



# **UNIVERSITÀ DEGLI STUDI DI CATANIA**

**DIPARTIMENTO DI INGEGNERIA ELETTRICA, ELETTRONICA E INFORMATICA**

---

**CORSO DI LAUREA IN INGEGNERIA INFORMATICA**

---

**Tommaso Michele Salvatore Ruscica**

**Analisi prestazionale di un'implementazione di Random Linear  
Network Coding per il risparmio energetico in una rete cloud**

**TESI DI LAUREA**

**Relatore:**

**Chiar.mo Prof. Ing. G. Schembra**

---

**ANNO ACCADEMICO 2012 / 2013**



## **Sommario**

*Network Coding* è una nuova tecnica di trasmissione che ha catturato l'interesse di esperti nell'Ingegneria della Teoria dell'Informazione e nelle Telecomunicazioni per la capacità nell'aumentare il throughput , la robustezza della rete e diminuire il consumo di energia.

In questa tesi si intende esplorare e conoscere le caratteristiche del *Network Coding* attraverso la simulazione su di una specifica rete.

Nelle varie simulazioni viene visto il comportamento del singolo nodo della rete e le varie fasi di codifica e decodifica di un messaggio.

Tale simulazione viene implementata con l'uso dell'ambiente di lavoro MATLAB.



# INDICE

Lista Simboli.....	VII
Lista Figure.....	IX
Lista Tabelle.....	X
CAPITOLO 1.....	1
Introduzione alle reti telematiche.....	1
1.1 Modello generale di una rete di TLC.....	3
1.2 Tipologia delle informazioni fondamentali.....	6
1.3 Topologie di rete.....	7
1.4 Modalità di trasferimento dell'informazione.....	10
1.5 Tecnologie di trasmissione.....	13
1.6 Panoramica Network Coding.....	15
1.7 Obiettivi.....	17
1.8 Panoramica Tesi.....	17
CAPITOLO 2.....	18
Principio di funzionamento,caratteristiche tecniche e vantaggi di una rete con RLNC.....	18
2.1 Network Coding.....	18
2.1.1 Esempio rete a farfalla.....	18
2.1.2 Sfide.....	21
2.1.2.1 Robustezza.....	23
2.1.3 Teorema principale di una rete Multicast.....	25
2.1.3.1 Min-cut Max-flow.....	25
2.2 Linear Network Coding.....	27
2.2.1 Codifica.....	28
2.2.2 Decodifica.....	33
2.2.3 Campo di Galois.....	35
2.2.4 Eliminazione Gaussiana.....	40
2.2.4.1 Algoritmo di Gauss.....	42
2.2.4.2 Algoritmo di Gauss-Jordan.....	43
2.3 Random Linear Network Coding.....	44
2.4 Applicazioni Network Coding.....	46
2.4.1 Reti Ad-Hoc.....	47
2.4.2 P2P.....	47
2.4.3 Reti Wireless.....	48
2.4.3.1 Panoramica COPE.....	50
CAPITOLO 3.....	52

Simulazione di una rete con MATLAB applicando RLNC.....	52
3.1 MATLAB.....	52
3.1.1 Campo di Galois in MATLAB.....	53
3.2 Introduzione al simulatore.....	56
3.3 File RLNC.m.....	59
3.3.1 Setup.....	59
3.3.2 Inizializzazione.....	61
3.3.3 Creazioni messaggi.....	62
3.3.4 Invio.....	63
3.3.5 Ricezione.....	65
3.3.6 Codice completo.....	66
3.4 File NODO.m.....	68
3.4.1 Inizializzazione.....	68
3.4.2 Primo nodo (nodo immaginario).....	69
3.4.3 Secondo nodo (nodo reale).....	70
3.4.4 Nodi intermedi e nodo destinatario.....	71
3.4.5 Codice completo.....	75
3.5 Valutazione delle prestazioni.....	80
CAPITOLO 4.....	85
Conclusioni.....	85
Riferimenti.....	86

## **Lista Simboli**

NC	<i>Network Coding</i>
LNC	<i>Linear Network Coding</i>
RLNC	<i>Random Linear Network Coding</i>
GF	<i>Campo di Galois</i>
P2P	<i>Peer-to-peer</i>
LAN	<i>Local Area Network</i>
TCP	<i>Transmission Control Protocol</i>
$G=(V,E)$	<i>Grafico Rete</i>
V	<i>Insieme di nodi</i>
E	<i>Insieme di collegamenti</i>
COPE	<i>Opportunistic Coding</i>
IP	<i>Internet Protocol Address</i>
MAC	<i>Media Access Control</i>
S	<i>Sorgente</i>

XOR	<i>OR Esclusivo</i>
N	<i>Nodo corrente</i>
$G^N$	<i>Global Encoding Vector nodo N</i>
$Y^N$	<i>Information Vector nodo N</i>
$g^{loc}$	<i>Local Encoding vector</i>
$G^{prec}$	<i>Encoding Vector nodi precedenti</i>
$Y^{prec}$	<i>Information Vector nodi precedenti</i>
	Numero uscite
m	<i>Numero ingressi</i>
k	<i>Numero messaggi</i>
n	<i>Numero nodi rete</i>
M	<i>Messaggio</i>

# **Lista Figure**

Figura 1.0 -	Esempio di rete di calcolatori.....	2
Figura 1.1 -	Esempio di applicazione del modello generale a livelli(rete TCP/IP trasportata su rete telefonica).....	5
Figura 1.2 -	Articoli NC pubblicati negli anni.Dati presi a Luglio/2010[4].....	16
Figura 2.1 -	Rete a farfalla con routing tradizionale. a)nodo B invia simbolo b. b)nodo B invia simbolo a.....	19
Figura 2.2 -	Rete a farfalla utilizzando NC.....	20
Figura 2.3 -	Nodo 1 e 2 inviano un singolo pacchetto per unità di tempo.....	23
Figura 2.4 -	Connessione Unicast in una rete con capacità collegamenti unitaria[3].....	26
Figura 2.5 -	LNC:Local/Global coding vector di un Linear Network Coding 2-dimensionale.....	27
Figura 2.6 -	Pacchetto ricevuto da un nodo.....	29
Figura 2.7 -	LNC:Local/Global encoding vector aggiornati.....	31
Figura 2.8 -	Formato pacchetto NC.Adattato da [8].....	32
Figura 2.9 -	Esempio di RLNC.....	44
Figura 2.10 -	a)Download completo-ogni utente scarica il file. b)Download condiviso-ogni utente scarica una parte di file[32].....	47
Figura 2.11 -	Esempio di rete ad-hoc wireless:store-and-forward contro network coding.....	49
Figura 2.12 -	Esempio di Opportunistic Coding (COPE).....	50
Figura 3.1 -	Rete utilizzata come esempio nella simulazione.....	60
Figura 3.2 -	Numero di trasmissioni per ricevere un messaggio con rete a farfalla con NC.....	80
Figura 3.3 -	Numero di trasmissioni per ricevere un messaggio con rete a farfalla senza NC.....	81
Figura 3.4 -	Numero di trasmissioni per ricevere un un blocco di tre messaggi con rete a farfalla modificata con NC.....	82
Figura 3.5 -	Numero di trasmissioni per ricevere un un blocco di tre messaggi con rete a farfalla modificata senza NC.....	83

## **Lista Tabelle**

Tabella 2.1 -	Operazioni logiche XOR.....	21
Tabella 2.2 -	Operazioni somma e prodotto di GF(2).....	36
Tabella 2.3 -	Rappresentazioni differenti di elementi di un campo finito.....	37
Tabella 2.4 -	Somma e prodotto di GF( $2^2$ ). Polinomio irriducibile è $p(x) = x^2 + x + 1$ .....	38
Tabella 2.5 -	Somma e prodotto di GF( $2^3$ ). Polinomio irriducibile è $p(x) = x^3 + x + 1$ .....	39
Tabella 3.1 -	Funzione gf MATLAB, polinomio primitivo di default.....	53

# CAPITOLO 1

## Introduzione alle reti telematiche

Oggiorno, le reti di comunicazioni quali internet, peer-to-peer, wireless e reti ad-hoc, rappresentano un ruolo fondamentale nella vita di tutti i giorni. Una rete è costituita da un insieme di nodi collegati tra di loro dove uno o più nodi(Sorgenti) inviano un messaggio destinato a uno o più nodi(Destinatari). In informatica e telecomunicazioni, una rete di calcolatori è una particolare rete di telecomunicazione che permette lo scambio o condivisione di dati informativi e risorse(sia hardware sia software) tra diversi calcolatori attraverso cavi, sistemi radio o altri sistemi elettromagnetici e ottici. Ogni dispositivo di ricetrasmissione è univocamente distinguibile tramite *indirizzo di rete* per consentire l'instradamento dell'informazione da e verso i destinatari corretti. L'indirizzo di rete può essere espresso sotto vari formati a seconda del tipo e del ruolo del dispositivo, per esempio: un numero telefonico; un indirizzo *IP*; un indirizzo *MAC* e così via. L'insieme degli indirizzi di rete costituisce lo *spazio degli indirizzi* per quella rete.

I collegamenti di rete tra i dispositivi, attraverso i quali avviene lo scambio dell'informazione, possono essere di tipo fisico (per esempio il doppino di rame che collega un telefono alla centrale più vicina; una fibra ottica che collega due apparati di trasmissione; la portante trasmittiva di un ponte radio) oppure di tipo logico (per esempio, il circuito equivalente che collega

direttamente due dispositivi qualsiasi della rete, astratto dalla complessità e dalla modalità dei collegamenti fisici effettivamente utilizzati per realizzarlo). Il corretto trasferimento dell'informazione è assicurato mediante funzionalità specifiche (funzionalità di rete) come la **segnalazione** (per l'inizio e la fine dello scambio informativo), la **commutazione** (per l'instradamento tra gli utenti finali), la **trasmissione** (per il trasferimento fisico del segnale), la **gestione** (per il controllo della correttezza dello scambio e per l'utilizzo ottimale delle risorse di rete )[19].



Figura 1.0 Esempio di rete di calcolatori

## **1.1 Modello generale di una rete di TLC**

Il modello astratto di rete, standardizzato dall'ITU-T nella raccomandazione G.805[14] e applicabile a qualsiasi rete di telecomunicazioni, indipendente dal tipo di tecnologia impiegato, è un modello a strati o livelli differenti legati tra di loro da una relazione di tipo client-server, con una scomposizione a dettaglio crescente di tipo top-down.

Ogni livello è caratterizzato da un'informazione caratteristica che lo contraddistingue e che può essere descritta in modo indipendente sia dall'informazione caratteristica del relativo livello client che da quella del relativo livello server, ad esclusione dei punti di passaggio tra un livello e l'altro (che rappresentano l'adattamento dell'informazione tra i diversi livelli), rendendo di fatto ogni livello un'entità logicamente coerente e autonoma in se stessa.

Il modello architetturale, oltre alle relazioni tra i livelli, individua e modella per ciascun livello anche le relative funzionalità di rete, definendo la capacità di elaborazione del segnale e delle informazioni di controllo, compreso l'instradamento, e le relazioni di connettività all'interno del livello, tramite una rappresentazione astratta di tipo generale che può essere applicata in modo ricorsivo.

Ogni livello è descrivibile come una (sotto)rete composta da punti di connessione e dai loro collegamenti. Il punto di connessione rappresenta un dispositivo di rete e modella le funzioni di segnalazione, di commutazione e di controllo dell'integrità dell'informazione. Il collegamento, in quanto elemento di connessione tra due nodi diversi, rappresenta i canali logici sui quali vengono trasmesse le informazioni e modella la funzione di trasmissione dell'informazione caratteristica del livello.

Ogni collegamento di un livello può essere a sua volta scomposto in modo ricorsivo come risultante di un trasporto da parte di un livello server, caratterizzato dai propri punti di connessione e collegamenti e così via, a partire dal livello più alto, in cui i punti di connessione sono gli utenti finali e i collegamenti rappresentano connessioni logiche dirette tra gli utenti, fino a raggiungere il livello più basso, non ulteriormente scomponibile, che è quello del trasporto fisico diretto del segnale, in cui i punti di connessione rappresentano i trasmettitori e i ricevitori fisici e i collegamenti coincidono con il mezzo fisico diretto tra un trasmettitore e il ricevitore immediatamente adiacente (per esempio: un cavo, una fibra ottica, la portante tra due antenne).

Nella Figura 1.1 è possibile vedere un esempio di applicazione del modello generale a livelli (TCP/IP trasportata su rete telefonica)[19].

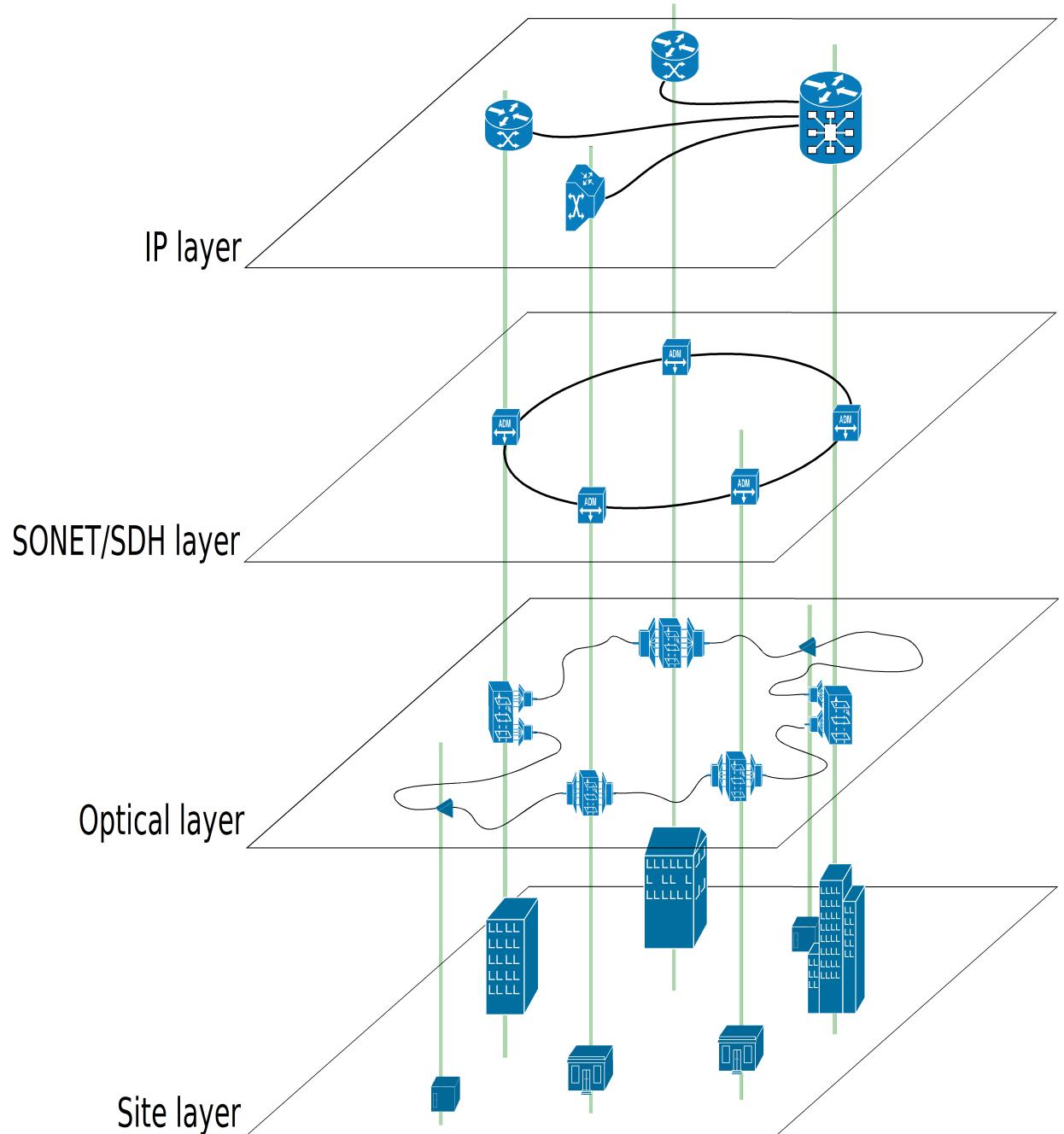


Figura 1.1 Esempio di applicazione del modello generale a livelli (rete TCP/IP trasportata su rete telefonica).

