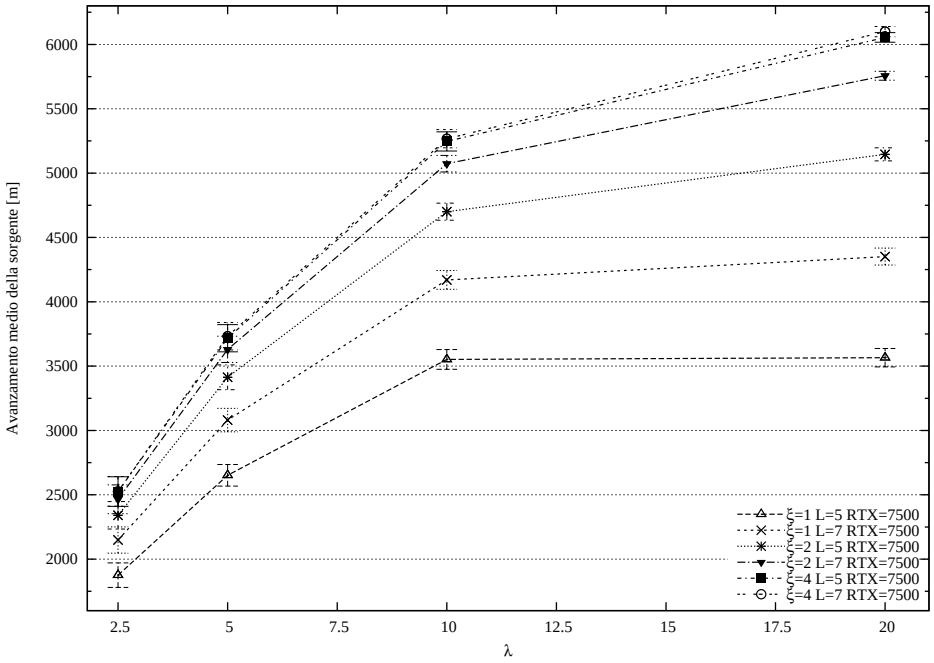

Figura A.28: L'avanzamento medio di sorgente, $d_{tx} = 7500\ m$.

Il raggio di completezza medio

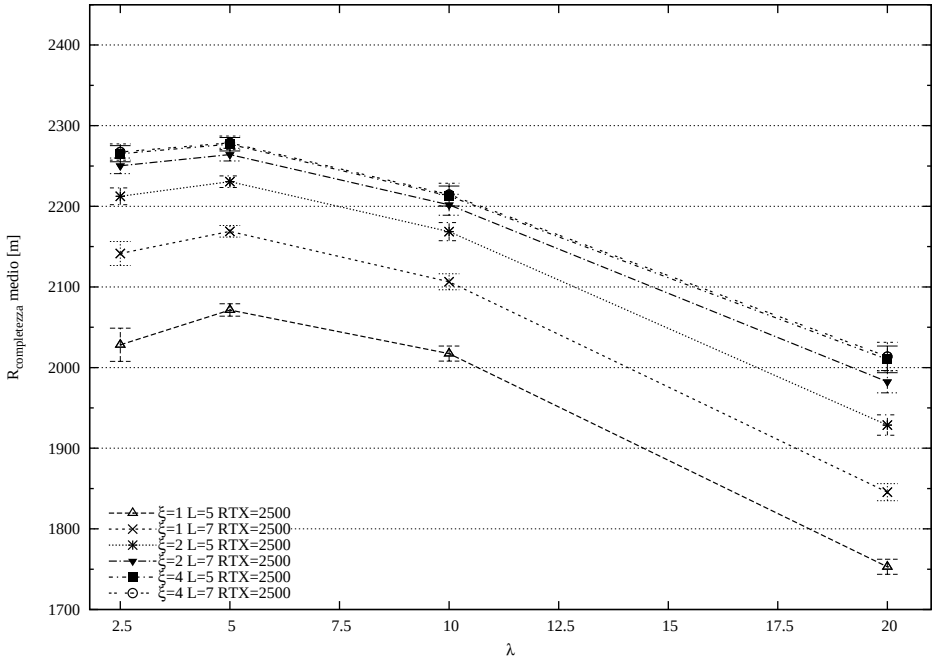

Figura A.29: Il raggio di completezza medio, $d_{tx} = 2500\ m$.

