UNIVERSITÀ degli STUDI di CATANIA

DIPARTIMENTO DI INGEGNERIA ELETTRICA ELETTRONICA E INFORMATICA

Corso di Laurea Magistrale in Ingegneria Informatica

ELABORATO OMNeT++ no.1

Progetto sviluppato nell'ambito del Corso di

Reti per l'Automazione Industriale

Alberto Attilio Brincat	O55000293
Paolo Walter Modica	O55000258
Salvatore Quattropani	O55000294
Andrea Giuseppe Viola	O55000291

ANNO ACCADEMICO 2015-2016

Indice generale

INTRODUZIONE....3

Requisiti richiesti....3

Obiettivi di progetto....3

IMPLEMENTAZIONE DEL MODULO FISICO....4

Il modulo fisico - SubGHzPhy....4

SubGHzPhy – metodi principali....6

Il ricetrasmettitore - SubGHzRadioModel....7

SubGHzRadioModel – metodi principali....7

IMPLEMENTAZIONE DEL GENERATORE DI TRAFFICO....9

Il modulo applicativo – PeriodicAppLayer....9

PeriodicAppLayer – metodi principali....9

CONFIGURAZIONE DEGLI SCENARI DI SIMULAZIONE....12

Definizione del nodo di rete....12

Le reti....14

Rete 1....14

Rete 2....16

CONCLUSIONE....26

Introduzione

Il progetto illustrato nella presente relazione tratta dell'ideazione e dello sviluppo di un modello di simulazione per transceiver wireless che operano nella banda di frequenze a 915 MHz, e basati sul modello di simulazione del modulo fisico del protocollo IEEE 802.15.4 presente nel framework INETMANET di OMNeT++.

Requisiti richiesti

I requisiti richiesti dal committente del progetto per il modulo fisico del modello di simulazione da sviluppare sono:

- Banda di frequenza sfruttata: 915 MHz;
- Datarate configurabile, impostato di default a 250 kbps;
- Dimensione della Header e del CRC della frame a livello fisico configurabile, impostato di default a 9 byte;
- Possibilità di simulare la trasmissione su N canali di trasmissione, con N configurabile.

Obiettivi di progetto

Nell'elaborazione del presente progetto si sono preposti i seguenti obiettivi:

- Costruzione di un nuovo modulo basato sul modulo fisico di IEEE 802.15.4, opportunamente modificato per rispondere alle esigenze di progetto espresse dalla committenza;
- Costruzione di un nuovo modulo applicativo, che funga da generatore di traffico;
- Creazione di uno scenario di simulazione per validare il corretto funzionamento del modello in termini di datarate, comunicazione su più canali, packet loss ratio e rilevazione di eventuali collisioni.
- Raccolta di dati statistici relativi al funzionamento delle reti utilizzate negli scenari di simulazione di cui sopra.

Implementazione del modulo fisico

Lo sviluppo ed implementazione dell'elaborato è stata effettuata utilizzando l'ambiente di simulazione software OMNeT++.

Il primo passo nella realizzazione ha richiesto la creazione di un nuovo progetto, denominato “SubGHz”, a cui si è impostato il riferimento al framework INETMANET 2.0 per renderlo pienamente compatibile con i moduli d'utilità presenti in quest'ultimo.

Il modulo fisico - SubGHzPhy

Successivamente si è proceduto alla creazione di un nuovo Simple Module, **SubGHzPhy**, relativo al modulo fisico del modello da implementare.

Questo lavoro è stato svolto dal gruppo composto da Paolo Walter Modica e Salvatore Quattropani.

Tale modulo prevede la creazione di 3 file:

- SubGHzPhy.ned, file descrittore di rete, nel quale sono presentati i parametri caratteristici del livello fisico di un nodo di rete wireless utilizzante il protocollo IEEE 802.15.4, nonché le connessioni al livello superiore e i segnali usati per la rilevazione dei dati e raccolta delle statistiche sul funzionamento del nodo;
- SubGHzPhy.cc, file C++ nel quale è contenuta la definizione del modulo fisico e dei suoi metodi funzionali;
- SubGHzPhy.h, file header che contiene la dichiarazione dei metodi funzionali utilizzati nel file precedente, nonché le inclusioni di eventuali librerie di utilità contenute nel framework INETMANET.

Al fine di soddisfare i requisiti specificati dal committente del progetto, sono state apportate le seguenti modifiche al codice costituente il modulo SubGHzPhy, differenziandolo così dal modulo di riferimento Iee802154Phy:

- dichiarazione del parametro **datarate**, che specifica la frequenza di trasmissione del nodo wireless, espressa in kbps, ed è usato per l'impostazione della velocità di trasmissione del radiotrasmettitore. **datarate** verrà successivamente reimpostato nella configurazione dello scenario di simulazione per testare il funzionamento del modello.
- Dichiarazione del parametro **collisionNum**, utilizzato per il conteggio delle collisioni, rilevate dal modulo fisico, nel caso in cui avvengano trasmissioni da parte di 2 o più nodi.
- Dichiarazione del parametro **hl** (header length), che specifica la dimensione dell'header della Frame dati trasmessa dai nodi. Questo parametro, che viene utilizzato nel modulo SubGHzRadioModel che verrà descritto in seguito, verrà successivamente reimpostato nella configurazione dello scenario di simulazione per testare il funzionamento del modello.
- Dichiarazione del parametro **radioMod**, che specifica il tipo di ricetrasmettitore usato dal modulo SubGHzPhy. Di default tale parametro è impostato al valore **SubGHzRadioModel.**
- Dichiarazione dell'oggetto **radioMod**, di tipo **SubGHzRadioModel.**
- Dichiarazione dei **segnali** e dei rispettivi vettori di **statistiche**, utilizzati per catturare i valori relativi alle grandezze da misurare per valutare le prestazioni del modello.

```
package subghz;

simple SubGHzPhy
{
    parameters:
        //int HeaderLength =default(9);
        bool debug;
        double channelNumber = default(0);
        bool NoBitError = default(true);
        double datarate @unit("bps") = default(250kbps);        //come da progetto - Elaborato
        int collisionNum=default(0);                            //utilizzato per conteggiare i
        int hl=default(9);                                      //utilizzato per impostare l'h
        string radioMod @enum("SubGHzRadioModel") = default("SubGHzRadioModel");
```

Dichiarazione dei parametri di SubGHzPhy

SubGHzPhy – metodi principali

Al fine di soddisfare i requisiti specificati dal committente del progetto, è stato necessario apportare le seguenti modifiche ai principali metodi del modulo fisico SubGHzPhy, derivanti direttamente dal modulo Ieee802154Phy. Tali modifiche riguardano:

- **initialize:** funzione richiamata per l'inizializzazione del modulo fisico in fase di simulazione. Le modifiche qui apportate sono relative all'impostazione e all'utilizzo del radioModel, in questo caso di tipo SubGHzRadioModel.

```
//da qui

std::string rModel =  par("radioMod").stdstringValue();
if (rModel=="")
    rModel = "SubGHzRadioModel";

radioMod= (SubGHzRadioModel *) createOne(rModel.c_str());
radioMod->initializeFrom(this);

radioMod = createRadioModel();
radioMod->initializeFrom(this);
EV<<"Impostiamo la Frame Header a "<<hl<<" bit"<<endl;
radioMod->setHeaderLength(hl);

//a qui, dichiarazione del SubGHzRadioModel e impostazione della headerLength
```

Utilizzo del SubGHzRadioModel su SubGHzPhy

- **handleLowerMsgEnd**: funzione che si occupa della gestione della fase finale della ricezione delle frame. Le modifiche qui apportate sono relative alla gestione del lossrate e alla raccolta dei dati statistici al riguardo, ed in particolare hanno permesso di considerare come pacchetti persi anche le frame in ingresso affette da eccessivo rumore, e non solo quelle vittime di collisione come previsto prima della modifica stessa.

```
//da qui
numGivenUp++;                    //pacchetti persi
lossRate = (double)numGivenUp/((double)numReceivedCorrectly+(double)numGivenUp);
emit(lossRateSignal, lossRate);
//a qui, aggiunto per gestire la ricezione di una airframe troppo disturbata - c
```

Modifica al metodo handleLowerMsgEnd

Il ricetrasmettitore - SubGHzRadioModel

Successivamente si è proceduto alla creazione di una nuova classe, **SubGHzPhyRadioModel**, che descrive le caratteristiche e le funzionalità del ricetrasmettitore Wi-Fi utilizzato.

SubGHzRadioModel estende la classe **IRadioModel** presente nel framework INETMANET, e le sue funzioni principali riguardano il controllo sulla correttezza delle frame ricevute, per mezzo del calcolo del CRC, il calcolo della BER rilevata, la rilevazione di collisioni sul mezzo wireless ed il calcolo della durata del tempo necessario alla trasmissione di una frame.

Questa classe prevede la creazione di 2 file:

- SubGHzRadioModel.cc, file C++ nel quale è contenuta la definizione dei metodi funzionali della classe e le relative variabili usate;
- SubGHzRadioModel.h, file header nel quale sono contenute le dichiarazione dei metodi funzionali utilizzati nel file precedente, nonché le inclusioni di eventuali librerie di utilità contenute nel framework INETMANET.

SubGHzRadioModel – metodi principali

Al fine di soddisfare i requisiti specificati dal committente del progetto, è stato necessario apportare le seguenti modifiche ai principali metodi della classe SubGHzRadioModel. In particolare, tali modifiche riguardano:

- l'inserimento del metodo **setHeaderLength**, necessario per impostare la dimensione della header delle frame al valore scelto in fase di configurazione dello scenario di simulazione;
- la sostituzione, nel metodo **calculateDuration**, del valore costante def_phyHeaderLength con il parametro **HeaderLength** presente nella classe.

```
Register_Class(SubGHzRadioModel);

static const double BER_LOWER_BOUND = 1e-10;

void SubGHzRadioModel::initializeFrom(cModule *radioModule)
{
    // read from Ieee802154phy

    EV<<"Entrata su initializeFrom"<<endl;
    EV<<"Header= "<<HeaderLength<<endl;
    snirThreshold = dB2fraction(radioModule->par("snirThreshold").doubleValue());
    ownerRadioModule = radioModule;
}

//funzione per l'impostazione della headerLength - come da progetto - Elaborato 1

void SubGHzRadioModel::setHeaderLength(int hl)
{
    HeaderLength = hl;
    EV<<"Entrata su setHeaderLength"<<endl;
    EV<<"Nuova HeaderLength= "<<HeaderLength<<endl;
}

double SubGHzRadioModel::calculateDuration(AirFrame *airframe)
{
    EV<<"Entrata su calculateDuration"<<endl;
    return ((this->HeaderLength)*8 + airframe->getBitLength())/airframe->getBitrate() ; //
}
```

Definizione dei metodi di SubGHzRadioModel

Implementazione del generatore di traffico

Lo sviluppo dell'elaborato progettuale è proseguito con la creazione del generatore di traffico periodico, modulo del livello applicativo cui è demandato il compito di generare periodicamente dei pacchetti dati, di dimensione scelta in fase di configurazione dello scenario di simulazione, al fine di testare il corretto funzionamento del modulo fisico implementato precedentemente.

Questo lavoro è stato svolto dal gruppo composto da Alberto Attilio Brincat e Andrea Giuseppe Viola.

Il modulo applicativo – PeriodicAppLayer

L'implementazione del generatore di traffico periodico prevede la creazione di 3 file:

- periodicAppLayer.ned, file descrittore di rete, nel quale sono presentati i parametri caratteristici del livello applicativo che realizza il generatore di traffico, nonché le connessioni al livello inferiore e i segnali usati per la rilevazione dei dati e raccolta delle statistiche sul funzionamento del nodo;
- periodicAppLayer.cc, file C++ nel quale è contenuta la definizione del modulo applicativo e dei suoi metodi funzionali;
- periodicAppLayer.h, file header che contiene la dichiarazione dei metodi funzionali utilizzati nel file precedente, nonché le inclusioni di eventuali librerie di utilità contenute nel framework INETMANET.

PeriodicAppLayer – metodi principali

Per l'implementazione del modulo applicativo sono state utilizzate delle funzioni primitive per la comunicazione con il modulo fisico definite nel modello del protocollo LLDN (standard IEEE 802.15.4e), fornito dal committente del presente progetto.

Al fine di soddisfare i requisiti specificati precedentemente, il modulo relativo al generatore di traffico prevede tra i suoi parametri caratteristici il valore booleano **isTransmitter**, che permette di distinguere i nodi della rete predisposta per lo scenario di simulazione in *trasmittenti* (generatori e trasmettitori di traffico nella rete) e *riceventi* (ricevitori del traffico dati).

Fatta eccezione per le primitive di comunicazione con il livello fisico, i principali metodi che implementano le funzionalità del modulo applicativo sono:

- **initialize**, funzione richiamata per l'inizializzazione del modulo in fase di simulazione. Permette di definire il comportamento del livello applicativo a seconda che il relativo nodo sia configurato come trasmettitore o ricevitore.
- **transmitDataFrame**, metodo preposto per la creazione e l'invio di un nuovo pacchetto dati al modulo del livello fisico (che si occuperà successivamente di trasmetterlo nella rete).
- **handleMessage**, metodo preposto alla gestione dei messaggi ricevuti dal modulo applicativo del nodo. I messaggi ricevuti vengono gestiti in maniera diversa secondo il tipo dello stesso, in particolare:
    - i self message vengono gestiti dal metodo handleSelfMessage;
    - i pacchetti dati vengono decapsulati e ne viene conteggiata la dimensione del payload, che viene poi utilizzata per calcolare il *throughput* di rete.
- **handleSelfMessage**, metodo preposto per la gestione dei self message. L'implementazione realizzata per il livello applicativo prevede che questi possano essere trasmessi solo dai nodi della rete configurati come trasmettitori. I self message da gestire sono due:
    - *StartTimer*, timer d'inizializzazione per i nodi della rete;
    - *TxTimer*, timer che regola la finestra temporale entro il quale i nodi hanno la possibilità di trasmettere frame dati.

```
void PeriodicAppLayer::initialize() {
    lowerLayerIn = gate("lowerLayerIn");
    lowerLayerOut = gate("lowerLayerOut");
    maxChannel = par("maxChannel");
    isTransmitter= par("isTransmitter").boolValue();

    period = par("period").doubleValue();
    startTime = par("startTime").doubleValue();

    totbit = 0;
    //distinzione del comportamento dei nodi- come da progetto - Elaborato 1
    if(isTransmitter){
        setTxMode();
        cMessage *tim = new cMessage("StartTimer");
        scheduleAt(period, tim);
    }else{
        setRxMode();
    }
}

//metodo per la creazione e l'invio di un pacchetto dati - come da progetto - Elaborato 1

void PeriodicAppLayer::transmitDataFrame() {
        cPacket *pkt = new cPacket("pkt");
        pkt->setBitLength(1024);
        send(pkt, lowerLayerOut);
}
```

Metodi initialize e transmitDataFrame

```
void PeriodicAppLayer::handleMessage(cMessage *msg) {
    if(msg->isSelfMessage()) {
      handleSelfMessage(msg);
        return;

    }

    if(!msg->isPacket() && msg->getArrivalGate() == lowerLayerIn) {
        handlePhyPrimitive(msg);
        return;
    }
    //gestione dei pacchetti dati in ingresso - come da progetto - Elaborato 1

    if(msg->isPacket() && msg->getArrivalGate() == lowerLayerIn && !isTransmitter){
        cPacket *pkt_rcv = dynamic_cast<cPacket *>(msg);
        totbit += pkt_rcv->getBitLength();
        if(pkt_rcv->getKind()==PACKETOK){
            EV <<"Nodo trasmette: " <<totbit <<" bit"<<endl;
            simsignal_t sig = registerSignal("Throughput");
            double thr = (double)(totbit/simTime().dbl());
            EV <<"Nodo cronometra: " <<simTime().dbl() <<" secondi"<<endl;
            EV <<"Nodo misura: " <<thr <<" bit/s"<<endl;
            emit(sig, thr);
            sig = registerSignal("Latency");
            emit(sig, simTime().dbl() - pkt_rcv->getSendingTime());
        }
        delete pkt_rcv;
    }
}
```

Metodo handleMessage

Configurazione degli scenari di simulazione

Ultimato lo sviluppo e l'implementazione dei moduli costituenti il modello del nodo transceiver wireless, si procede nella definizione e configurazione di opportune reti, costituite da nodi del suddetto tipo, il cui funzionamento in uno specifico scenario di simulazione servirà a validare il corretto comportamento del modello costruito.

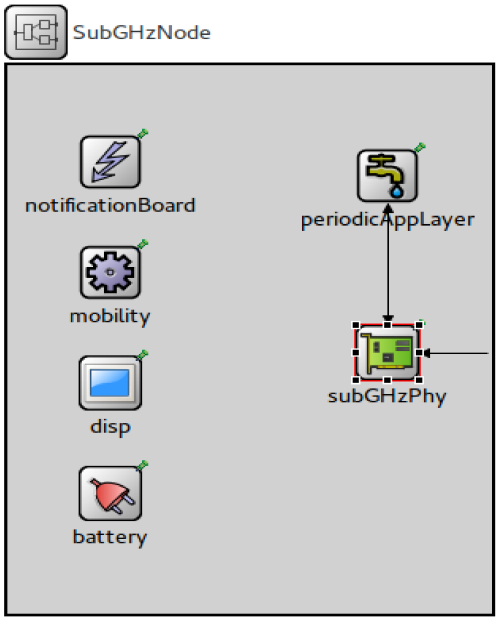
Definizione del nodo di rete

Il nodo transceiver da utilizzare nelle reti degli scenari di simulazione è stato definito e configurato a partire dal modello Ieee802154Node, presente all'interno del framework INETMANET.

Il lavoro di definizione del nodo della rete è stato svolto dal gruppo composto da Alberto Attilio Brincat e Andrea Giuseppe Viola.

Il modello di nodo così ottenuto, denominato **SubGHzNode**, è costituito dai seguenti moduli:

- **periodicAppLayer**, generatore periodico di traffico al livello applicativo;
- **SubGHzPhy**, modulo del livello fisico sviluppato sulla base delle specifiche richieste dal committente;
- **battery**, **display**, **mobility** e **notificationBoard**, moduli provenienti da Ieee802154Node e fondamentali per il corretto funzionamento del modello del nodo.



Il nodo SubGHzNode

Le reti

Per l'esecuzione delle simulazioni necessarie per testare il corretto funzionamento del modello creato, sono state predisposte le seguenti configurazioni di rete.

Rete 1

La prima rete è costituita da due nodi, un nodo trasmittente (**transmitter**) ed un nodo ricevente (**receiver**), interagenti in uno scenario predisposto per simulare il modello di propagazione a *spazio libero*. Il lavoro di definizione del suddetto scenario è stato svolto dal gruppo composto da Paolo Walter Modica e Salvatore Quattropani.

In fase di configurazione, i parametri dei nodi della rete sono stati così inizializzati:

- **Datarate** = 250 kbps;
- **Frequenza della portante** = 915 MHz;
- **Dimensione Header** = 9 byte;
- **Dimensione Frame** = 80 byte;
- **Canale n. 9**.

Il presente scenario è stato sottoposto a differenti run di simulazione, al fine di determinare:

- il Throughput della rete al variare della distanza tra il nodo trasmittente ed il nodo ricevente;
- il Throughput della rete ed il suo andamento rispetto al Workload della stessa, al variare del periodo di trasmissione delle frame.

Le simulazioni effettuate al fine di determinare l'andamento del Throughput al variare della distanza dei nodi, inizializzata al valore di 100 metri, hanno messo in evidenza che, nel modello di propagazione a spazio libero, questo non varia all'aumentare della distanza ed entro i 170 m, soglia oltre la quale la potenza del segnale trasmesso diventa troppo esigua per essere percepita dal ricevente come significativa, e che determina quindi la perdita totale delle frame trasmesse.

Il throughput della rete, ed il suo andamento rispetto al workload al variare del periodo di trasmissione della frame, sono quindi stati determinati nei successivi run di simulazione tenendo i due nodi della rete ad una distanza pari a 100 m.

Il risultato ottenuto dalle suddette prove ha messo in evidenza che il throughput della rete

- è crescente al decrescere del periodo di trasmissione della frame;
- cresce in modo proporzionale al workload della rete, pur mantenendosi di molto inferiore allo stesso (in termini di bit/s), fino a raggiungere un valore oltre il quale risulta costante al crescere del workload. Tale valore indica la condizione di **saturazione della rete**, in cui la stessa non raggiunge la condizione a regime e non riesce a fornire delle prestazioni commisurate al carico di lavoro richiesto.