## 1.2 Tipologia delle informazioni fondamentali

In una rete di Telecomunicazioni si possono identificare tre tipologie di informazioni di base convogliate attraverso la rete stessa. I tre tipi di informazione sono indipendenti tra loro (spesso anche a livello di circuiti) ma svolgono funzioni complementari che, una volta combinate insieme, garantiscono la corretta operabilità della rete.

Le tre tipologie di informazioni fondamentali sono:

- il contenuto informatico vero e proprio scambiato tra gli utenti e il relativo insieme di collegamenti: questo complesso costituisce il *Data plane* (piano dei dati)
- l'insieme delle informazioni di servizio che vengono scambiate direttamente tra i nodi della rete per determinare in modo autonomo, automatizzato e in tempo reale la gestione ottimale del segnale, per esempio tramite opportuni protocolli di segnalazione; questo complesso costituisce il *Control plane* (piano di controllo o di segnalazione)
- l'insieme delle informazioni che i nodi scambiano con entità di controllo esterne alla rete, tipicamente si sistemi di gestione di rete, per funzionalità di monitoraggio, supervisione, gestione e controllo della configurazione dei nodi; questo complesso costituisce il *Management plane* (piano di gestione)

L'insieme delle funzionalità logiche della rete suddivise nei tre piani della Figura 1.1 costituisce l'architettura logico-funzionale di rete[19].

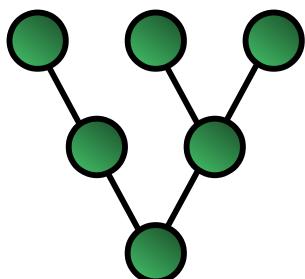
## 1.3 Topologie di rete

Un rete di TLC dal punto di vista fisico è un'interconnessione di rami e di nodi dove i rami hanno funzionalità di collegamento tra i nodi e i nodi funzionalità di commutazione. Essa è rappresentabile dunque attraverso un grafo. La topologia rappresenta le modalità di interconnessione fisica o logica dei nodi tra di loro.

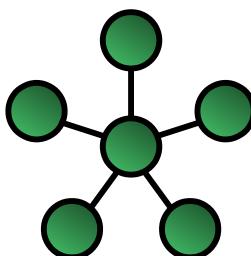
Nell'ambito di una rete è possibile definire delle strutture topologiche elementari, che sono:

- topologia ad *albero*: struttura lineare a livelli in cui per ogni coppia di nodi esiste un solo percorso di collegamento; ogni nodo è collegato a un solo nodo del livello superiore tramite un solo ramo e a uno o più nodi del livello inferiore tramite uno o più rami dedicati (diramazione);
- topologia a *stella*: struttura a raggiera in cui i nodi sono tutti collegati tramite un solo ramo dedicato a un nodo centrale; in realtà è un caso particolare di topologia ad albero, caratterizzata da un unico livello di collegamenti tutti afferenti a un unico nodo;
- topologia *ad anello*: struttura circolare in cui ogni nodo è collegato in modo lineare a due nodi adiacenti;
- topologia *lineare*: struttura lineare in cui ciascun nodo tranne i nodi estremi è collegato a due nodi adiacenti; anche questo è un caso particolare di topologia ad albero, caratterizzata da livelli costituiti da un unico nodo senza diramazioni;

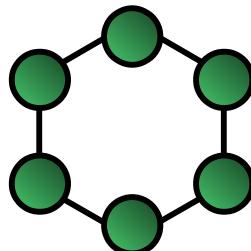
- topologia *a bus*: struttura in cui tutti i nodi condividono lo stesso unico collegamento;
- topologia *completamente magliata*: struttura in cui ogni nodo è connesso direttamente con tutti gli altri nodi della rete con un ramo dedicato per ciascun collegamento;
- topologia *parzialmente magliata* o *magliata*: combinazione tra una sottorete a topologia completamente magliata e una o più sottoreti a topologia diversa, per esempio ad albero o a stella.



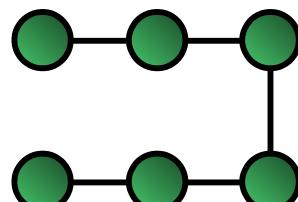
Topologia ad albero



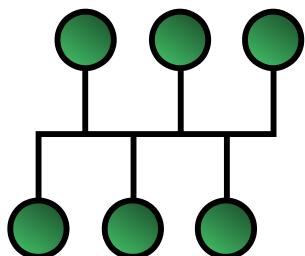
Topologia a stella



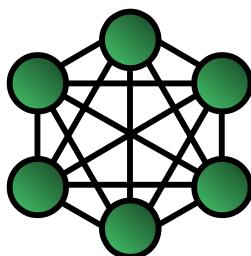
Topologia ad anello



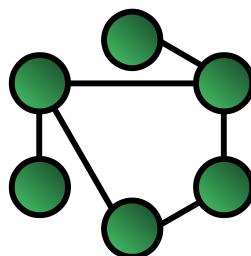
Topologia lineare



Topologia a bus



Topologia compl.  
magliata



Topologia magliata

Per ogni topologia, il numero di rami R necessari per il collegamento è esprimibile come funzione del numero N dei nodi della rete. In particolare:

- per reti a stella, ad albero e lineari:  $R = N - 1$
- per reti ad anello:  $R = N$
- per reti a bus:  $R = 1$
- per reti completamente magliate:  $R = N(N-1)/2$
- per reti parzialmente magliate:  $(N-1) < R < N(N-1)/2$

Ciascuna topologia presenta caratteristiche diverse di complessità e di robustezza (capacità di sopravvivenza ai guasti di nodi e/o collegamenti). Per esempio la topologia completamente magliata rappresenta la robustezza massima rispetto ai guasti dei collegamenti (in assenza di isolamento di un nodo, è sempre possibile determinare il percorso che colleghi due nodi qualsiasi della rete) ma la relazione quadratica tra il numero dei nodi e il numero dei collegamenti necessari ne limita di fatto l'applicazione a reti con un numero relativamente piccolo di nodi. Analogamente, le topologie a stella o ad albero consentono di realizzare reti con un numero di nodi anche elevato ma presentano maggiori vulnerabilità ai guasti: in una rete a stella, per esempio, il guasto del nodo centrale compromette l'intera rete. Tuttavia per tutti i tipi di topologia è possibile aumentare anche in modo significativo la robustezza ai guasti tramite l'applicazione di opportuni meccanismi di protezione di rete che, pur introducendo ridondanze nell'utilizzo delle risorse di rete, consentono di far fronte in modo efficace a condizioni di guasto e di eliminare o diminuire drasticamente le vulnerabilità intrinseche di ciascun tipo di topologia[19].

## 1.4 Modalità di trasferimento dell'informazione

Lo scopo di una rete di telecomunicazioni è quello di consentire lo scambio di informazioni tra due utenti finali posizionati in posizioni geograficamente distinte. Lo scambio viene realizzato tramite la conversione dell'informazione in un segnale e nel successivo trasporto di questo segnale da un utente all'altro utilizzando le risorse della rete.

Si distinguono quindi due funzionalità essenziali:

- la funzionalità di *accesso* alla rete che identificano a loro volta la parte di rete nota come *rete di accesso*;
- la funzionalità di *trasporto* all'interno della rete che identificano a loro volta la parte di rete nota come *rete di trasporto o core network*.

La funzionalità di accesso comprende l'interazione diretta tra l'utente finale con il primo elemento della rete, che costituisce così il *nodo terminale* o più semplicemente il *terminale*; la conversione dell'informazione in segnale, tipicamente eseguita dal terminale; il trasferimento del segnale verso la funzionalità di trasporto.

La funzionalità di trasporto si attua attraverso un particolare modo di trasferimento e comprende l'instradamento del segnale tra i nodi interni della rete e il mantenimento dell'integrità del relativo contenuto informativo lungo tutto il percorso, in modo da raggiungere il o i terminali di destinazione per restituire l'informazione al o agli utenti finali tramite le funzioni di accesso.

Ogni nodo della sezione di trasporto della rete determina su quale collegamento ricevere e trasmettere il segnale tramite la funzionalità di *commutazione*, che si può realizzare secondo due meccanismi fondamentali:

- la *commutazione di circuito*, in cui una frazione fissa di capacità della rete viene stabilmente allocata per il segnale, costituendo così un canale diretto tra gli utenti finali.
- La *commutazione di pacchetto*, in cui l'informazione viene adattata in strutture quantizzate, i pacchetti, che vengono instradati singolarmente e in modo individuale, senza una preallocazione deterministica o dedicata delle risorse.

A titolo di esempio, la commutazione di circuito è la modalità tipicamente impiegata per il trasporto della telefonia mentre la commutazione di pacchetto viene tipicamente impiegata per il trasporto di dati di tipo informatico, per esempio nelle reti di calcolatori.

Le reti di telecomunicazioni, in particolare quelle che prevedono il trasporto integrato di [telefonia](#) e dati (*Next Generation Networking*), si stanno evolvendo verso una modalità di trasporto convergente in cui tutte le informazioni sono pacchettizzate, la scelta dell'instradamento avviene pacchetto per pacchetto ma per ciascun pacchetto tale instradamento è di tipo deterministico, sfruttando i meccanismi tipici della commutazione di circuito (predefinizione del canale) (es. commutazione a circuito virtuale): in questo modo la capacità della rete viene allocata solo in presenza effettiva del singolo pacchetto, secondo il meccanismo tipico della commutazione di pacchetto, potendo però allo stesso tempo sfruttare il collegamento statico tra due nodi (cioè il circuito fisso dedicato).

Tradizionalmente, i nodi per scambiarsi i messaggi utilizzano degli algoritmi di routing (*store-and-forward*).

Lo **store-and-forward** (liberamente traducibile come "immagazzina e riinvia") è una tecnica nella quale un'informazione (suddivisa in pacchetti), nel suo percorso tra le singole stazioni (o nodi) della rete, deve essere totalmente ricevuta, prima di poter essere ritrasmessa nel collegamento in uscita.

In pratica, il pacchetto viene ricevuto e verificato, e solo allora viene inoltrato al nodo successivo; ciò implica quindi lo svantaggio che se tra due singoli nodi vi è un ritardo, questo verrà moltiplicato per tutti i nodi che l'informazione dovrà attraversare per giungere a destinazione.

Allo stesso tempo, lo *store-and-forward* è alla base del funzionamento delle reti a *commutazione di pacchetto* e consente di raggiungere un'elevata utilizzazione delle risorse di rete, traendo massimo vantaggio dalla multiplazione statistica dei dati[19].

## 1.5 Tecnologie di trasmissione

La tecnologia di trasmissione è la modalità con cui l'informazione viene fisicamente trasmessa. Tipicamente, l'informazione viene convertita in un segnale di tipo elettrico, che si presta sia all'elaborazione di tipo digitale che alla conversione e riconversione in formati fisici differenti senza perdita di contenuto informativo e che può essere trasferito a velocità elevatissime, confrontabili con la velocità della luce.

L'informazione può essere trasferita direttamente nel suo formato elettrico tramite cavi realizzati in materiale conduttore (per esempio, il doppino telefonico o i cavi di collegamento per Ethernet realizzati in rame), oppure convertita in segnale ottico e trasmessa tramite fibra ottica o ancora convertita in un segnale elettromagnetico e trasmessa via radio (come nel caso delle radiocomunicazioni televisive, *wireless* e via satellite).

Il segnale elettrico può essere trasferito sia in forma analogica che digitalizzata, tramite un passaggio di conversione analogico-digitale. Dato che la digitalizzazione consente di applicare tecniche di elaborazione che riducono e in certa misura annullano gli effetti del degrado trasmissivo dovuto ai disturbi, alla distanza e alle caratteristiche del mezzo fisico (attenuazione, ritardo di propagazione eccetera), nonché tecniche di compressione senza perdita di informazione per segnali ad alta necessità di banda (come per esempio le applicazioni basate su video), quasi tutte le reti di telecomunicazioni utilizzano nel loro interno una trasmissione basata su

formati digitali. La trasmissione in forma analogica viene tipicamente impiegata solo nelle tratte terminali della rete (per esempio, nel collegamento tra la centralina telefonica e l'apparecchio domestico), dove la limitata lunghezza della tratta rende trascurabili o comunque accettabili gli effetti di degrado, o solo per specifiche applicazioni che per le loro caratteristiche presentano maggior robustezza nei confronti dei disturbi e del degrado trasmittivo (come nel caso delle trasmissioni televisive o radiofoniche di tipo analogico, caratterizzate da un'elevata ridondanza che compensa le eventuali perdite di trasmissione)[19].

## 1.6 Panoramica Network Coding

*Network Coding* è una recente tecnica che cambia la tradizionale idea di routing, infatti ogni nodo della rete esegue la combinazione lineare dei pacchetti in ingresso per ogni uscita del nodo.

Questo metodo è stato introdotto da R.Ahlsweide *et al.* [1] agli inizi del nuovo millennio, dimostrando come il network coding può aumentare il throughput della rete e migliorarne la robustezza.

Un esempio molto pratico è dato da un classico puzzle: *una gang di 5 pirati trova un tesoro. Il capitano dei pirati, preoccupato che il tesoro possa essere preso e tenuto da un solo pirata, conserva il tesoro in una cassaforte. Dopo egli ad ogni pirata dice qualcosa in modo tale che ognuno di essi da solo non riesca ad aprire la cassaforte.*

La soluzione è semplice, in pratica il capitano ha chiuso la cassaforte con un numero arbitrario (es. 81), quindi a un pirata ha detto  $x=8$  ad un altro  $y=1$  a un terzo  $x+y$  e ad un altro  $x+2y$  in modo tale che due pirati qualsiasi possano aprire la cassaforte, uno solo invece no.[20]

L'idea di base è relativamente semplice: i nodi intermedi ricombinano i pacchetti ricevuti, inviando una combinazione lineare di questi creando una nuova rappresentazione del pacchetto originale. Per poter decodificare e ritrovare il pacchetto originale, il ricevitore/destinatario deve avere un numero sufficiente di combinazioni lineari indipendenti di questi pacchetti.

L'argomento del network coding coinvolge principalmente il networking, graph theory, algoritmi di ottimizzazione, teoria dell'informazione, sicurezza, reti ad-hoc e monitoring network.

Un rilevante numero di studi di ricerca riguardo il network coding fu riportato da [1] risultando in continuo avanzamento, come illustrato in Figura 1.2.