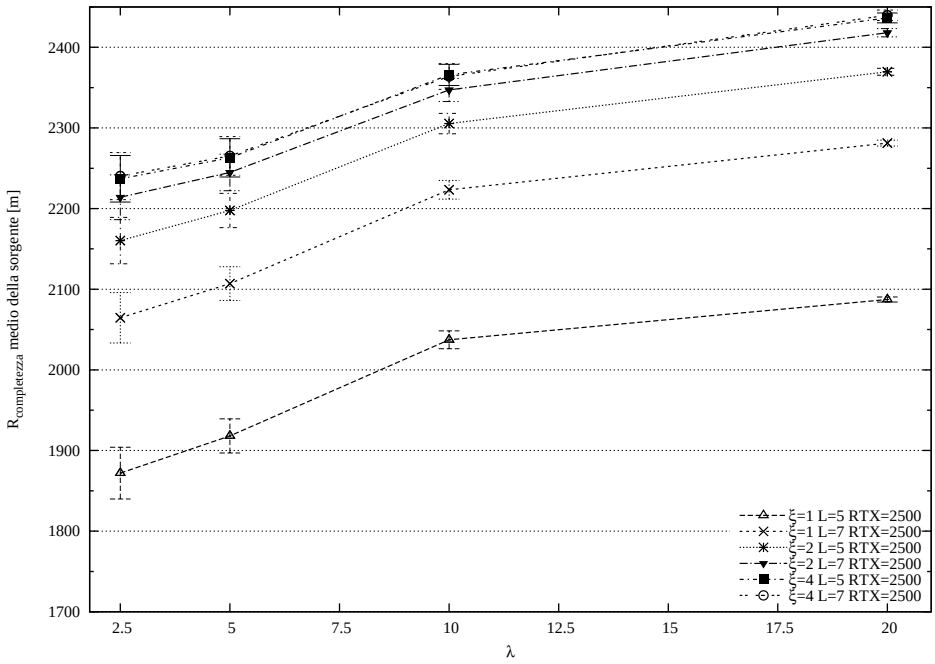

Figura A.30: Il raggio di completezza della sorgente medio, $d_{tx} = 2500\ m$.

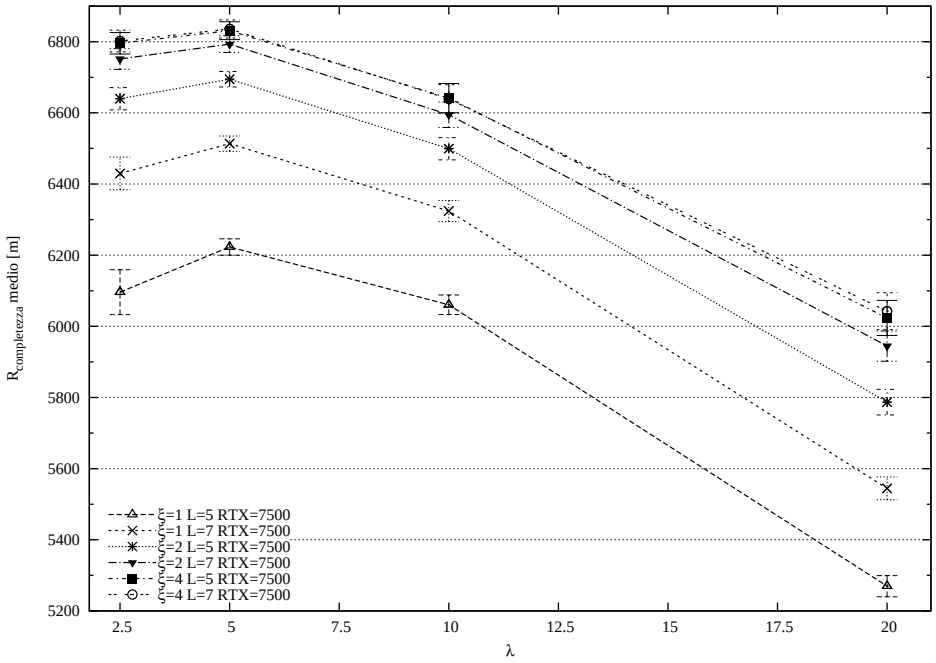

Figura A.31: Il raggio di completezza medio, $d_{tx} = 2500\ m$.