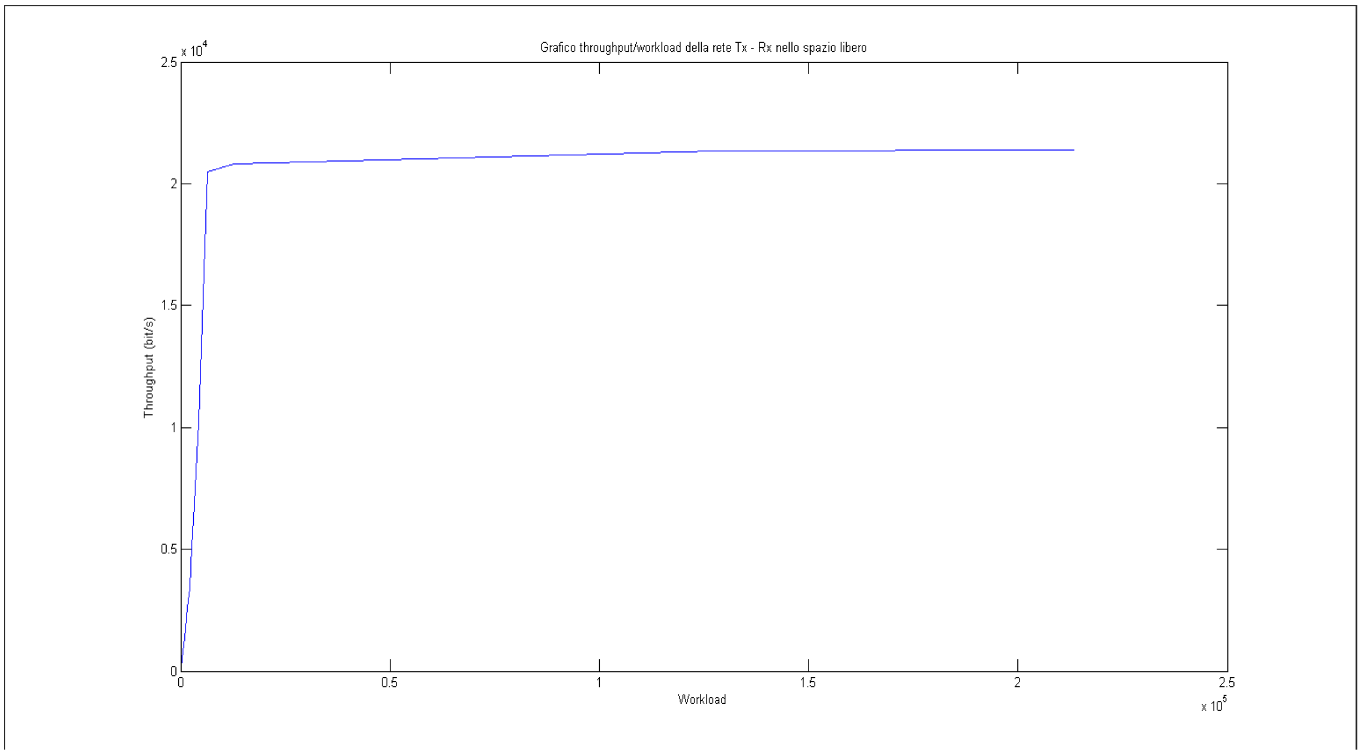


Grafico dell'andamento del throughput rispetto al workload

Rete 2

La seconda rete è costituita da quattro nodi, due nodi trasmittenti (**transmitter1** e **transmitter2**) e due nodi riceventi (**receiver1** e **receiver2**), interagenti tra di loro in uno scenario di simulazione predisposto per emulare le caratteristiche di un ambiente urbano reale, caratterizzato dalla presenza di *segnali interferenti* e di *alto rumore di canale*.

Le simulazioni svolte in tale scenario sono state mirate alla valutazione del comportamento della rete in oggetto in due distinte situazioni:

1. Le due coppie di trasmettitori e ricevitori operano in canali diversi, ma adiacenti;
2. Le due coppie di trasmettitori e ricevitori operano sullo stesso canale.

Per entrambi i casi sono state svolte dieci run di simulazione, con differenti valori di distanza tra i nodi. Il lavoro di definizione dei suddetti scenari è stato svolto dal gruppo composto da Alberto Attilio Brincat e Andrea Giuseppe Viola.

CASO 1

La prima coppia di nodi (**transmitter1** e **receiver1**) opera sul canale numero 9, la seconda (**transmitter2** e **receiver2**) sul canale 8.

La distanza tra i nodi costituenti le coppie è stata aumentata di 5 metri per simulazione, partendo da un valore di 5 m si è arrivato ad uno finale di 50 m.

I risultati ottenuti sono stati abbastanza soddisfacenti, in quanto non vi sono state collisioni ed il Packet Loss, come ci si aspettava, ha registrato un incremento lineare con l’aumento della distanza tra le coppie di nodi.

Di seguito sono riportate le curve che mostrano i risultati ottenuti relativi al **Packet Loss Ratio** (**PLR**) per ciascun ricevitore.