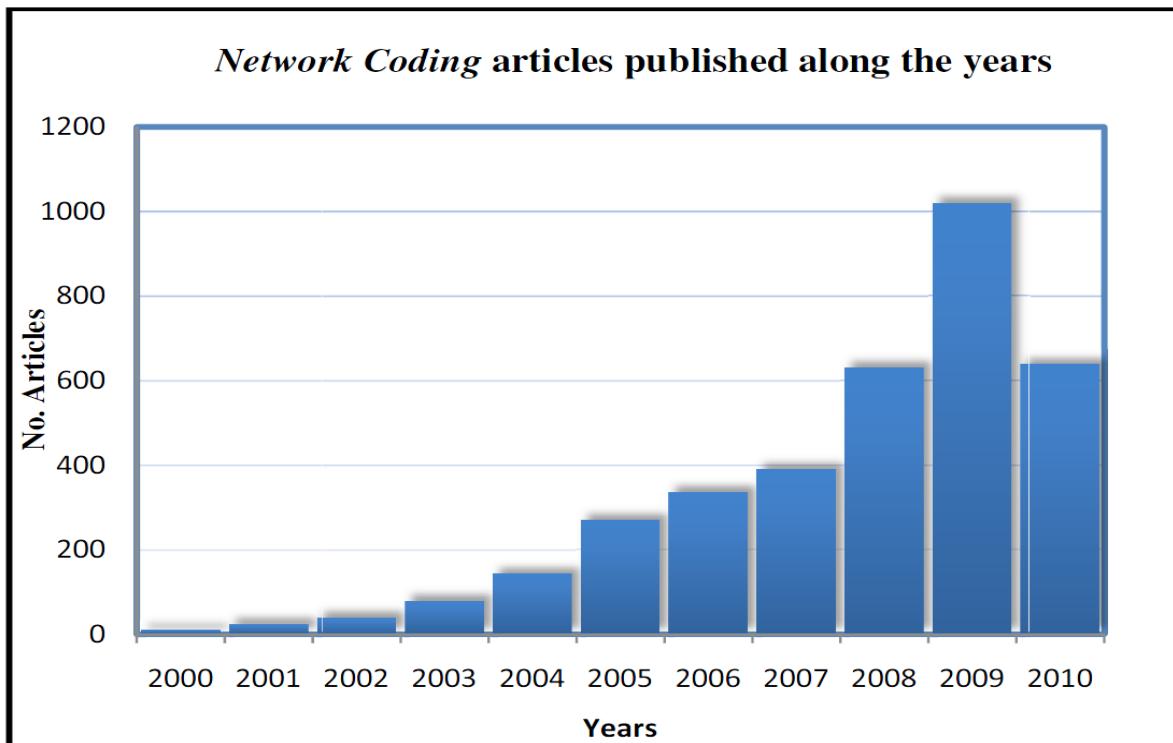


Figura 1.2. Articoli NC pubblicati negli anni. Dati presi a Luglio/2010[4].

## **1.7 Obiettivi**

L'obiettivo principale della tesi è esplorare e conoscere le caratteristiche della tecnica di comunicazione basata su RLNC ( Random Linear Network Coding ) creando un simulatore di una rete generica con MATLAB.

## **1.8 Panoramica Tesi**

Nel Capitolo 2 verranno introdotte le caratteristiche e le applicazioni del Network Coding e delle tecniche derivanti da questa quali il Linear Network Coding ( LNC ) e il Random Linear Network Coding ( RLNC ), con le loro relative differenze.

Nel Capitolo 3 verrà presentato e documentato il codice relativo al simulatore creato in ambiente MATLAB.

L'ultimo Capitolo sarà dedicato alle conclusioni sulla tesi.

# CAPITOLO 2

## **Principio di funzionamento, caratteristiche tecniche e vantaggi di una rete con RLNC**

In questo capitolo vedremo una semplice rete multicast, cioè un singolo nodo sorgente che invierà dei messaggi a più nodi destinatari, con l'uso del NC sottolineando le sue proprietà e applicazioni.

### **2.1 Network Coding**

I benefici del NC in una rete di comunicazione sono stati dimostrati dallo studio di diversi soggetti quali il throughput [1], ritardo end-to-end [2], risorse wireless [5], problemi di sicurezza e robustezza [3].

#### **2.1.1 Esempio rete a farfalla**

Assumendo una rete di comunicazione  $G=(V,E)$ , dove  $V$  e  $E$  sono gli insiemi di vertici(nodi) e connessioni, rispettivamente. Tale rete per semplicità avrà le seguenti caratteristiche:

- Aciclica
- Lineare

- Multicast
- Capacità canali unitaria
- Senza ritardi
- Direzionale

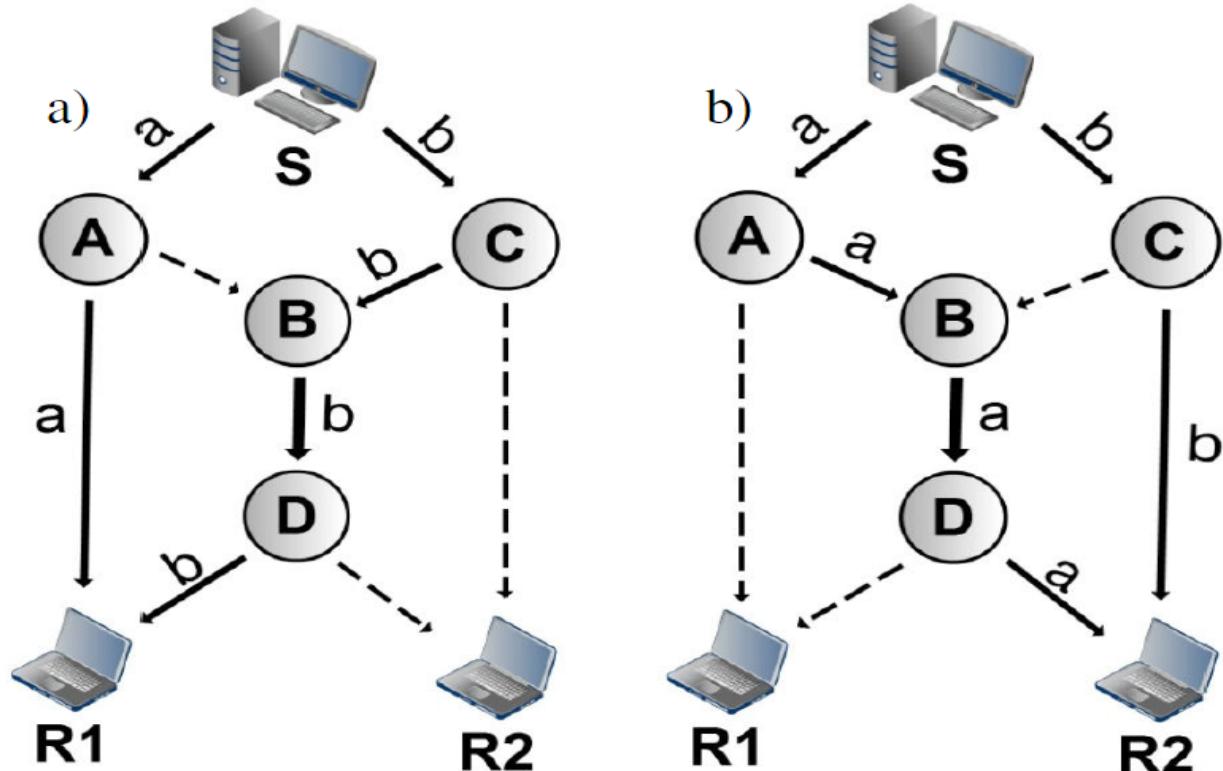


Figura 2.1. Rete a Farfalla con routing tradizionale. a) nodo B invia simbolo *b*. b) nodo B invia simbolo *a*.

La rete a farfalla è un classico esempio dove è possibile dimostrare il guadagno di throughput usando NC.

Nella Figura 2.1 si ha una sorgente **S** che invia due messaggi, *a* e *b* a due destinatari **R1** e **R2** in ogni slot di tempo.

Con routing tradizionale, il destinatario **R1** riceverà entrambi i messaggi attraverso il percorso {AR1} e il messaggio *b* lungo il percorso

$\{ \text{CB}, \text{BD}, \text{DR1} \}$ . Il destinatario **R2** riceverà il messaggio  $b$  lungo il percorso  $\{ \text{CR2} \}$  e il messaggio  $a$  lungo  $\{ \text{AB}, \text{BD}, \text{DR2} \}$ , come illustrato in Figura 2.1.

Inoltre si avranno 5 tempi di trasmissione per completare l'operazione.

Ora poiché la rete è di tipo multicast, assumeremo che entrambi i destinatari riceveranno da S entrambi i messaggi simultaneamente. Se i nodi inviano solamente le informazioni che ricevono (tradizionale *store-and-forward*) il canale  $\{ \text{BD} \}$  sarà un “collo di bottiglia” poiché potrà inviare solo uno dei due messaggi per volta ai destinatari.

Con NC entrambi i messaggi possono essere inviati tramite lo **XOR** (o combinazione lineare) dei due al nodo intermedio **B**, come si evince dalla Figura 2.2.

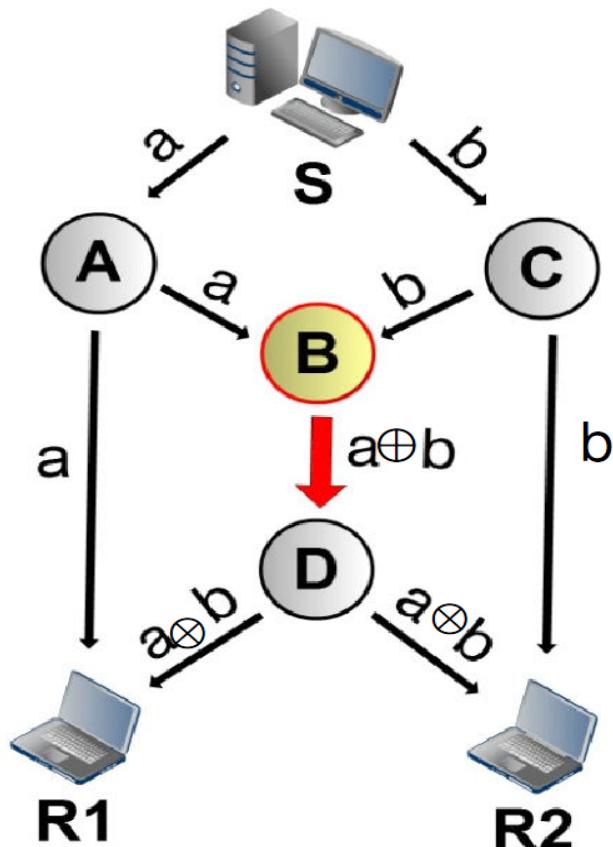


Figura 2.2. Rete a farfalla utilizzando NC

Un nuovo pacchetto  $c$  viene creato nel nodo  $B$ , dove  $c = a \oplus b$  e inviato attraverso  $BD$ .  $R1$  e  $R2$  riceveranno, rispettivamente,  $\{a, a \oplus b\}$  e  $\{b, a \oplus b\}$  in modo tale da ricavare  $a$  o  $b$  risolvendo un semplice sistema di equazioni e diminuendo i tempi di trasmissione a 4.

Il simbolo “ $\oplus$ ” denota l'operatore logico **XOR** che è specificato nella tabella 2.1.

$\oplus$	0	1
0	0	1
1	1	0

Tabella 2.1. Operazioni logiche XOR

Quindi i nodi intermedi della rete “codificano” (combinano) i pacchetti ricevuti e inviano questo nuovo pacchetto creato ai ricevitori.

### 2.1.2 Sfide

Lo sviluppo del NC ha creato un certo numero di sfide a dei problemi [12]:

- **COMPLESSITA'**

Il NC richiede che ogni nodo della rete abbia funzionalità addizionali.

Una questione importante è valutare la complessità richiesta dal network coding e investigare sul compromesso tra complessità e performance.

- **SICUREZZA**

La sicurezza nelle reti è un importante requisito, soprattutto per le reti bancarie, garantendo transazioni sicure contro attacchi esterni.

I meccanismi attualmente in vigore sono stati progettati attorno al presupposto che gli unici soggetti che possono manomettere i dati sono la fonte e la destinazione. Il NC richiede a nodi intermedi di fare delle operazioni sui dati. Quindi bisogna creare dei meccanismi diversi per permettere alla rete di modificare i dati mantenendo l'autenticità dei dati.

- **INTEGRAZIONE CON INFRASTRUTTURA ESISTENTE**

Integrare nuove tecnologie, quali NC, in architetture di reti già esistenti è uno dei problemi maggiori. L'ideale sarebbe applicare questa tecnica senza incorrere in drastici cambiamenti hardware e software. Una questione aperta è proprio come è possibile integrare il network coding negli attuali protocolli di rete.

### 2.1.2.1 Robustezza

- **ROBUSTEZZA ALLA PERDITA DI PACCHETTI**

La perdita di pacchetti in una rete ( es. wireless ) è dovuta a vari fattori ad esempio un sovraccarico del buffer oppure una collisione.

Ci sono alcuni metodi per ovviare a questo problema, utilizzando per esempio il protocollo di trasmissione TCP dove i pacchetti ricevuti a un nodo destinatario sono riconosciuti tramite un messaggio di ritorno al nodo sorgente e, se il nodo sorgente non riceve questo messaggio, ritrasmette il pacchetto. Un metodo alternativo è l'uso dell'*erasure coding*. Applicato a un nodo sorgente, introduce un grado di ridondanza ai pacchetti in modo tale che il messaggio può essere recuperato anche se solo un sottoinsieme di pacchetti inviati dalla sorgente sono ricevuti dal destinatario.

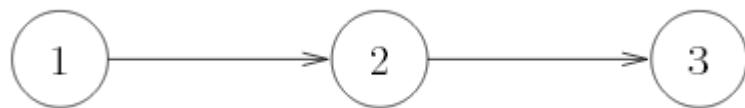


Figura 2.3. Nodo 1 e 2 inviano un singolo pacchetto per unità di tempo.

Considerando la rete in Figura 2.3, i pacchetti si perdono tra i nodi 1 e 2 con probabilità  $P_{12}$  e tra 2 e 3 con probabilità  $P_{23}$ . Con l'*erasure coding* applicato al nodo 1 avremo un rate di informazione di  $(1-P_{12})(1-P_{23})$  pacchetti per unità di tempo con una capacità di  $(1-P_{12})(1-P_{23})$ , ma la capacità del sistema è molto più grande.

Applicando l'erasure coding tra i nodi 1-2 e 2-3 si avrà un rate di 1-P12 e 1-P23 rispettivamente per unità di tempo.

Quindi questa soluzione non viene usata nei pacchetti della rete a causa del ritardo. Ogni stage dell'erasure coding introduce un certo grado di ritardo causato dalla decodifica e se applicato ad ogni connessione tra i nodi questo ritardo può essere grande. Aggiungendo una fase extra di erasure coding alla rete, questa diventa una speciale forma di network coding. Quindi NC può essere usato per aumentare la robustezza della rete.

- **ROBUSTEZZA A ERRORI DI COLLEGAMENTO**

NC è anche robusta contro i guasti di alcuni collegamenti sulla rete senza l'utilizzo di un *rerouting*.

### **2.1.3 Teorema principale di una rete Multicast**

Con una rete Unicast, solo un destinatario riceverà il messaggio in un determinato tempo, mentre con una rete Multicast una stessa informazione può essere mandata a più destinatari.

Il teorema principale del NC [3], provato da Ahlswede *et al.* [1], dimostra che le condizioni necessarie per garantire una comunicazione multicast con un certo rate per ogni ricevitore sono le stesse per una comunicazione unicast poiché i nodi intermedi possono combinare le informazioni ricevute.

#### **2.1.3.1 Min-cut Max-flow**

Il teorema del Minimo-Taglio Massimo-Flusso fu provato da Menger [6] e successivamente dimostrato da altri come Ford-Fulkerson [4]. In questa tesi il teorema sarà spiegato avendo per riferimento [3].