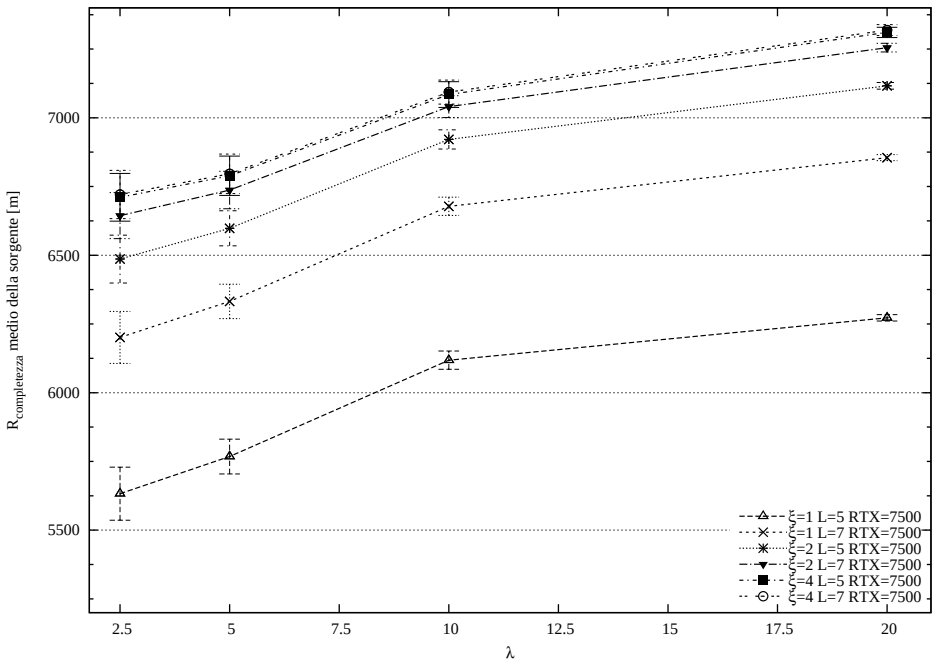

Figura A.32: Il raggio di completezza della sorgente medio, $d_{tx} = 2500\ m$.

Bibliografia

[1] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *Proceedings of ACM WUWNet*, Los Angeles, CA, Settembre 2006, pp. 41–47.

[2] M. Stojanovic, "Acoustic Underwater Comunications," *Encyclopedia of Telecommunications*, John Wiley and Sons, 2003.

[3] I. Akyildiz, D. Pompili, T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Ad Hoc Networks Journal*, Elsevier, Marzo 2005, vol. 3, Issue 3, pp. 257–279.

[4] I. Akyildiz, D. Pompili, T. Melodia, "State of the art in protocol research for underwater acoustic sensor networks," *Proceedings of ACM WUWNet 2006*, pp. 11–22.

[5] J. Partan, J. Kurose, J. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mob. Comput. Commun.*, Ottobre 2007, rev. 11, pp. 23–33.

[6] M. Stojanovic, "Recent Advances in High-Speed Underwater Acoustic Communications," IEEE J. Oceanic Eng. , Aprile 1996, pp. 125–136.

[7] A. F. Harris III, M. Zorzi, "Energy-efficient routing protocol design considerations for underwater networks," *Proceedings of IEEE SECON*, Giugno 2007.

[8] P. Casari, M. Rossi, M. Zorzi, "Towards Optimal Broadcasting Policies for HARQ based on Fountain Codes in Underwater Networks," *Fifth Annual Conference on Wireless on Demand Network Systems and Services 2008*, Gennaio 2008, pp. 11–19.

[9] P. Casari, S. Marella, M. Zorzi, "A comparison of multiple access techniques in clustered underwater acoustic networks," *Proceedings of IEEE/OES OCEANS*, Aberdeen, Scotland, Giugno 2007.

[10] P. Casari, M. Stojanovic, M. Zorzi , "Exploiting the Bandwidth-Distance Relationship in Underwater Acoustic Networks," *OCEANS 2007*, Settembre 2007 - Ottobre 2007, pp. 1–6.

[11] M. Luby, "LT codes," *The 43rd annual IEEE Symposium on proceedings of foundations of computer science*, 2002 , pp. 271-280.

[12] D.J.C. MacKay, "Fountain codes," *IEE Proceedings of Communications*, Dicembre 2005, vol. 152, n. 6, pp. 1062–1068.

[13] N. Rahnavard, F. Fekri, "CRBcast: a collaborative rateless scheme for reliable and energy-efficient broadcasting in wireless sensor networks," *Proceedings of the Fifth international Conference on information Processing in Sensor Networks*, April 2006, Nashville, Tennessee, USA, pp. 276–283.