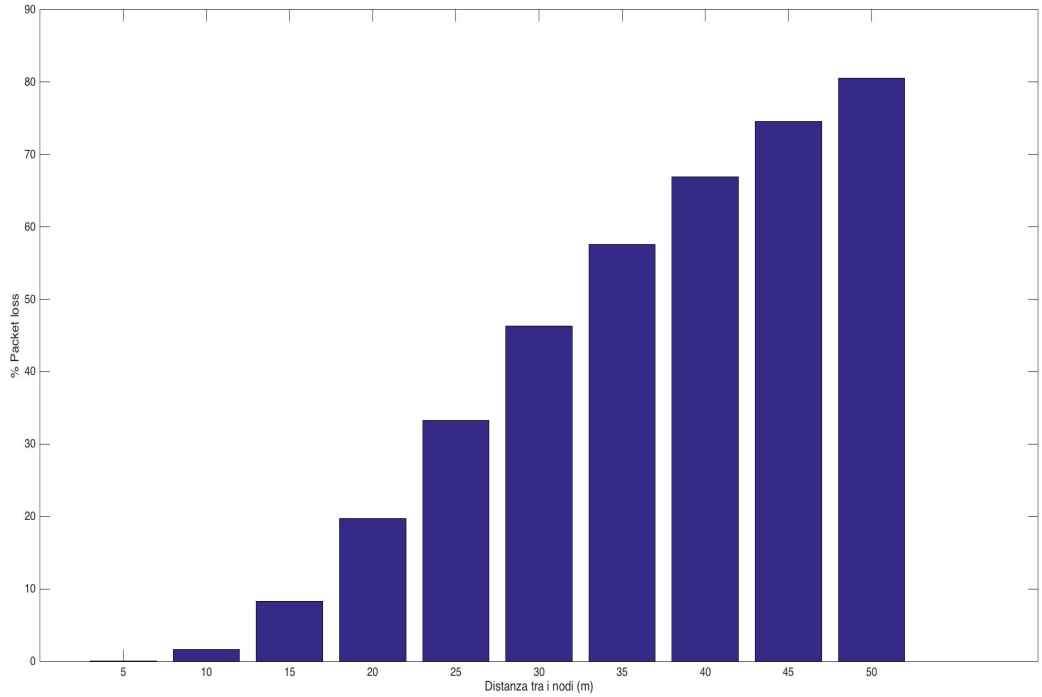


Grafico del Packet Loss Ratio in funzione della distanza, per Receiver1

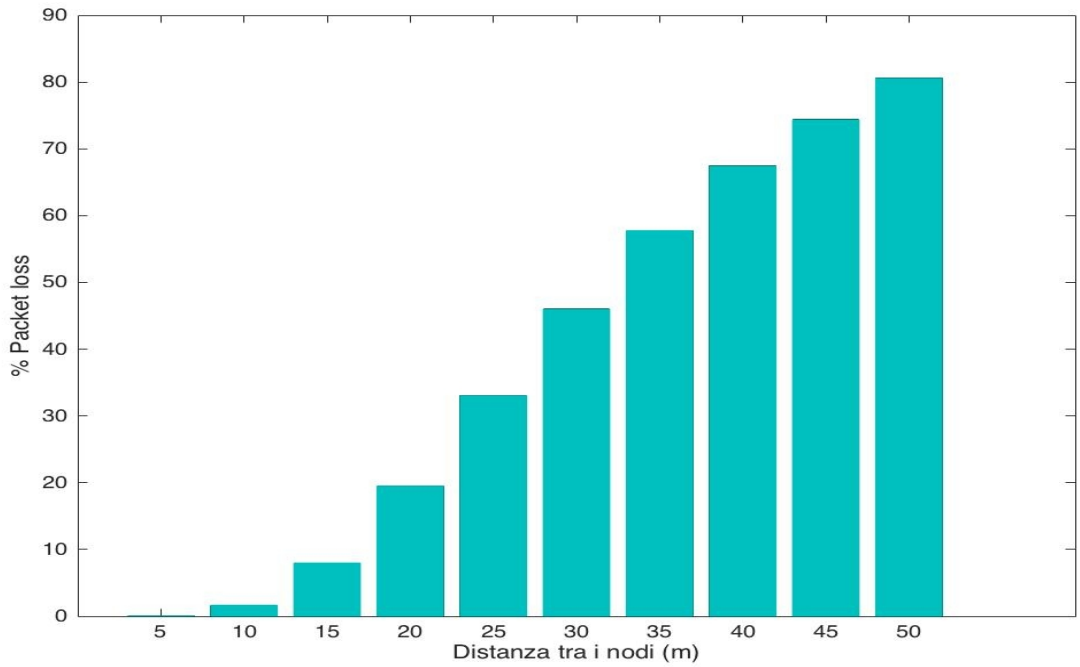


Grafico del Packet Loss Ratio in funzione della distanza, per Receiver2

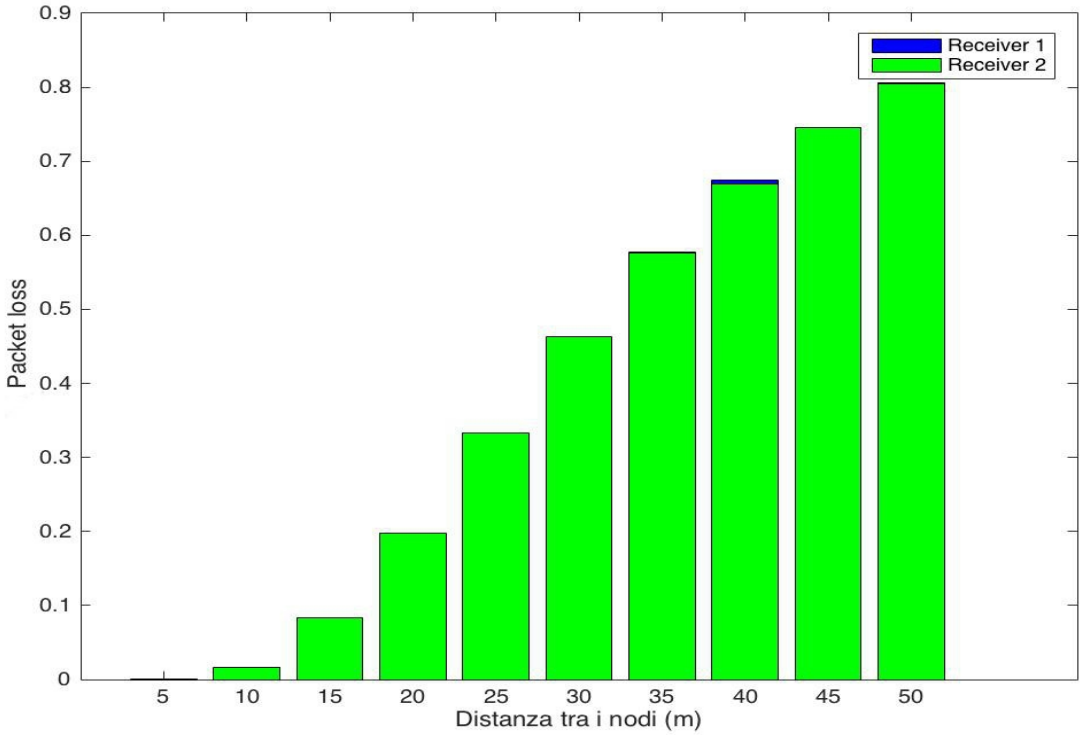


Grafico di confronto fra il Packet Loss Ratio di Receiver1 e Receiver2

Come si evince dal precedente grafico, i valori di PLR rilevati in corrispondenza dei due nodi riceventi presentano uno scostamento minimo.

Di seguito vengono riportati i grafici relativi al throughput di rete, misurato in corrispondenza dei nodi Receiver1 e Receiver2, il quale risulta essere decrescente al crescere della distanza tra nodo trasmittente e corrispettivo ricevente.