Considerato una rete  $\mathbf{G} = (V, E)$ , con un insieme di nodi  $V$  e collegamenti  $E$ , assumendo capacità unitaria di ogni collegamento con un nodo sorgente (  $S$  ) e uno destinatario (  $R$  ).

Si definisce *taglio*, tra  $S$  e  $R$ , un insieme di collegamenti che non possono essere rimossi senza disconnettere  $S$  da  $R$ , e il più piccolo valore (la somma delle capacità dei collegamenti nel taglio) si chiama *min-cut*. Da questo teorema si deduce che il massimo rate di informazione inviato da  $S$  a  $R$  è uguale al valore del min-cut.

Quindi se definiamo  $h = \text{min-cut}$  possiamo dire che esisteranno  $h$  collegamenti-percorsi disgiunti tra  $S$  e  $R$ , come in Figura 2.4.

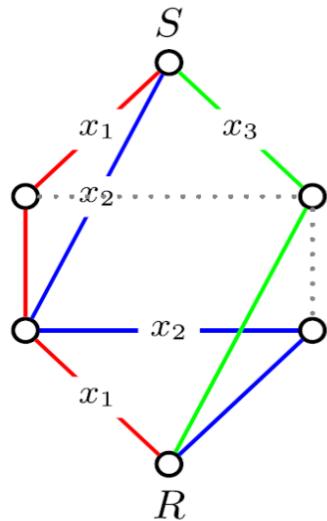


Figura 2.4. Connessione Unicast in una rete con capacità collegamenti unitaria[3].

Dalla topologia della rete di Figura 2.5 possiamo concludere che il valore min-cut di questa rete è uguale a 3 poiché esistono 3 collegamenti-percorsi disgiunti tra **S** e **R** che inviano i simboli  $x_1$ ,  $x_2$  e  $x_3$  al ricevitore **R**.

## 2.2 Linear Network Coding

E' stato visto come nel NC i nodi intermedi della rete combinano un numero di pacchetti ricevuti in uno o più pacchetti che inviano a loro volta ad altri nodi. Il *Linear Network Coding* ( LNC ) [2, 7], permette invece di operazioni nel campo binario, l'uso di un campo più grande applicando operazioni più complesse quando si combinano i pacchetti nei nodi intermedi.

In LNC ogni dato è processato utilizzando un campo finito  $GF(p)$  con  $p$  numero primo o, considerando un *Campo di Galois* ( GF ),  $p = 2^q$  con  $m$  intero, dove GF si riferisce a elementi da  $[0, 2^{(q-1)}]$ .

Presa una rete  $G = (V, E)$ , la rete a farfalla è ancora una volta un buon esempio per capire come LNC funziona ( Figura 2.5 ).

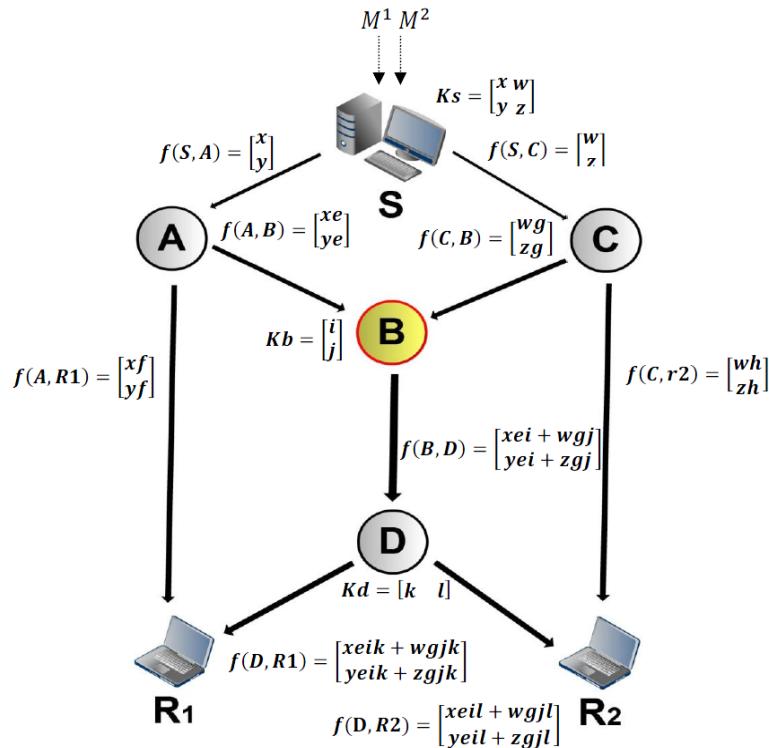


Figura 2.5. LNC:Local/global coding vector di un linear network coding 2-dimensionale

### 2.2.1 Codifica

Come visto in Figura 2.5, abbiamo  $M^1, \dots, M^k$  pacchetti originali generati da una sorgente  $S$ , dove  $k=2$  ( linear network code 2-dimensionale ). Ad ogni pacchetto è associato una sequenza di *encoding vectors*  $g_1, \dots, g_k$  nel campo  $GF(p)$  e uguali a un *information vector*,

$$Y = \sum_{i=1}^n g_i M^i$$

essendo la sommatoria eseguita in ogni posizione di simbolo (su un campo finito  $GF(p)$  ) allora,

$$Y_{(n)} = \sum_{i=1}^k g_i M_n^i$$

dove  $Y_n$  e  $M_n^i$  è il  $n$ -esimo simbolo di  $Y$  e  $M$ .

Ogni nodo quindi creerà un *information vector* e un *global encoding vector* (generato da un *local encoding vector*) in base al numero dei suoi collegamenti in ingresso ed in uscita e invierà la relativa riga al relativo nodo successivo.

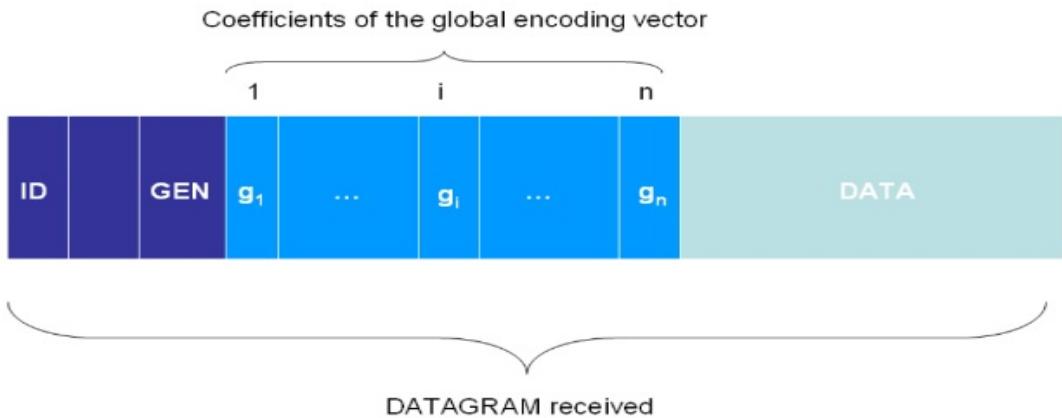


Figura 2.6. Pacchetto ricevuto da un nodo

In Figura 2.6 è possibile vedere l'Header di un pacchetto codificato con LNC, costituito dall'information vector e dall'encoding vector. Il nodo a sua volta può modificare tale pacchetto applicando le regole del LNC ed inviarlo a un nodo successivo.

In pratica, definito  $K=numero\ di\ messaggi$ ,  $l=numero\ di\ collegamenti\ uscenti\ dal\ nodo$ ,  $m=numero\ di\ collegamenti\ in\ ingresso\ al\ nodo$ ,  $n=numero\ nodi\ rete$ ,  $N=nodo\ corrente$ , allora ogni nodo della rete genera una matrice chiamata **local encoding vector**  $g_{(l \times m)}^{loc}$  tramite il quale genera un **global encoding vector**  $G_{(l \times k)}^N$  :

$$G_{(l \times k)}^N = g_{(l \times m)}^{loc} \cdot G_{(m \times k)}^{prec}$$

Gli elementi di queste matrici fanno parte del campo finito scelto.

L'***information vector*** di ogni nodo invece è così creato :

$$Y_{(l \times 1)}^N = g_{(l \times m)}^{loc} \cdot Y_{(m \times 1)}^{prec}$$

Come è possibile notare ogni riga delle matrici del *global encoding vector* e dell'*information vector*, viene inviata a un collegamento in uscita dal nodo.

I pedici delle seguenti formule si riferiscono alla dimensione della matrice, mentre gli apici indicano se la matrice è generata localmente dal nodo (*loc*) oppure se è costituita da vettori provenienti dai collegamenti in ingresso al nodo (*prec*) oppure indicano la matrice del nodo corrente (*N*).

Nel caso del nodo sorgente S, questo si comporterà come se “ricevesse” da dei canali immaginari i pacchetti originali quindi avremo:

$$Y^{prec} = M_{(k \times 1)}$$

$$G^{prec} = [0]$$

Quindi l'*information vector* ricevuto dal nodo S sarà il vettore costituito dai messaggi originali M.

Aggiornando la Figura 2.5, usando il campo associato  $\text{GF}(2) = \{0,1\}$  con  $M^1 = a$  e  $M^2 = b$  avremo il risultato in Figura 2.7.

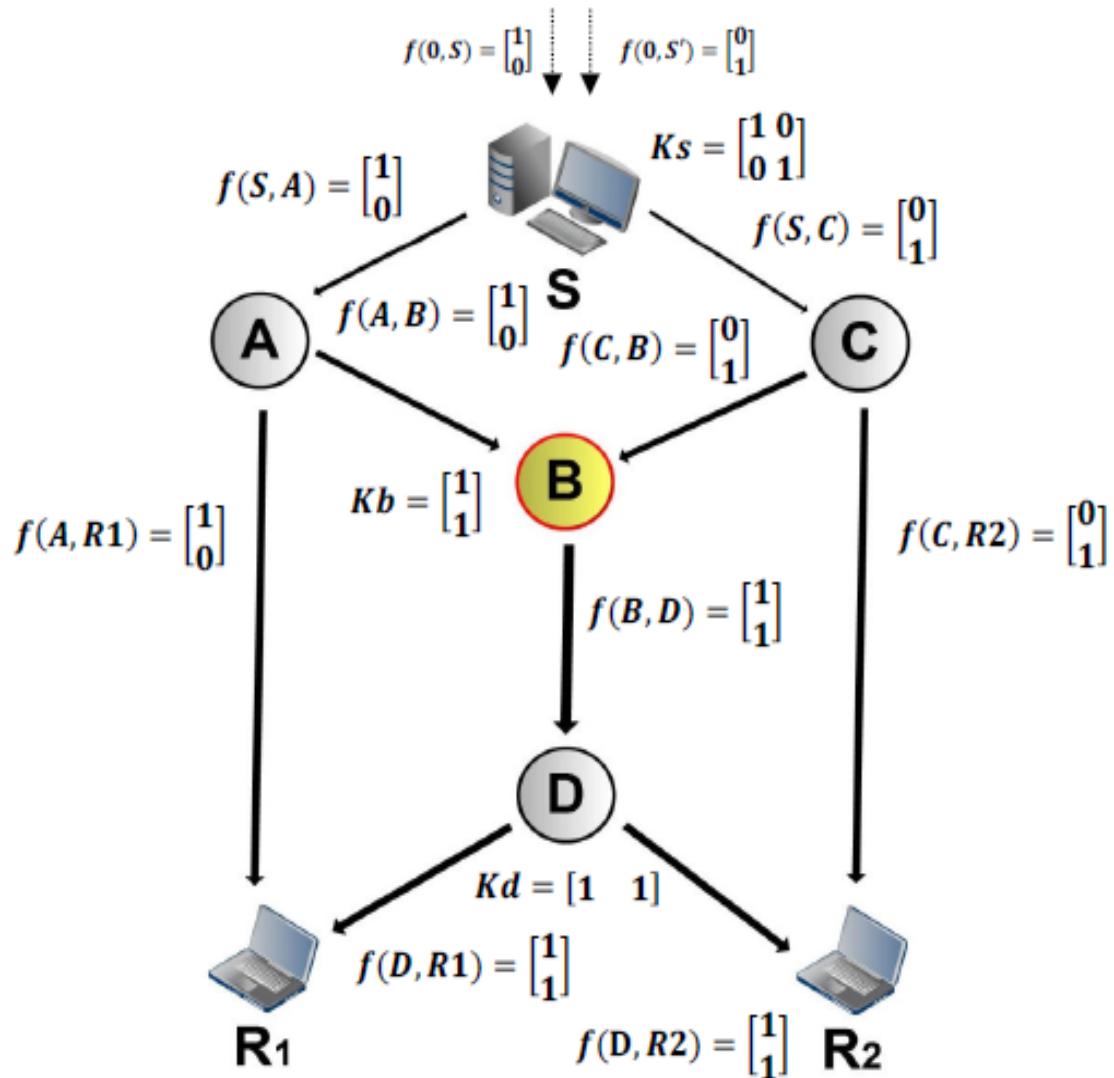


Figura 2.7 LNC:local/global encoding vector aggiornati

Con LNC, addizione e sottrazione vengono eseguite su un campo finito GF dove i pacchetti risultanti sono combinazione lineare degli originali con la stessa lunghezza di quelli originali.

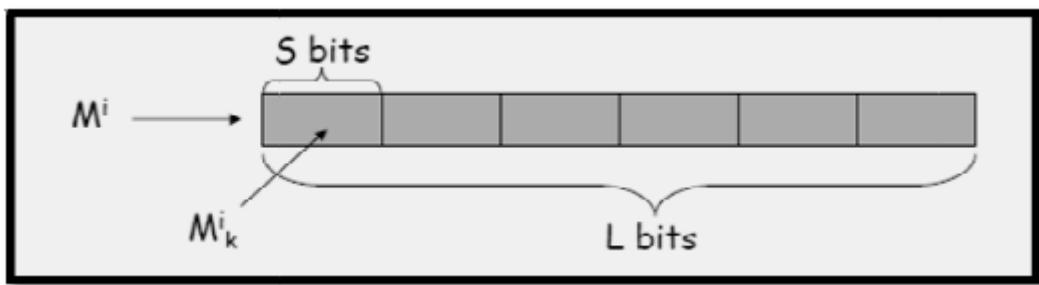


Figura 2.8. Formato pacchetto NC. Adattato da [8]

In Figura 2.8 è illustrato il formato di un pacchetto NC dove L sta per la lunghezza di ogni pacchetto e S per S bits di  $M_k^i$ .

## 2.2.2 Decodifica

In LNC, i pacchetti originali ( $M^1, \dots, M^k$ ) sono ricavati dal nodo ricevente tramite una semplice eliminazione Gaussiana.

Il pacchetto codificato visto in precedenza trasporta due tipi di informazioni: **encoding vector**  $g_1, \dots, g_k$  e **information vector**  $Y$ . Quando un nodo (ricevente) riceve  $l$  numero di  $Y$  ( $l \geq k$ ) può decodificare questo e ricavare i pacchetti originali.

Il nodo ricevente per ricavare i pacchetti dovrà risolvere il seguente sistema (2.1):

$$\begin{pmatrix} Y_1^{prec} \\ \dots \\ Y_l^{prec} \end{pmatrix} = \begin{pmatrix} g_{(1,1)} & \dots & g_{(1,k)} \\ \dots & \dots & \dots \\ g_{(l,1)} & \dots & g_{(l,k)} \end{pmatrix} \begin{pmatrix} M^1 \\ \dots \\ M^k \end{pmatrix} = G_{(l \times k)}^{(prec, tot)} \begin{pmatrix} M^1 \\ \dots \\ M^k \end{pmatrix} \quad (2.1)$$

Questo è un sistema lineare con  $k$  equazioni e  $l$  incognite, quindi il rango della matrice  $G_{(l \times k)}^{(prec, tot)}$  nella (2.1) deve essere  $k$ , in altre parole i vettori devono essere linearmente indipendenti.