[14] Rahnavard, Nazanin; Vellambi, N. Badri, Fekri, Faramarz, "FTS: A Fractional Transmission Scheme for Efficient Broadcasting via Rateless Coding in Multihop Wireless Networks," *IEEE MILCOM 2007*, Ottobre 2007, pp.1–7.

[15] B. Williams, T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," *Proceedings of the 3rd ACM international Symposium on Mobile Ad Hoc Networking*, Losanna, Svizzera , Giugno 2002, pp. 194–205.

[16] L. Hyojun, K. Chongkwon, "Multicast tree construction and flooding in wireless ad hoc networks," *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, Agosto 2000, Boston, Massachusetts, Stati Uniti d'America, pp. 61–68.

[17] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, M. Zorzi, "ns2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," *Proceedings of the 2nd international Conference on Performance Evaluation Methodologies and Tools*, Nantes, Francia, Ottobre 2007; *ACM International Conference Proceeding Series*, vol. 321. *ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering)*, ICST, Brussels, Belgio, pp. 1–8.

[18] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks," *Proceedings of IEEE INMIC*, Pakistan, Dicembre 2001, pp. 62–68.

[19] C. E. Perkins , P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *Proceedings of the conference on Communications architectures, protocols and applications*, Londra, United Kingdom, Agosto - Settembre 2002, pp. 234–244.

[20] C. Perkins , E. Belding-Royer , S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," *RFC Editor*, 2003.

[21] D. B. Johnson , D. A. Maltz , J. Broch, "DSR: the dynamic source routing protocol for multihop wireless ad hoc networks," *Ad hoc networking*, Addison-Wesley Longman Publishing Co. Inc., Boston, 2001.

[22] P. Bose , P. Morin , I. Stojmenović , J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, Novembre 2001, vol. 7 n. 6, pp. 609–616.

[23] T. Melodia, D. Pompili, and I. F. Akyildiz. "On the interdependence of Distributed Topology Control and Geographical Routing in Ad Hoc and Sensor Networks," *Journal of Selected Areas in Communications*, Marzo 2005, vol. 23 n. 3, pp. 520–532.

[24] P. Xie and J.-H. Cui, "SDRT: A Reliable Data Transport Protocol for Underwater Sensor Networks," *University of Connecticut Technical Report UbiNet-TR06*, Febbraio 2006.

[25] S.S Kulkarni, L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," *Distributed Computing Systems*, 2005; *ICDCS* 2005; *25th IEEE International Conference on Proceedings*, Giugno 2005, pp. 7–16.

[26] A. Durresi, V.K. Paruchuri, S.S. Iyengar, R. Kannan, "Optimized broadcast protocol for sensor networks," *IEEE Transactions on Computers*, Aug. 2005 vol. 54, n. 8, pp. 1013–1024.

[27] A. Durresi, V. Paruchuri, "Broadcast Protocol for Energy-Constrained Networks," *IEEE Transactions on Broadcasting*, Marzo 2007, vol. 53, n. 1, pp.112–119.

[28] F. Dai, J. Wu, "Performance analysis of broadcast protocols in ad hoc networks based on self-pruning," *IEEE Transactions on Parallel and Distributed Systems*, Novembre 2004, vol. 15, n. 11, pp. 1027–1040.

Ringraziamenti

Un grazie al prof. Zorzi, per la disponibilità e per la cortesia dimostratami in questi anni.
Ringrazio Paolo Casari, per l'infinita pazienza che ha dimostrato nei miei confronti e per l'aiuto illimitato prestatomi in questi interminabili mesi passati in laboratorio.
Ringrazio il mio Amore Silvia, per essere stata al mio fianco in questi anni e specialmente in questi mesi intensi e delicati. Senza di te, meglio, senza di *Noi*, non avrei mai trovato la forza, l'ispirazione e la voglia di portare a termine questa lunga esperienza che oggi giunge al termine. La tenacia e l'impegno che ho messo in questa tesi saranno rivolti a noi ed alla costruzione del *nostro* futuro.
Ringrazio Claudia, Cosimo e Ilaria, Laura, per essermi stato vicino per non aver mai dubitato di me e per avermi supportato (e sopportato) in questi anni di Università.
Ringrazio i miei amici tutti, per avermi regalato in tutto questo tempo spensieratezza, gioia ed un'amicizia vera e sincera, cosa rara e preziosa di questi tempi.

A voi tutti: non cambiate mai.