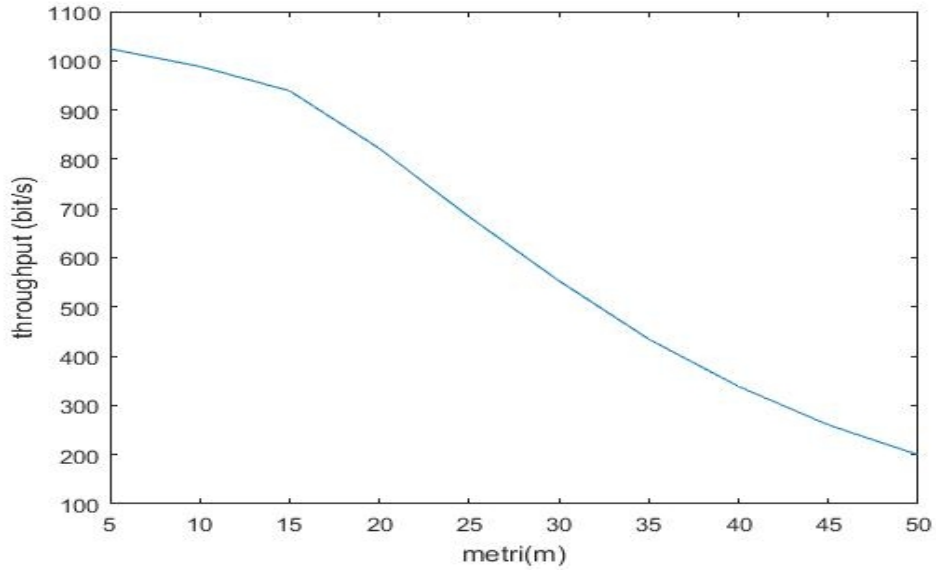


Grafico del Throughput in funzione della distanza, per Receiver1

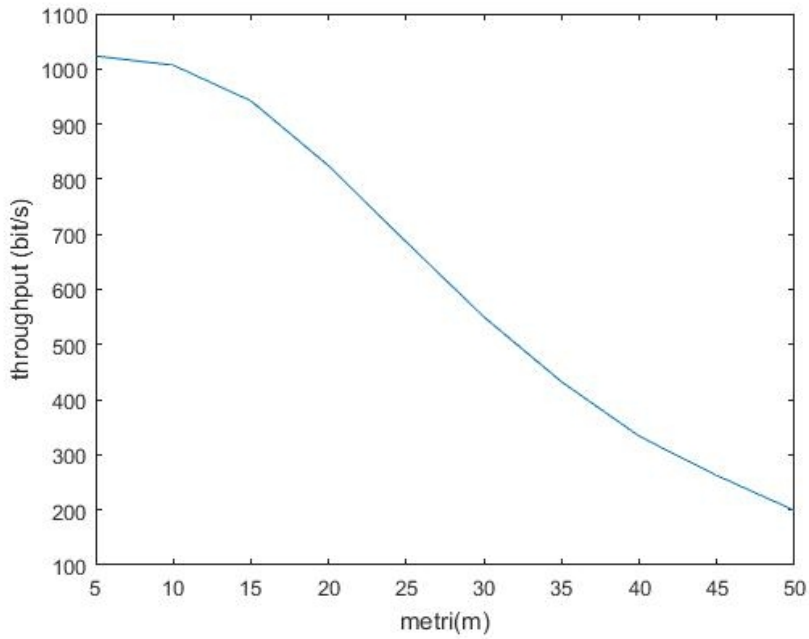


Grafico del Throughput in funzione della distanza, per Receiver2

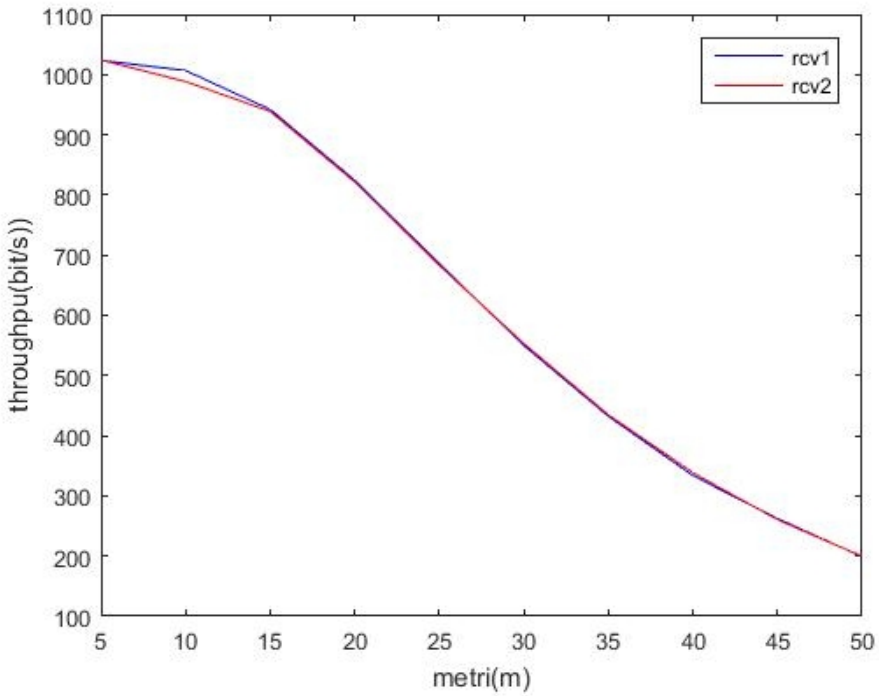


Grafico di confronto fra il throughput di Receiver1 e Receiver2

Come si evince dal precedente grafico, i valori di throughput rilevati in corrispondenza dei due nodi riceventi presentano uno scostamento minimo.

Inoltre, come è possibile osservare, la curva che descrive l'andamento del throughput segue un andamento opposto rispetto alla curva che descrive il Packet Loss Ratio: all'aumentare della distanza tra il nodo ricevente ed il corrispondente nodo trasmettitore, si assisterà ad un aumento del numero di frame perse ed alla conseguente riduzione del throughput rilevato.

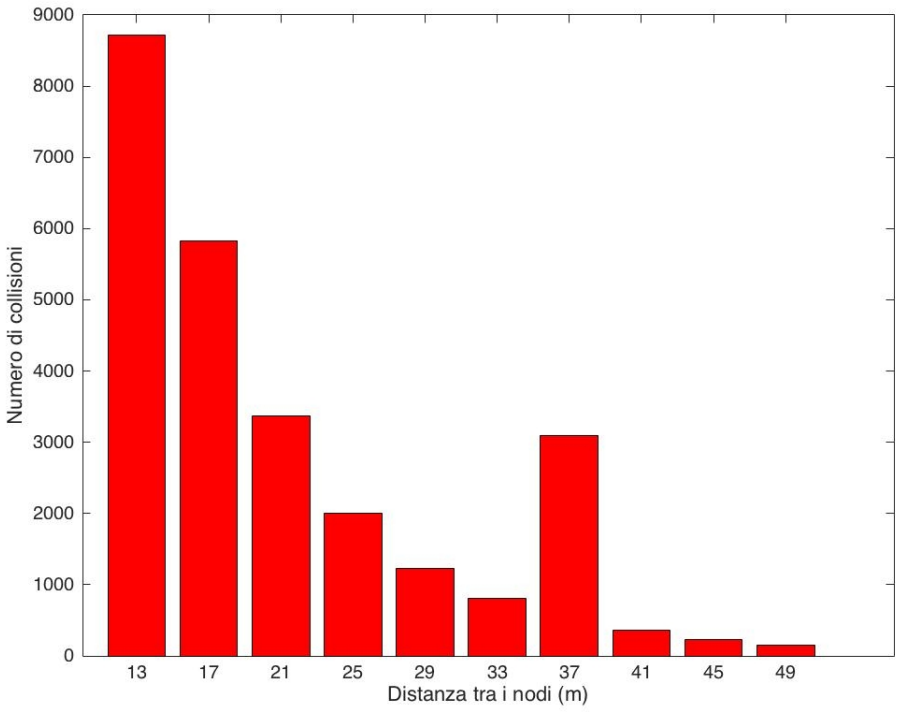
CASO 2

La prima coppia di nodi (**transmitter1** e **receiver1**) e la seconda (**transmitter2** e **receiver2**) operano sul canale 8. In questo particolare scenario di simulazione i nodi trasmittenti disturberanno l'uno la trasmissione dell'altro, generando delle **collisioni**.