Un efficiente schema di network coding mi permette di codificare i pacchetti in maniera tale che la probabilità di ricevere pacchetti linearmente dipendenti è molto bassa. Un modo semplice per ottenere ciò è nella scelta dei coefficienti nel campo finito  $\text{GF}(2^q)$ .

Risolvendo il sistema possiamo ricavare i pacchetti nel modo seguente:

$$M^{tot} = (G^{(prec, tot)})^{-1} \cdot Y^{(prec, tot)}$$

Condizione importante è che la dimensione di ogni messaggio inviato alla rete deve essere di dimensioni minore o uguale al numero di canali in ingresso al nodo destinatario.

In pratica se ho un messaggio  $M = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ , per poter ricavare il messaggio, il

nodo destinatario dovrà avere almeno 3 canali in ingresso.

Inoltre è possibile che il mio nodo ricevi un pacchetto ridondante che trasporta un'informazione già ricevuta. In questo caso il pacchetto non incrementa il rango della matrice G e quindi verrà scartato (*Non innovative packet*) a differenza di pacchetti che trasportano nuove informazioni (*Innovative packet*).

### 2.2.3 Campo di Galois

Il campo di Galois è un particolare campo di elementi finito. Definito come  $GF(p^q)$  con  $p$  numero primo esteso da  $GF(p)$ , è composto da tutte le  $m$ -ple ( polinomi di grado  $m-1$  ) di elementi di  $GF(p)$ .

L'operatore somma si ottiene applicando elemento per elemento la somma in  $GF(p)$  ( somma classica modulo  $p$ ), mentre l'operatore prodotto si può definire come prodotto classico tra i polinomi, con risultato limitato modulo  $p(x)$ , dove  $p(x)$  è un polinomio di grado  $q$  irriducibile, con coefficienti in  $GF(p)$ , detto “polinomio generatore del campo”.

In qualsiasi campo  $GF(p)$  esiste almeno un elemento  $a$  detto primitivo, che con le sue potenze da 0 a  $p-2$ , assume tutti i valori possibili del campo tranne 0 e li ripete poi ciclicamente con periodo  $p-1$ , per cui  $a^{(p-1)} = a^0 = 1$  .

Per completezza di rappresentazione si può porre  $a^{-\infty} = 0$  . Gli elementi di un campo esteso  $GF(p^q)$  si possono rappresentare in due modi:

- come polinomi di grado  $q-1$  con coefficienti in  $GF(p)$  ( o  $m$ -ple di elementi  $GF(p)$ ).
- come potenze  $k$ -sim e di un elemento primitivo:  $a^k$  con  $k = -\infty, 0, 1, \dots, p^q - 2$ .

La prima rappresentazione è la più comoda per l'applicazione dell'operatore somma, mentre la seconda si presta meglio alle operazioni di moltiplicazione grazie alle proprietà delle potenze per cui  $a^k \cdot a^j = a^{((j+k) \bmod (q-1))}$  .

I campi più interessanti sono i GF( $2^q$ ), i cui elementi non altro che m-ple di bit(byte). Interessano in quanto partendo dalle proprietà dell'algebra dei campi è possibile progettare delle tecniche di codifica e decodifica per la protezione degli errori di blocchi di bit o nelle tecniche del NC.

Un esempio di GF(2) è dato dalla Tabella 2.2.

<b>+</b>	<b>0</b>	<b>1</b>
<b>0</b>	0	1
<b>1</b>	1	0
<b>x</b>	<b>0</b>	<b>1</b>
<b>0</b>	0	0
<b>1</b>	0	1

Tabella 2.2 Operazioni somma e prodotto di GF(2).

Quando  $q \geq 1$ , GF( $p^q$ ) può essere rappresentato come un campo di classi di polinomi con coefficienti di GF(p). Ogni polinomio irriducibile di grado p rende lo stesso campo a un isomorfismo. Per esempio, per GF( $2^3$ ), il modulo può essere preso come  $x^3 + x^2 + 1$  o  $x^3 + x + 1$ . Usando il modulo  $x^3 + x + 1$ , gli elementi di GF( $2^3$ ) quali 0,  $x^0, x^1, \dots$  possono essere rappresentati come polinomi con grado minore di 3.

La Tabella 2.3 contiene differenti rappresentazioni di elementi di un campo finito. Le colonne sono potenza, rappresentazione polinomiale, rappresentazione vettoriale e l'intero binario che corrisponde alla rappresentazione vettoriale.

potenza	Rappr. polinomiale	Rappr. vettoriale	Intero binario
0	0	000	0
$x^0$	1	001	1
$x^1$	$x$	010	2
$x^2$	$x^2$	100	4
$x^3$	$x+1$	011	3
$x^4$	$x^2+x$	110	6
$x^5$	$x^2+x+1$	111	7
$x^6$	$x^2+1$	101	5

Tabella 2.3 Rappresentazioni differenti di elementi di un campo finito.

I codici lineari sono ottimali in una rete multicast con una sorgente S con una dimensione del campo di Galois sufficientemente grande[13].

La Tabella 2.4 e 2.5 sono un esempio delle operazioni su GF(  $2^2$  ) e GF(  $2^3$  ) rispettivamente.

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$x$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Tabella 2.4 Somma e prodotto di GF( $2^2$ ). Polinomio irriducibile è  $p(x) = x^2 + x + 1$

+	0	1	2	3	4	5	6	7
0	0	1	2	3	0	1	2	3
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Tabella 2.5 Somma e prodotto di GF( $2^3$ ).Polinomio irriducibile è  $p(x) = x^3 + x + 1$

## 2.2.4 Eliminazione Gaussiana

L'eliminazione Gaussiana viene utilizzata per facilitare la risoluzione del sistema (2.1) in fase di decodifica quando si hanno molte equazioni.

Esso è un algoritmo usato in algebra lineare per determinare le soluzioni di un sistema di equazioni lineari, per calcolare il rango o l'inversa di una matrice. L'algoritmo, attraverso l'applicazione di operazioni elementari dette *mosse di Gauss*, riduce la matrice in una forma detta a scalini. La matrice così ridotta permette il calcolo del rango della matrice (che sarà uguale al numero di scalini/pivot) nonché la risoluzione del sistema lineare ad essa associato. Un'estensione a tale metodo, nota come *metodo di eliminazione di Gauss-Jordan* riduce ulteriormente la matrice permettendo il calcolo dell'inversa.

Il primo passo è quello di mettere in matrice il sistema di equazioni lineari nel seguente modo:

$$\left\{ \begin{array}{l} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = b_2 \\ \dots \\ a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n}x_n = b_m \end{array} \right.$$

La matrice sarà:

$$\begin{pmatrix} a_{1,1} & \dots & a_{1,n} & b_1 \\ \dots & \dots & \dots & \dots \\ a_{m,1} & \dots & a_{m,n} & b_m \end{pmatrix}$$

Detta *matrice completa dei coefficienti del sistema*. I coefficienti del sistema lineare (e quindi della matrice) sono elementi di un campo  $K$ , quale ad esempio quello dei numeri reali  $\mathbf{R}$  o complessi  $\mathbf{C}$  (nel nostro caso  $\mathbf{GF}(p)$ ). L'ultima colonna è la **colonna dei termini noti**.

Le mosse di Gauss sono operazioni che modificano una matrice in uno dei seguenti modi:

- scambiando due righe;
- moltiplicando una riga per un numero diverso da zero;
- sommando una riga ad un multiplo di un'altra riga.

Le mosse di Gauss hanno la seguente importante proprietà: se applicate alla **matrice completa dei coefficienti** di un sistema lineare, non modificano lo spazio delle soluzioni del sistema. In altre parole, cambia il sistema ma le soluzioni restano invariate: un vettore  $(x_1, \dots, x_n)$  è soluzione del vecchio sistema se e solo se lo è del nuovo.

### 2.2.4.1 Algoritmo di Gauss

L'algoritmo di Gauss trasforma una qualsiasi matrice in una matrice a scalini tramite mosse di Gauss. Funziona nel modo seguente:

1. Se la prima riga ha il primo elemento nullo, scambiala con una riga che ha il primo elemento non nullo. Se tutte le righe hanno il primo elemento nullo, vai al punto 3.
2. Per ogni riga  $A_i$  con primo elemento non nullo, eccetto la prima ( $i > 1$ ), moltiplica la prima riga per un coefficiente scelto in maniera tale che la somma tra la prima riga e  $A_i$  abbia il primo elemento nullo (quindi coefficiente =  $-A_{i1}/A_{11}$ ). Sostituisci  $A_i$  con la somma appena ricavata.
3. Adesso sulla prima colonna tutte le cifre, eccetto forse la prima, sono nulle. A questo punto ritorna al punto 1 considerando la sottomatrice che ottieni cancellando la prima riga e la prima colonna.

Il risultato dell'algoritmo non è sempre lo stesso, dipende dalle scelte effettuate. Durante lo svolgimento è importante avere in mente l'obiettivo finale cioè ottenere una matrice a scala come questa:

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -6 & 5 \\ 0 & 0 & -2 \end{pmatrix}$$

#### 2.2.4.2 Algoritmo di Gauss-Jordan

Dopo aver ridotto la matrice a scalini, è possibile usare una versione dell'algoritmo di Gauss in senso inverso, cioè dal basso verso l'alto, per ottenere una matrice che in ogni colonna contenente un pivot abbia solo il pivot come numero non nullo, (questa matrice risultante è anche detta matrice a scalini in forma ridotta): basta usare ogni riga, partendo dall'ultima, per eliminare tutte le cifre diverse da zero che stanno sopra al pivot di questa riga. Infine, sempre con mosse di Gauss (moltiplicando righe), possiamo ottenere che ogni pivot abbia valore 1.

Ad esempio, portando avanti l'algoritmo descritto sopra otteniamo:

$$\begin{pmatrix} -1 & 1 & 0 \\ 0 & -6 & 0 \\ 0 & 0 & -4 \end{pmatrix} \rightarrow \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -4 \end{pmatrix} \rightarrow \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -4 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

L'algoritmo di Gauss-Jordan trasforma la matrice dei coefficienti di un sistema in una matrice fatta nel modo seguente: le variabili corrispondenti alle colonne che non contengono pivot sono dette *libere*; ciascuna altra variabile compare in una sola equazione, e quindi può essere espressa in funzione delle variabili libere e dei termini noti. Lo spazio delle soluzioni si ottiene assegnando valori arbitrari alle variabili libere, e calcolando le altre variabili di conseguenza.

## 2.3 Random Linear Network Coding

RLNC fu proposto da T. Ho et al.[9]. La principale differenza tra RLNC e LNC è la selezione casuale delle combinazioni lineari che ogni nodo esegue su un campo GF(p).

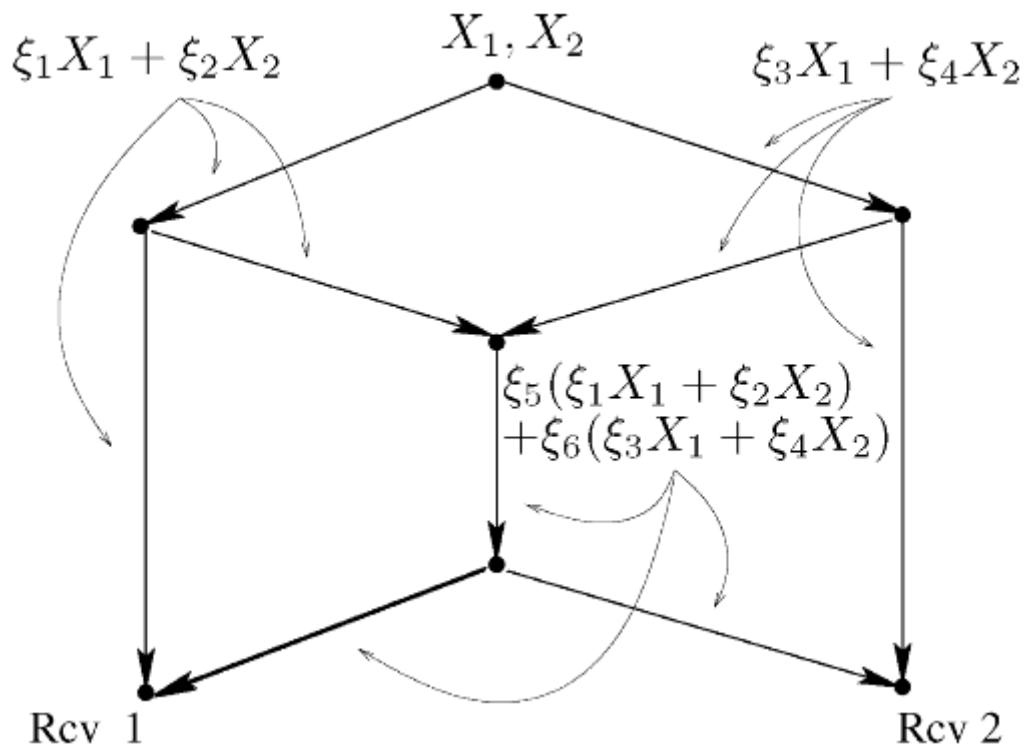


Figura 2.9. Esempio di RLNC

In pratica tutti i nodi tranne i ricevitori creano dei nuovi pacchetti facendo le combinazioni lineari casuali dei pacchetti in ingresso su di un campo finito. Il primo risultato che può produrre questa tecnica è la capacità multicast con probabilità esponenziale che si avvicina a 1. Inoltre in un contesto di un

problema di codifica di una sorgente distribuita, si dimostra che RLNC permette la compressione quando necessario in una rete, generalizzando gli esponenti degli errori noti per la codifica lineare Slepian-Wolf[15] in modo naturale. Questo approccio non solo recupera la capacità ma offre numerosi vantaggi.

Con RLNC, l'*encoding vector* visto in precedenza è costituito da coefficienti scelti da ogni nodo in maniera casuale e uniforme, in questo modo affinchè le combinazioni lineari siano indipendenti e quindi non si hanno errori nella rete, si deve usare un campo finito GF( $p$ ) sufficientemente grande.

Nel caso in cui i coefficienti vengono scelti in modo casuale, la probabilità di ricevere una combinazione lineare dipendente è legata al campo di Galois[16]. Il risultato delle simulazioni [17,18] ci dice che la probabilità diventa trascurabile anche per campi di piccole dimensioni come per  $q=8$ . Inoltre RLNC permette ai nodi di operare in modo completamente indipendente e decentralizzato, ottimo per le reti wireless.

Quindi con RLNC la probabilità di selezionare combinazioni linearmente dipendenti è almeno:

$$\left(1 - \frac{d}{q}\right)^v \quad \text{per } q > d$$

dove  $d$  è il numero di ricevitori,  $q$  la dimensione del campo di Galois e  $v$  il numero massimo di canali che ricevono pacchetti con coefficienti indipendenti casuali[9].

Più grande è  $q$  e più robusto è il sistema, mentre  $d$  e  $v$  meglio se piccoli.

## **2.4 Applicazioni Network Coding**

il NC ha numerosi utilizzi nelle seguenti aree:

- alternativa a forward error correction e ARQ in reti wireless tradizionali con perdita di pacchetti. Esempio: Coded TCP[21], Multi-user ARQ[22].
- Robusto e resistente agli attacchi della rete come snooping, intercettazioni,replicazioni o dati corrotti[23,24].
- Distribuzione digitale di file e P2P file sharing. Esempio: Avalanche di Microsoft.
- Storage distribuito[25,26].
- Aumento throughput in reti wireless. Esempio: COPE[27],Coding-aware routing[28,29].
- Trasmissione a basso consumo bidirezionale in reti con sensori wireless.
- Aumento della capacità di una rete molti-a-molti broadcast.
- Buffer e riduzione del ritardo in reti di sensori: Spatial buffer multiplexing[30].
- Riduce il numero di pacchetti ritrasmessi e quindi migliora la larghezza di banda della rete[31].

### 2.4.1 Reti Ad-Hoc

Il significato letterale di *ad hoc* in Latino sta “per questo”, quindi una rete ad-hoc è una LAN ( Local area Network) costruita con dispositivi connessi tra di loro. Principalmente le reti ad-hoc, mobile e P2P possono essere migliorate con l'uso del NC.

### 2.4.2 P2P

Un famoso metodo di questo tipo è BitTorrent, dove ogni *peer* analizza e trasmette ogni tipo di file sulla rete, usando il protocollo BitTorrent. Un *peer* è un computer che esegue un'istanza di un client.

A.J. Gonzàlez [32] spinse l'interesse nell'utilizzo del NC nelle reti peer-to-peer (P2P).

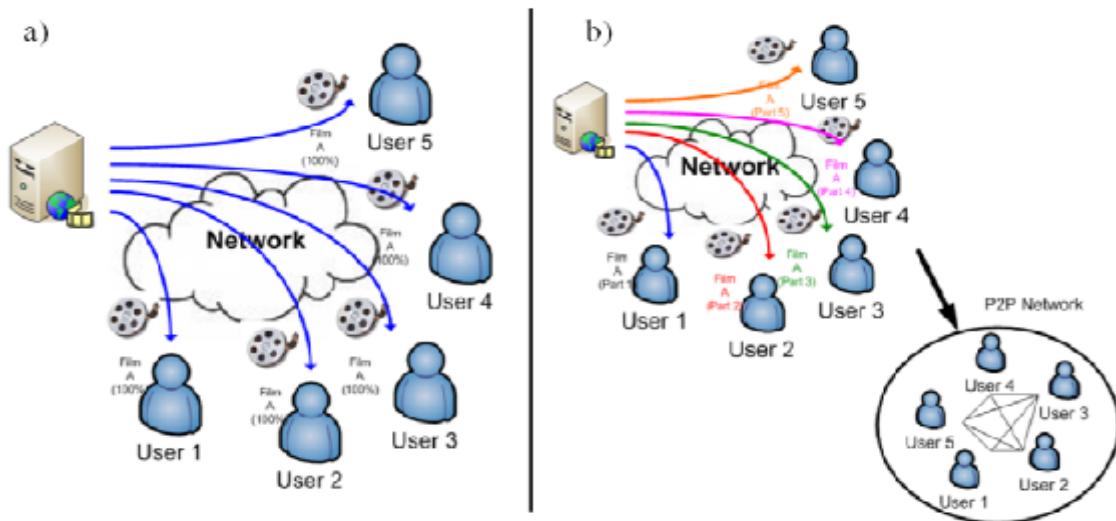


Figura 2.10 a) download completo-ogni utente scarica il file. b) download condiviso-ogni utente scarica una parte del file[32].

In queste reti si vide come il NC riusciva ad aumentare il throughput. Considerando uno streaming P2P con download condiviso, come illustrato in Figura 3.1, mandando una combinazione lineare di blocchi differenti del file originale che il peer desidera mandare nella rete, si ha un aumento del throughput. La ridondanza dell'informazione disponibile tra peers migliora l'uso delle caratteristiche del NC, aumentando il numero di possibili combinazioni lineari di pacchetti, risolvendo il problema della posizione dell'ultimo blocco mancante per completare il download.

Con l'aumento dell'efficienza della rete, si riduce la probabilità di perdita di informazione.

#### 2.4.3 Reti Wireless

Le due principali proprietà delle reti wireless [10] che differiscono da quelle classiche sono la possibilità di trasmettere in broadcast e anche l'interferenza. Le reti wireless sono costantemente affette da diverse problematiche quali un basso throughput e la morte di nodi di trasmissione.

Una ricerca sviluppata chiamata COPE project[11] dimostrano i benefici del NC nelle reti wireless, a livello MAC, anche usando una semplice codifica con codifica XOR, infatti per questo [11] è chiamato “*Xors in the air*”.

Ahlswede et al.[1], in un primo approccio alle reti wireless, ignorò l'interferenza su un ricevitore. Un esempio del livello fisico di una rete wireless con e senza NC è rappresentato dalla Figura 2.11.

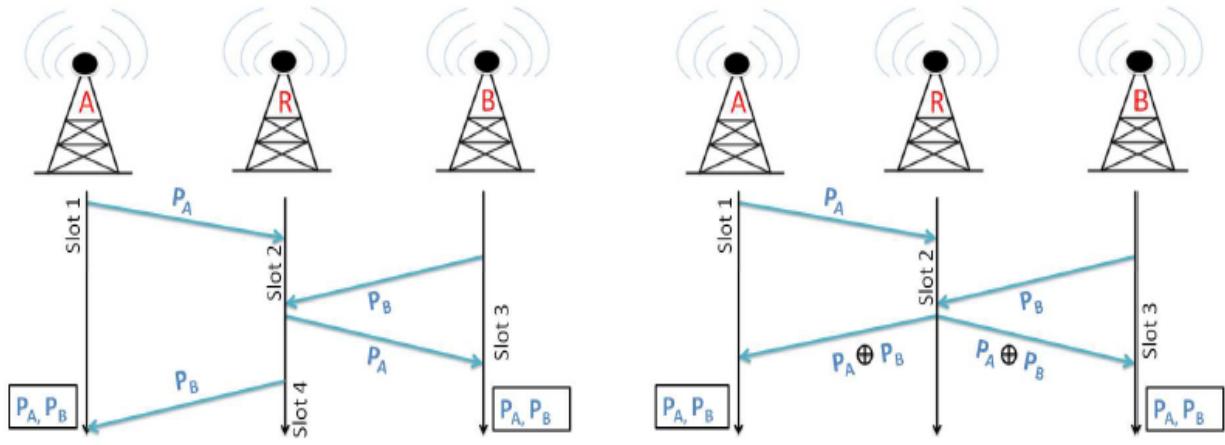


Figura 2.11 Esempio di rete ad-hoc wireless: *store-and-forward* contro *network coding*.

Nella Figura 2.11 vediamo una rete wireless costituita da due nodi **A** e **B** interessati a scambiarsi due messaggi  $P_a$  e  $P_b$ . La distanza tra i due è tale che non possono scambiarsi messaggi direttamente, quindi devono comunicare attraverso un altro nodo **R**.

Nella prima parte della figura vediamo il classico metodo di trasmissione store-and-forward dove durante i primi due slot di tempo i nodi **A** e **B** inviano i loro messaggi a **R**. Una volta che **R** ha ricevuto i messaggi, invia in uno slot di tempo il messaggio  $P_a$  e in un altro slot il messaggio  $P_b$  ai rispettivi ricevitori (che sono i nodi **A** e **B**) per un totale di 4 slot di tempo.

Nella seconda parte invece vediamo il metodo di trasmissione con l'uso del NC. In questo caso il nodo **R** una volta che ha ricevuto i due messaggi, invia lo **XOR** dei due pacchetti ricevuti al nodo **A** e **B** per un totale di 3 slot di tempo.

Questo esempio mostra come il network coding è particolarmente utile quando si sovrappongono flussi di dati.

### 2.4.3.1 Panoramica COPE

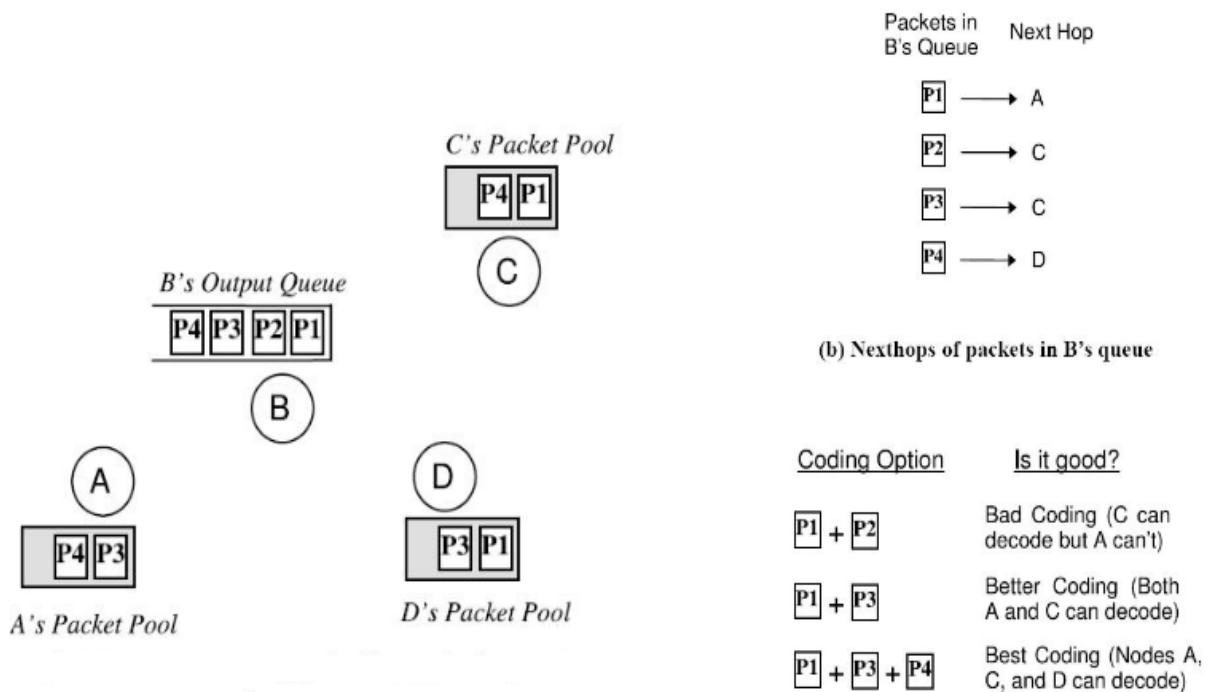


Figura 2.12 Esempio di Opportunistic Coding (COPE)

Nella Figura 2.12 vediamo un esempio di COPE. Il nodo B ha 4 pacchetti in coda che devono essere inviati secondo la lista in (b). Gli altri nodi (A-D-C) hanno già ricevuto e salvato alcuni pacchetti. Il nodo B può prendere alcune decisioni, ma dovrebbe scegliere l'ultima poiché è quella che massimizza il numero di pacchetti inviati in una singola trasmissione.

Questa architettura introdotta in [11] inserisce dei livelli di codifica tra i livelli IP e MAC per permettere l'invio multiplo di pacchetti in una singola trasmissione.

Definiamo come *coding gain* come il rapporto tra il numero di trasmissioni richieste in un approccio non-coding e il minimo numero di trasmissioni usando COPE.

Vedendo la Figura 2.12, COPE riduce il numero di trasmissioni da 4 a 3

producendo un coding gain di  $\frac{4}{3}=1,33$ .

Ma qual'è il massimo coding gain che è possibile avere? La capacità di una generica rete con NC per traffico unicast è ancora una questione aperta [33,34]. Comunque prendendo in esempio alcune reti si è visto che il massimo coding gain raggiungibile è 2.

# CAPITOLO 3

## Simulazione di una rete con MATLAB applicando RLNC

In questo capitolo verrà presentato il simulatore di una generica rete creato usando l'ambiente di sviluppo MATLAB, identificando sia gli aspetti positivi che quelli negativi nell'uso del RLNC.

### 3.1 MATLAB

**MATLAB** (abbreviazione di **M**atrix **L**aboratory) è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato dalla *MathWorks*. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi. Nonostante sia specializzato nel calcolo numerico, un toolbox opzionale interfaccia MATLAB con il motore di calcolo simbolico di *Maple*. MATLAB è usato da milioni di persone nell'industria e nelle università per via dei suoi numerosi tool a supporto dei più disparati campi di studio applicati e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, GNU/Linux e Unix[35].

### 3.1.1 Campo di Galois in MATLAB

MATLAB ha la capacità di generare campi di Galois (campi finiti) e lavorare con essi. Tutte le funzioni del GF si trovano all'interno del *Communications Toolbox* di MATLAB stesso.

La principale funzione utilizzata è **gf(X, q)**, che crea un array GF con  $2^q$  elementi, con  $q$  un intero compreso tra 1 e 16, da una matrice  $X$ . Il risultato sarà un array GF che MATLAB riconoscerà e lavorerà diversamente da un array di interi.

Nella Tabella 3.1 troviamo una lista dei polinomi primitivi che la funzione *gf* usa di default, per ogni  $\text{GF}(2^q)$  creato.

<b>q</b>	<b>Polinomio Primitivo di Default</b>	<b>Rappresentazione Intera</b>
1	$D+1$	3
2	$D^2+D+1$	7
3	$D^3+D+1$	11
4	$D^4+D+1$	19
5	$D^5+D^2+1$	37
6	$D^6+D+1$	67
7	$D^7+D^3+1$	137
8	$D^8+D^4+D^3+D^2+1$	285
9	$D^9+D^4+1$	529

10	$D^{10} + D^3 + 1$	1033
11	$D^{11} + D^2 + 1$	2053
12	$D^{12} + D^6 + D^4 + D + 1$	4179
13	$D^{13} + D^4 + D^3 + D + 1$	8219
14	$D^{14} + D^{10} + D^6 + D + 1$	17475
15	$D^{15} + D + 1$	32771
16	$D^{16} + D^{12} + D^3 + D + 1$	69643

Tabella 3.1 funzione *gf* MATLAB, polinomio promitivo di default.

Quando si usa la funzione *gf* su operazioni tra array, i semplici operatori “+”, “-” e “\*” possono essere usati. Un esempio è dato dal codice seguente che crea un array GF( $2^2$ ) con  $q=2$  e aggiunge un altro array GF con lo stesso campo:

```

x=0:3          %riga di vettori [0 1 2 3]
q=2            %GF(4)
a=gf(x,q)      %crea array GF in GF(4)
b=gf([0,1,3,1],q)
c=a+b          %crea c dalla somma di a e b
%l'output sarà -->
a = GF(2^2) array. Primitive polynomial = D^2+D+1 (7 decimal)
Array elements =
0 1 2 3
b = GF(2^2) array. Primitive polynomial = D^2+D+1 (7 decimal)
Array elements =
0 1 3 1
c = GF(2^2) array. Primitive polynomial = D^2+D+1 (7 decimal)
Array elements =
0 0 1 2

```

Facendo una semplice analisi, le uscite indicano i valori degli array GF  $a$ ,  $b$  e  $c$  indicando:

- il campo, GF( $2^2$ )=GF(4).
- Il polinomio primitivo per il campo, di default per GF(4).
- I valori che l'array GF contiene.
- Il comando che crea  $c$  mostra come, avendo definito la variabile  $b$  come array GF, si può sommare  $a$  a  $b$  usando l'operatore  $+$ . MATLAB farà l'operazione nello stesso campo GF(4).
- $C$  avrà lo stesso campo usato da  $a$  e lo stesso polinomio primitivo, infatti il secondo elemento di  $c$  è zero poiché la somma di qualsiasi valore con se stesso, in un campo di Galois con caratteristica 2, è zero.