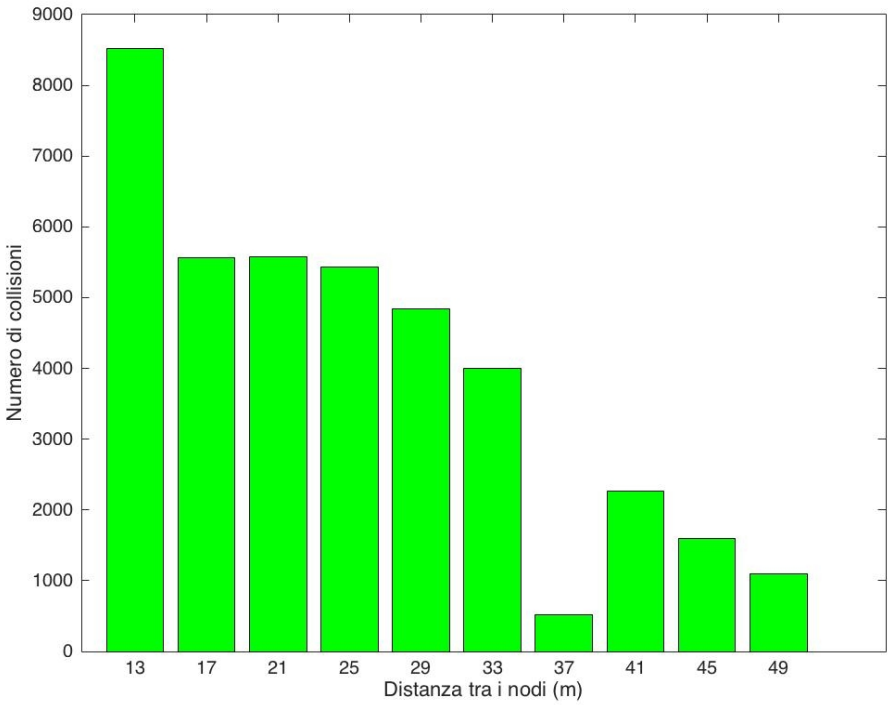
La distanza reciproca dei nodi è stata aumentata progressivamente di 2 metri per ciascun run di simulazione eseguito.

I risultati ottenuti dalle simulazioni eseguite sulla rete così configurata risultano essere differenti rispetto a quelli ottenuti nel caso 1, in quanto le trasmissioni intercorse sullo stesso canale generano dei fenomeni di collisione, che si registrano decrescenti, con andamento lineare, all'aumentare della distanza tra i nodi. Un eccezione a tale andamento è stata riscontrata al valore di distanza corrispondente a 37 m, dove è presente un picco superiore di collisione per il primo ricevitore, ed inferiore nel secondo.

Di seguito i grafici che mostrano l'andamento del numero di collisioni rilevate, al variare della distanza reciproca dei nodi, nella configurazione di rete utilizzata.



Numero di collisioni al variare della distanza, per Receiver1



Numero di collisioni al variare della distanza, per Receiver2

Di seguito i grafici che mostrano l'andamento del Packet Loss Ratio, al variare della distanza reciproca dei nodi, nella configurazione di rete utilizzata.

Come si evince da tali grafici, la presenza di entrambi i trasmettitori della rete nelle immediate vicinanze del nodo Receiver2 ha influito negativamente sulle prestazioni del nodo, che presenta un PLR medio maggiore rispetto a quello rilevato sul Receiver1.