### 3.2 Introduzione al simulatore

Il simulatore realizzato in MATLAB è costituito da due file “*.m*”, **RLNC.m** e **NODO.m**. Il primo m-file è uno script contenente delle serie di istruzioni che può essere richiamato da altri m-file o direttamente dalla Command Window. Il secondo m-file è invece una funzione che deve essere salvata col nome della funzione stessa.

Il compito del simulatore è aprire un qualsiasi file ( .png, .pdf, .txt, ecc), leggere i dati in esso contenuti e inserirli in un vettore colonna. Questo vettore colonna sarà il messaggio inviato alla rete secondo le tecniche del RLNC. Il nodo destinatario decodificherà i messaggi ricevuti e ricostruirà (se può) il messaggio originale.

Per quanto riguarda la rete, è stata presa come esempio la rete della Figura 3.4, una rete aciclica, direzionale, senza ritardi con capacità dei canali unitaria.

Verrà spiegato più avanti il perchè della disposizione numerica dei nodi, infatti il nodo 2 è il “reale” nodo sorgente della rete, mentre il nodo 7 è il nodo “reale” destinatario.

Il nodo 1 e 9 sono dei nodi immaginari poiché si suppone che il nodo sorgente ricevi i messaggi da dei canali “immaginari”(1) e il nodo destinatario invii i messaggi ricevuti decodificati in un canale “immaginario”(9).

La rete condivide 4 matrici (G\_TX/G\_RX/Y\_TX/Y\_RX). Queste matrici conservano i relativi information vector e encoding vector dei vari nodi.

$G_{TX}$  è la matrice che conserva gli encoding vector che vengono trasmessi.

$G_{RX}$  invece conserva gli encoding vector ricevuti.

Stessa cosa vale per  $Y_{TX}$  e  $Y_{RX}$ . Ogni riga di queste matrici corrisponde al relativo nodo, mentre le colonne corrispondono alle uscite del nodo. Per esempio il nodo 2 ha un ingresso e 4 uscite quindi avremo alla fine della computazione che  $G_{TX}\{2,1..4\}$  avrà dei vettori.

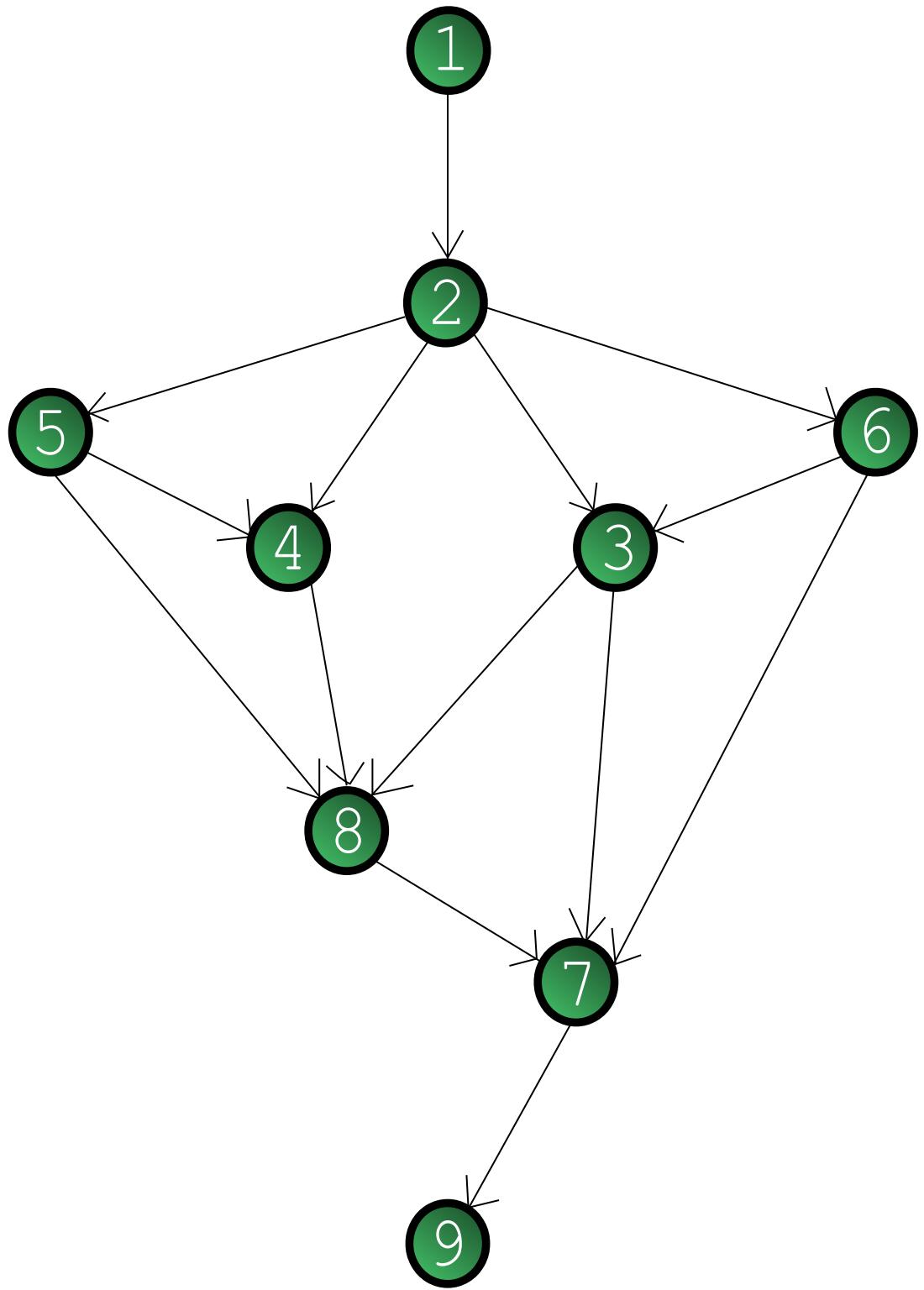


Figura 3.1 Rete utilizzata come esempio nella simulazione

### **3.3 File RLNC.m**

#### **3.3.1 Setup**

La prima parte del codice è dedicata alla configurazione della rete scelta.

```
%*****SETUP*****%  
  
M=[0,1,0,0,0,0,0,0,0;  
    0,0,1,1,1,1,0,0,0;  
    0,0,0,0,0,0,1,1,0;  
    0,0,0,0,0,0,0,1,0;  
    0,0,0,1,0,0,0,1,0;  
    0,0,1,0,0,0,1,0,0;  
    0,0,0,0,0,0,0,0,1;  
    0,0,0,0,0,0,1,0,0;  
    0,0,0,0,0,0,0,0,0];  
  
blocco=3;  
  
nodo_finale=7;  
  
q=10;  
  
n=size(M,1);  
  
canali_in=size(find(M(:,n-1)==1),1);
```

La matrice  $M$  è una matrice quadrata che permette al simulatore di capire *come* i nodi sono collegati tra di loro, infatti ogni riga/colonna corrisponde ad un nodo. Ragionando per righe, il valore 1 è dato quando vi è un collegamento uscente a quel nodo, diversamente ragionando per colonne, il valore 1 è dato per un collegamento entrante da quel nodo.

La variabile ***blocco*** indica quanti messaggi mando in un unica trasmissione. Nella scelta della dimensione del blocco bisogna stare attenti ad una condizione importante cioè questa dimensione deve essere di dimensioni minore o uguale al numero di canali in ingresso al nodo destinatario.

La variabile ***nodo\_finale*** indica il numero del nodo reale della rete.

La variabile ***q*** è l'intero compreso tra 1 e 16 per creare il campo di Galois.

La variabile ***n***, tramite la funzione *size(M,1)*, restituisce il numero di righe della matrice M, quindi indica il numero totale di nodi della rete.

La variabile ***canali\_in***, tramite la funzione *find* restituisce il numero di canali in ingresso al nodo reale tramite il calcolo degli “uno” della matrice M lungo la colonna relativa al nodo.

### 3.3.2 Inizializzazione

La fase successiva è caratterizzata semplicemente dall'inizializzazione delle variabili/matrici utilizzate in tutto il codice.

```
%****INIZIALIZZAZIONE****%
Z=[];
G_RX{n,n}=[];
Y_RX{n,n}=[];
G_TX{n,n}=[];
Y_TX{n,n}=[];
Z_finale=[];
s=1;
Xo=[];
A=[];
t=1;
l=0;
```

### 3.3.3 Creazioni messaggi

In questo blocco di istruzioni si sceglierà il file da inviare alla rete e che sarà codificato.

```
%*****MEX*****%
fid=fopen('windows.jpg');
fid2=fopen('windows.jpg');

x=fread(fid2);
dim=size(X,1);
```

Tramite la funzione *fopen('x.estensione')* si riesce ad aprire il file “x.estensione” che ci servirà per poterlo modificare/convertire oppure ottenere informazioni da esso.

La funzione *fread(y)* invece permette di leggere file binari da un file (in questo caso viene passato come argomento il file aperto precedentemente).

La variabile *fid* è uguale a *fid2*, solo che la prima viene utilizzata per leggere la matrice del messaggio da inviare mentre la seconda serve solo per specificare la dimensione totale del messaggio da inviare.

Infatti la variabile *dim* mi restituisce il numero di righe della matrice X (matrice del messaggio originale da inviare).

### 3.3.4 Invio

Questa fase è dedicata all'invio del messaggio originale sulla rete.

```
%*****INVIO****%
while( size(Z_finale,1)<dim )
    for nodo=n:-1:1
        [G_RX,Y_RX,G_TX,Y_TX,Z_finale,s,l,nodo_finale] =
NODO(G_RX,Y_RX,G_TX,Y_TX,M,Xo,nodo,q,blocco,n,s,Z_finale,fid,l,nodo_finale);
        if nodo~=nodo_finale && n
            for dest=1:l
                G_TX{nodo,dest}=G_RX{nodo,dest};
                Y_TX{nodo,dest}=Y_RX{nodo,dest};
                G_RX{nodo,dest}=[];
                Y_RX{nodo,dest}=[];
            end;
        end;
    end;
end;
```

Tutto il ciclo, grazie al **while**, viene ripetuto fin quando la matrice contenente il mio messaggio decodificato dal nodo destinatario (*Z\_finale*) è uguale alla dimensione del messaggio originale.

Il primo ciclo **for** è un ciclo decrescente dal nodo finale della rete (*n*) fino al primo (1) che esegue la funzione NODO definita nell'm-file NODO.m.

La condizione ***if*** insieme al secondo ciclo ***for*** fa in modo che se il nodo corrente è diverso dal *nodo finale* e dall'ultimo (*n*), Le informazioni contenute nelle matrici dell'encoding vector e dell'information vector di ricezione ( $G_{RX}, Y_{RX}$ ) vengono passate alle relative matrici di trasmissione ( $G_{TX}, Y_{TX}$ ) del nodo corrente.

In pratica, la funzione NODO viene eseguita ad ogni nodo partendo dall'ultimo. Quindi bisogna stare attenti nello scegliere adeguatamente la numerazione dei nodi poiché il problema che si può presentare consiste nel fatto che se viene processato un nodo che ha ad esempio 2 canali in ingresso ma riceve solo un messaggio si avrà errore. Inoltre è stata effettuata questa scelta in modo da inviare più messaggi contemporaneamente sulla rete.

Nell'esempio della rete di Figura 3.4 non è stato chiamato il nodo 7 come 8 poiché se fosse stato così il nodo invece di 3 messaggi ne avrebbe ricevuti 2 (solo quelli provenienti da 3 e 6).

### 3.3.5 Ricezione

L'ultima parte di codice del file RLNC.m consiste nel ricreare un file con la matrice decodificata dal nodo destinatario ( $Z_{finale}$ ).

Ovviamente ci si aspetta che la matrice dei messaggi originali sia uguale alla  $Z_{finale}$ .

```
%****RICEZIONE****%
fclose(fid);
fclose(fid2);

dest=fopen('data.jpg','w');
z=uint8(Z_finale.x)

fwrite(dest,z);
fclose(dest);
```

Quindi vengono chiusi i file utilizzati precedentemente per creare la matrice dei messaggi.

La variabile ***dest*** conterrà il nuovo file creato tramite la matrice  $Z_{finale}$  utilizzando la funzione ***fopen('nuovo\_file.estensione','w')***, che permette di aprire il file di nome “nuovo\_file.estensione” con permessi solo di scrittura.

La variabile ***z*** contiene la conversione degli elementi di  $Z_{finale}$ , da elementi nel campo GF a uint8, per poter creare il file con ***fread(dest,z)***.

### 3.3.6 Codice completo

```
%****SETUP****%
M=[0,1,0,0,0,0,0,0,0;
0,0,1,1,1,1,0,0,0;
0,0,0,0,0,0,1,1,0;
0,0,0,0,0,0,0,1,0;
0,0,0,1,0,0,0,1,0;
0,0,1,0,0,0,1,0,0;
0,0,0,0,0,0,0,0,1;
0,0,0,0,0,0,1,0,0;
0,0,0,0,0,0,0,0,0];

blocco=3;
nodo_finale=7;
q=16;
n=size(M,1);
canali_in=size(find(M(:,nodo_finale)==1),1);

%****INIZIALIZZAZIONE****
Z=[];
G_RX{n,n}=[];
Y_RX{n,n}=[];
G_TX{n,n}=[];
Y_TX{n,n}=[];
z_finale=[];
s=1;
Xo=[];
A=[];
t=1;
```

```

%*****CREAZIONE MESSAGGI*****%
fid=fopen('windows.jpg');
fid2=fopen('windows.jpg');

X=fread(fid2);
dim=size(X,1);

%*****INVIO*****%
while( size(Z_finale,1)<dim )
    for nodo=n:-1:1
        [G_RX,Y_RX,G_TX,Y_TX,Z_finale,s,l,nodo_finale] =
NODO(G_RX,Y_RX,G_TX,Y_TX,M,Xo,nodo,q,blocco,n,s,Z_finale,fid,l,nodo_finale);
        if nodo~=nodo_finale && n
            for dest=1:l
                G_TX{nodo,dest}=G_RX{nodo,dest};
                Y_TX{nodo,dest}=Y_RX{nodo,dest};
                G_RX{nodo,dest}=[];
                Y_RX{nodo,dest}=[];
            end;
        end;
    end;
end;

%*****RICEZIONE*****%
fclose(fid);
fclose(fid2);

dest=fopen('data.jpg','w');
z=uint8(Z_finale.x)

fwrite(dest,z);
fclose(dest);

```

## 3.4 File NODO.m

Questo m-file è una funzione che, come si è visto nello script RLNC.m, viene eseguita ad ogni nodo.

La funzione viene richiamata nello script nel modo seguente:

*[output]=nome\_funzione[input]*

```
[G_RX,Y_RX,G_TX,Y_TX,Z_finale,s,l,nodo_finale]=NODO(G_RX,Y_RX,G_TX,Y_TX,M,Xo,  
nodo,q,blocco,n,s,Z_finale,fid,l,nodo_finale)
```

### 3.4.1 Inizializzazione

```
%*****INIZIALIZZAZIONE****%  
nodi_destinatari=find(M(nodo,:)==1);  
nodi_precedenti=find(M(:,nodo)==1);  
l=size(nodi_destinatari,2); %uscite del nodo corrente  
m=size(nodi_precedenti,1); %ingressi del nodo corrente  
H=[];  
J=[];  
Z=[];  
o=0;
```

In questa fase, il nodo che sta eseguendo la funzione calcola tramite matrice  $M$  i nodi destinatari e i nodi precedenti e li mette in un vettore **nodi\_destinatari** e **nodi\_precedenti** rispettivamente.