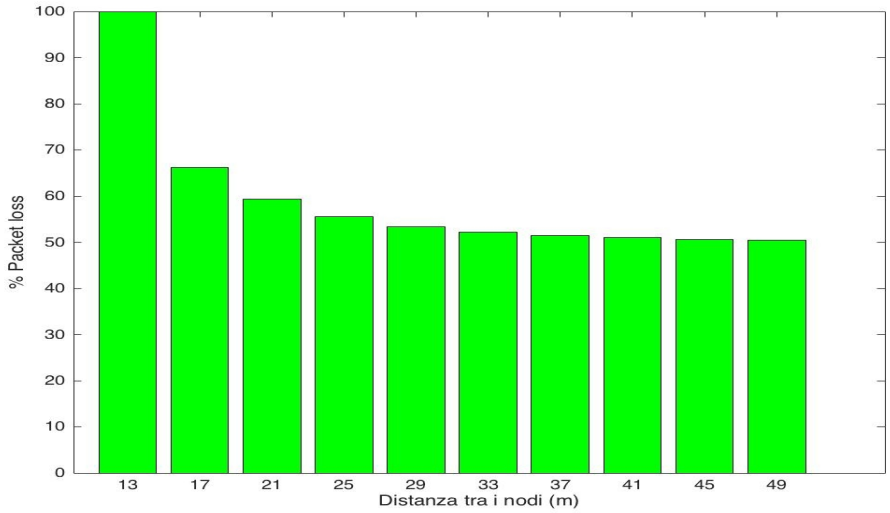


Grafico del Packet Loss Ratio in funzione della distanza, per Receiver1

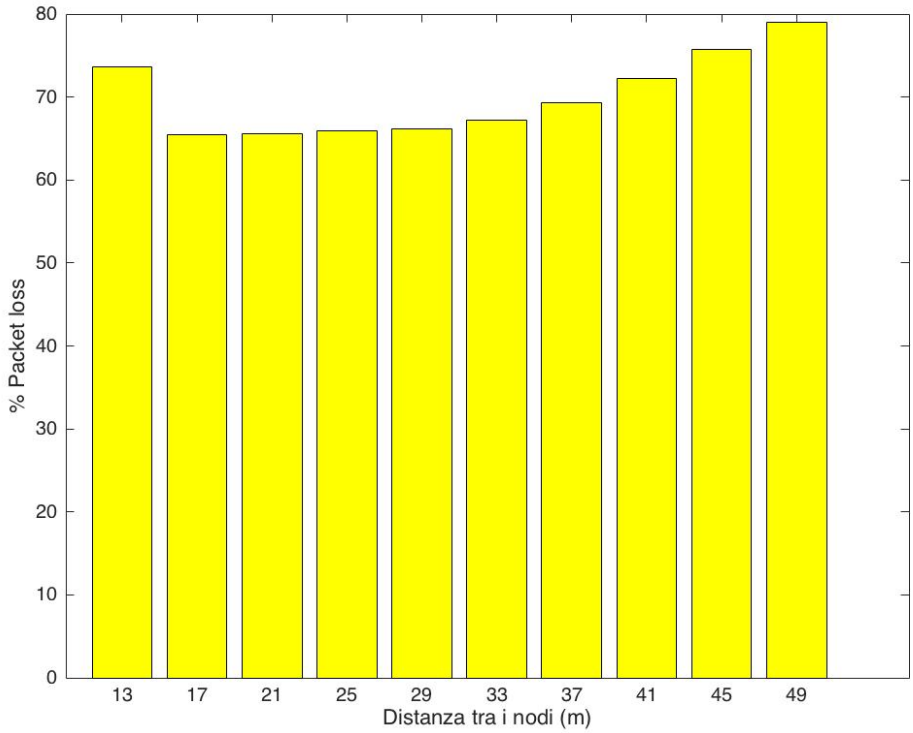


Grafico del Packet Loss Ratio in funzione della distanza, per Receiver2

Sono stati inoltre eseguiti diversi run di simulazione al fine di determinare l'andamento del throughput della rete rispetto al workload della stessa, al variare del periodo di trasmissione di una frame e della distanza tra nodi riceventi ed i rispettivi nodi trasmittenti.

Per ottenere i dati necessari a tale proposito, lo scenario di simulazione presentato nel caso 2 è stato così configurato:

- Le due coppie di nodi Tx-Rx operano su canali distinti ma adiacenti;
- periodo di trasmissione delle frame decrescente, a partire da 600 ms fino a 5 ms;
- distanza tra nodo trasmittente e nodo ricevente crescente, da 30 a 70 metri.

I risultati ottenuti dalle suddette prove, e relativi alle statistiche misurate sul nodo Receiver1, hanno messo in evidenza che il throughput della rete:

- è crescente al decrescere del periodo di trasmissione della frame;
- cresce in modo proporzionale al workload della rete, pur mantenendosi di molto inferiore allo stesso (in termini di bit/s), fino a raggiungere un valore oltre il quale risulta costante al crescere del workload. Tale valore indica la condizione di **saturazione della rete**, in cui la stessa non raggiunge la condizione a regime e non riesce a fornire delle prestazioni commisurate al carico di lavoro richiesto;
- throughput e workload della rete seguono l'andamento descritto sopra e risultano essere decrescenti al crescere della distanza fra nodo trasmittente e nodo ricevente.

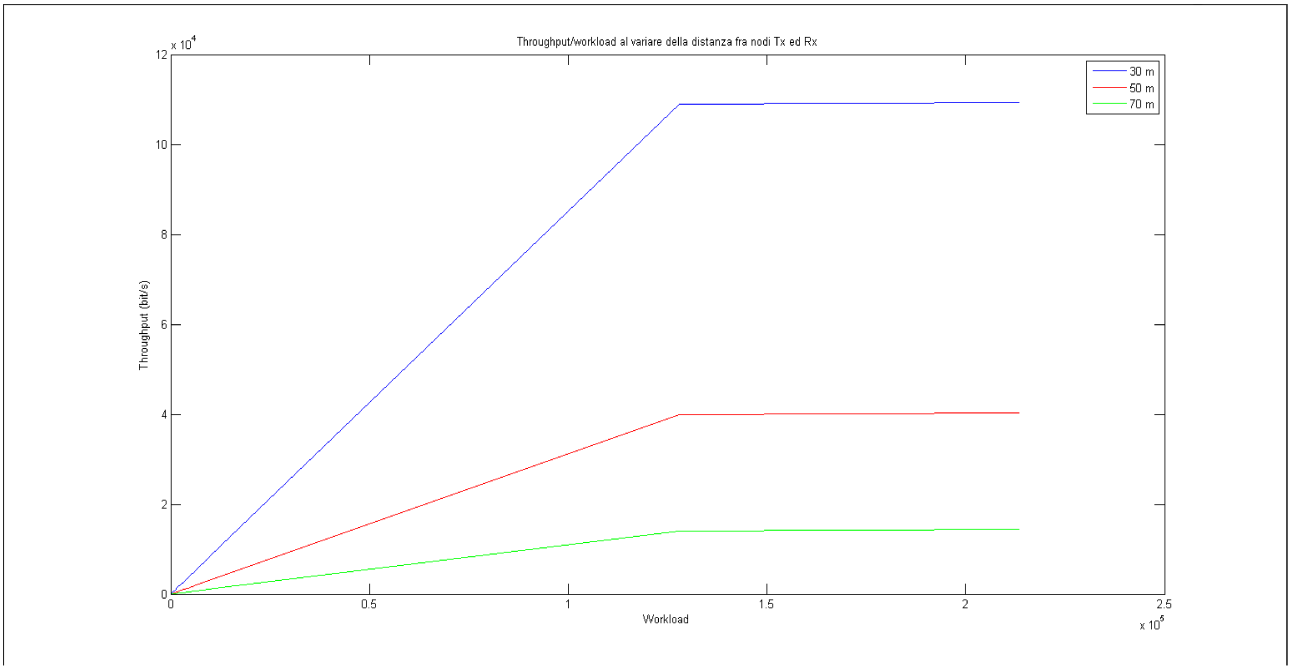


Grafico dell'andamento del throughput rispetto al workload, Rete 2

Conclusione

Nella presente relazione progettuale é stato discusso lo sviluppo e l'implementazione di un modello di simulazione di un ricetrasmettitore Wi-Fi, il cui modulo del livello fisico é basato su quello previsto dallo standard IEEE 802.15.4 e presente nel framework INETMANET di OMNeT++.

In seguito si sono predisposte due diverse topologie di reti, le quali sono state opportunamente configurate per costituire degli scenari di simulazione significativi e che sono serviti per la raccolta delle statistiche necessarie per verificare il corretto funzionamento del modulo realizzato.

Le simulazioni eseguite hanno messo in evidenza un significativo aumento del packet loss, e di conseguenza un abbassamento del throughput, all'aumentare della distanza tra nodo trasmittente e ricevente, e ciò è dovuto al fatto che si lavora in un uno scenario che prevede attenuazione del segnale e la presenza di segnali interferenti e disturbi.

Effettuando gli stessi rilevamenti statistici su di una rete operante in uno scenario configurato per emulare il modello di propagazione nello spazio libero, ovvero privo di attenuazioni e disturbi del segnale, il packet loss non dipenderà più dalla distanza dei nodi trasmittente e ricevente, almeno entro il raggio di copertura in cui il segnale verrà ricevuto ancora dal nodo con potenza sufficiente per essere considerato significativo.

Concluse le simulazioni sui due scenari differenti ed a seguito dell'analisi dei dati raccolti, il modello é risultato pienamente funzionante e le sue prestazioni coerenti con quelle che ci si aspettava di ottenere con modulo di rete Wi-Fi così costituito.