### 3.4.2 Primo nodo (nodo immaginario)

```
%*****1 NODO*****%
if nodo==1
    Xo=fread(fid,blocco);
    G_RX{1,1}=[0];
    if isempty(Xo)==1
        Y_RX{1,1}=1234567890;
    elseif isempty(Xo)==0
        Y_RX{1,1}=Xo;
    end;
end;
```

In questa sezione, inizialmente si andrà a leggere un determinato numero di messaggi (determinati da *blocco*) dalla matrice che contiene i dati del file, inserendoli in una matrice *Xo* che sarà inviata sulla rete.

Ogniqualvolta il primo nodo esegue queste istruzioni, *Xo* avrà sempre lo stesso *blocco* ma leggerà i vettori successivi fino alla fine (End-Of-File).

Successivamente si controlla *Xo*, se è vuoto (quindi End-Of-File), grazie alla funzione *isempty(Xo)==1* possiamo indicare la fine della trasmissione impostando *Y\_RX{1,1}=1234567890* (numero arbitrario). Se invece *isempty(Xo)==0* allora si invia *Xo* impostando *Y\_RX{1,1}=Xo*.

L'*encoding vector* inviato sarà invece 0.

### 3.4.3 Secondo nodo (nodo reale)

```
%****2 NODO****%
if nodo==2
    if Y_RX{1,1}==1234567890
        %non fa nulla..
    elseif Y_RX{1,1}~=1234567890
        x=size(Y_RX{1,1},1);
        g_loc=gf(randi(2*q-1,[1,x]),q);
        for i=1:l
            G_RX{nodo,i}=g_loc(i,:);
        end;
        y_loc=g_loc*Y_RX{1,1};
        for i=1:l
            Y_RX{nodo,i}=y_loc(i,:);
        end;
    end;
```

Il secondo nodo controlla l'*information vector* di trasmissione relativo al primo nodo, se questo è costituito dalla condizione imposta per l'End-of-File del primo nodo ( $Y_RX\{1,1\}=1234567890$ ), allora non fa nulla. In caso contrario crea l'*encoding vector locale* e ad ogni uscita del nodo prende una riga di  $g\_loc(1x3)$  e la impone uguale a  $G_RX\{2,I\}$  e così via per le altre uscite.

Per il calcolo dell'*information vector* si fa lo stesso procedimento, solo che si crea un  $y\_loc$  dal prodotto dell'encoding vector locale e dall'information vector ricevuto (dal primo nodo).

### 3.4.4 Nodi intermedi e nodo destinatario

```
%****NODI INTERMEDI****%
elseif nodo~=1 && 2 && nodo_finale && n %nodi REALI
    for v=1:m
        b=1;
        Bb=0;
        a=nodi_precedenti(v,1);
        dim=size(G_TX,2);
        for e=1:n
            if isempty(G_TX{a,e})==1
                b=b+1;
                Bb=Bb+1;
                if Bb==dim
                    o=1;
                    break;
                end;
            elseif isempty(G_TX{a,e})==0
                b=e;
                break;
            end;
        end;
        if Bb==dim
            o=1;
            break;
        end;
        H=vertcat(H,G_TX{a,b});
        if a~=n-1
            G_TX{a,b}=[];
        end;
        b=1;
        Bb=0;
        a=nodi_precedenti(v,1);
```

```

dim=size(Y_TX,2);

for e=1:n
    if isempty(Y_TX{a,e})==1
        b=b+1;
        Bb=Bb+1;
        if Bb==dim
            o=1;
            break;
        end;
    elseif isempty(Y_TX{a,e})==0
        b=e;
        break;
    end;
end;

if Bb==dim
    o=1;
    break;
end;

J=vertcat(J,Y_TX{a,b});

if a~=n-1
    Y_TX{a,b}=[];
end;

if nodo==2 && 1
%non fa nulla..
elseif nodo==nodo_finale
    a=size(H,1);
    b=size(J,1);
    if a==0 && b==0
        %non fa nulla..
    elseif a==blocco && b==blocco
        if det(H)==0
            G_TX{nodo,1}=[1234;1234;1234];
            %errore matrice singolare!
        elseif det(H)~=0
            G_TX{nodo,1}=H\J
        end;
    end;
end;

```

```

    end;

    elseif a>blocco && b>blocco
        c=rank(H);
        g=[];
        y=[];
        for d=1:c
            N=H(d,:);
            E=J(d,:);
            g=vertcat(g,N);
            y=vertcat(y,E);
        end;
        G_RX{nodo,1}=g\y;
    elseif a<blocco && b>blocco
        c=rank(H);
        g=[];
        for d=1:c
            N=H(:,d);
            g=horzcat(g,N);
        end;
        G_RX{nodo,1}=g\J;
    end;

elseif nodo~=2 && 1 && nodo_finale && o==0
    g_loc=gf(randi(2*q-1,[1,m]),q);
    A=g_loc*H;
    B=g_loc*J;
    for i=1:l
        G_RX{nodo,i}=A(i,:);
        Y_RX{nodo,i}=B(i,:);
    end;
end;

```

Nei nodi intermedi si prendono gli *encoding vector* e *information vector* provenienti dai canali in ingresso e si salvano in una matrice **H** e **J**, rispettivamente. Se non ci sono encoding/information vector in ingresso

allora si impone  $o=1$ , in modo tale da indicare successivamente che il corrente nodo non deve fare nessuna computazione.

Successivamente, il compito del nodo è calcolarsi l'*encoding vector* e l'*information vector* da inviare ai nodi successivi. Questo viene fatto calcolando sempre il  $g\_loc$  in base a ingressi e uscite, il quale poi permetterà di trovare  $G_{RX}$  del nodo corrente tramite  $A=g\_loc*H$  (stessa cosa per l'information vector con  $B=g\_loc*J$ ).

Nel caso del nodo finale, questo prenderà gli *encoding* e gli *information vector* ricevuti risolvendo il sistema di equazioni per trovare i messaggi originali. Il codice controlla se la dimensione di questi vettori è uguale al “*blocco*”, in questo caso riuscirà tranquillamente a risolvere il sistema poiché supponiamo che il nodo ricevi  $n$  *innovative packets*.

Nel caso in cui il nodo ricevi un numero minore rispetto a “*blocco*” allora si calcola il rango della matrice degli *encoding vector* che sarà minore del numero di “*blocco*” e si andrà ad eliminare una colonna in modo tale da poter risolvere il sistema senza problemi. In questo caso si potranno ricavare solo n messaggi, dove n è il rango della matrice degli *encoding vector*.

Nel caso in cui invece il nodo riceve un numero maggiore di messaggi rispetto a “*blocco*”, allora sicuramente si avranno *non innovative packets*, quindi si andrà a calcolare sempre il rango della matrice degli *encoding vector* che dovrebbe essere uguale a blocco ed eliminare le restanti righe dalle due matrici (*encoding e information vector*).

### 3.4.5 Codice completo

```
function [G_RX,Y_RX,G_TX,Y_TX,Z_finale,s,l,nodo_finale] =
NODO(G_RX,Y_RX,G_TX,Y_TX,M,Xo,nodo,q,blocco,n,s,Z_finale,fid,l,nodo_finale)

    %*****INIZIALIZZAZIONE*****
    nodi_destinatari=find(M(nodo,:)==1);
    nodi_precedenti=find(M(:,nodo)==1);
    l=size(nodi_destinatari,2);
    m=size(nodi_precedenti,1);
    H=[];
    J=[];
    Z=[];
    o=0;
    %****1 NODO****
    if nodo==1
        Xo=fread(fid,blocco);
        G_RX{1,1}=[0];
        if isempty(Xo)==1
            Y_RX{1,1}=1234567890;
        elseif isempty(Xo)==0
            Y_RX{1,1}=Xo;
        end;
    end;
    %****2 NODO****
    if nodo==2
        if Y_TX{1,1}==1234567890
            %non fa niente..
        elseif Y_TX{1,1}~=1234567890
            x=size(Y_TX{1,1},1);
            g_loc=gf(randi(2*q-1,[1,x]),q);
            for i=1:l
                G_RX{nodo,i}=g_loc(i,:);
            end;
        end;
    end;
```

```

y_loc=g_loc*Y_RX{1,1};
for i=1:l
    Y_RX{nodo,i}=y_loc(i,:);
end;
end;

elseif nodo==n
    if isempty(G_TX{nodo_finale,1})==0
        Z{s,1}=G_TX{nodo_finale,1};
        G_TX{nodo_finale,1}=[];
        Z_finale=vertcat(Z_finale,Z{s,1});
        s=s+1;
    end;

%*****NODI INTERMEDI****%
elseif nodo~=1 && 2 && nodo_finale && n %nodi REALI
    for v=1:m
        b=1;
        Bb=0;
        a=nodi_precedenti(v,1);
        dim=size(G_TX,2);
        for e=1:n
            if isempty(G_TX{a,e})==1
                b=b+1;
                Bb=Bb+1;
                if Bb==dim
                    o=1;
                    break;
                end;
            elseif isempty(G_TX{a,e})==0
                b=e;
                break;
            end;
        end;
    end;

```

```

    if Bb==dim
        o=1;
        break;
    end;

H=vertcat(H,G_TX{a,b});
    if a~=n-1
        G_TX{a,b}=[];
    end;

b=1;
Bb=0;
a=nodi_precedenti(v,1);
dim=size(Y_TX,2);
for e=1:n
    if isempty(Y_TX{a,e})==1
        b=b+1;
        Bb=Bb+1;
        if Bb==dim
            o=1;
            break;
        end;
    elseif isempty(Y_TX{a,e})==0
        b=e;
        break;
    end;
end;
if Bb==dim
    o=1;
    break;
end;

J=vertcat(J,Y_TX{a,b});
    if a~=n-1
        Y_TX{a,b}=[];
    end;
end;

```

```

if nodo==2 && 1
    %non fa nulla..
elseif nodo==nodo_finale
    a=size(H,1);
    b=size(J,1);
    if a==0 && b==0
        %non fa nulla..
    elseif a==blocco && b==blocco
        if det(H)==0
            G_TX{nodo,1}=[1234;1234;1234];
            %errore matrice singolare!
        elseif det(H)~=0
            G_TX{nodo,1}=H\J
        end;
    elseif a>blocco && b>blocco
        c=rank(H);
        g=[];
        y=[];
        for d=1:1:c
            N=H(d,:);
            E=J(d,:);
            g=vertcat(g,N);
            y=vertcat(y,E);
        end;
        G_TX{nodo,1}=g\y;
    elseif a<blocco && b>blocco
        c=rank(H);
        g=[];
        for d=1:1:c
            N=H(:,d);
            g=horzcat(g,N);
        end;
        G_TX{nodo,1}=g\J;
    end;

```

```
elseif nodo~=2 && 1 && nodo_finale && o==0

g_loc=gf(randi(2*q-1,[1,m]),q);

A=g_loc*H;
B=g_loc*J;

for i=1:l
    G_RX{nodo,i}=A(i,:);
    Y_RX{nodo,i}=B(i,:);
end;

end;
```

### 3.5 Valutazione delle prestazioni

Riprendendo la rete a farfalla (Figura 2.1) si può vedere analiticamente come il NC influisce sull'invio di più messaggi contemporanei nella rete. La Figura 3.2 illustra i vari slot temporali (numero di trasmissioni) che la rete compie per ricevere più messaggi con NC. Supponendo che  $M1=[1;2]$   $M2=[3;4]$   $M3=[5;6]$   $M4=[7;8]$

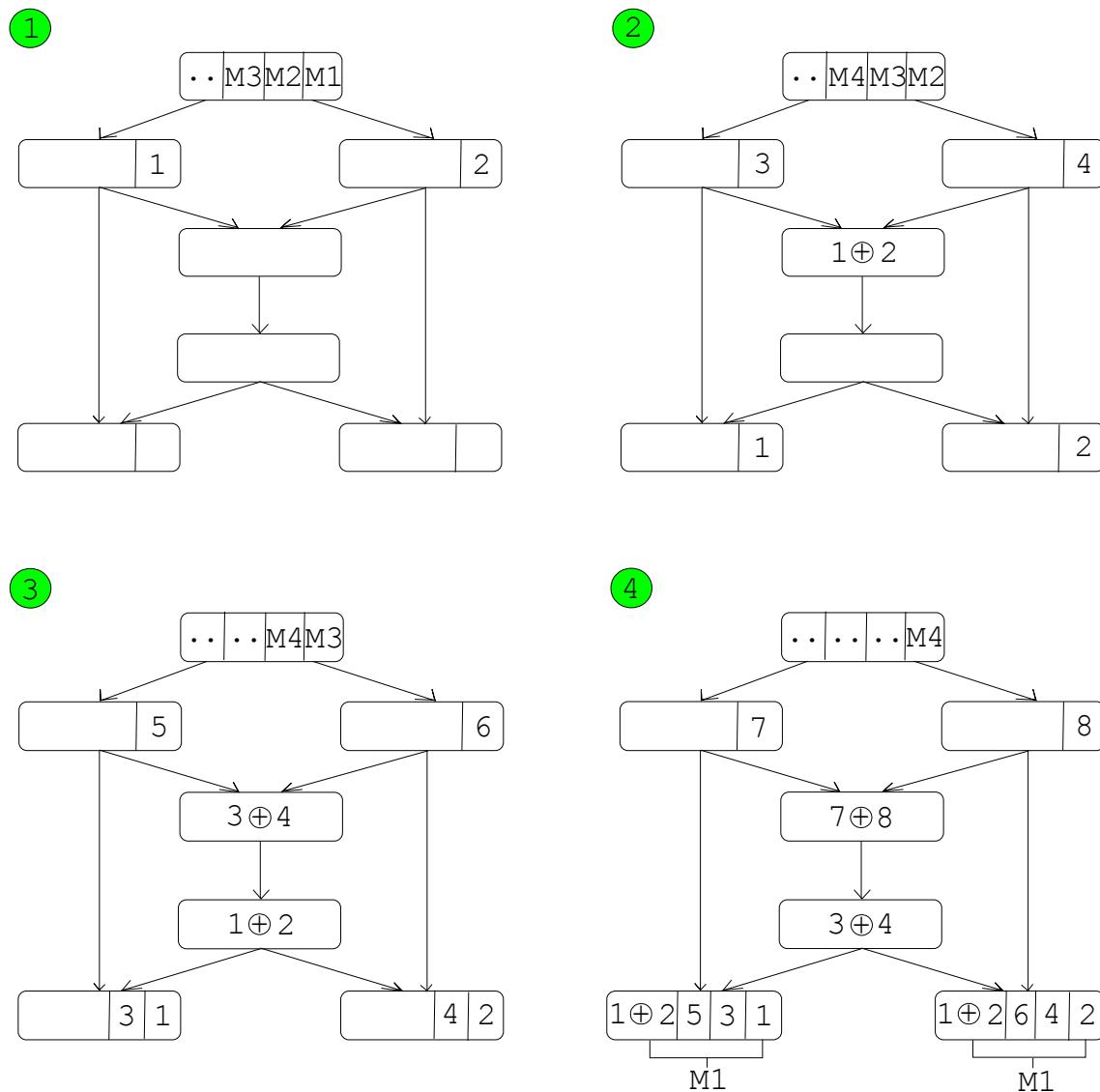


Figura 3.2 Numero di trasmissioni per ricevere un messaggio con rete a farfalla con NC

Come è possibile vedere il numero di trasmissioni necessario affinché i due nodi destinatari ricevano i primi due messaggi (primo blocco M1) è uguale a 4, diversamente come accade senza NC (Figura 3.3). (il primo slot viene omesso poiché uguale alla Figura 3.2).

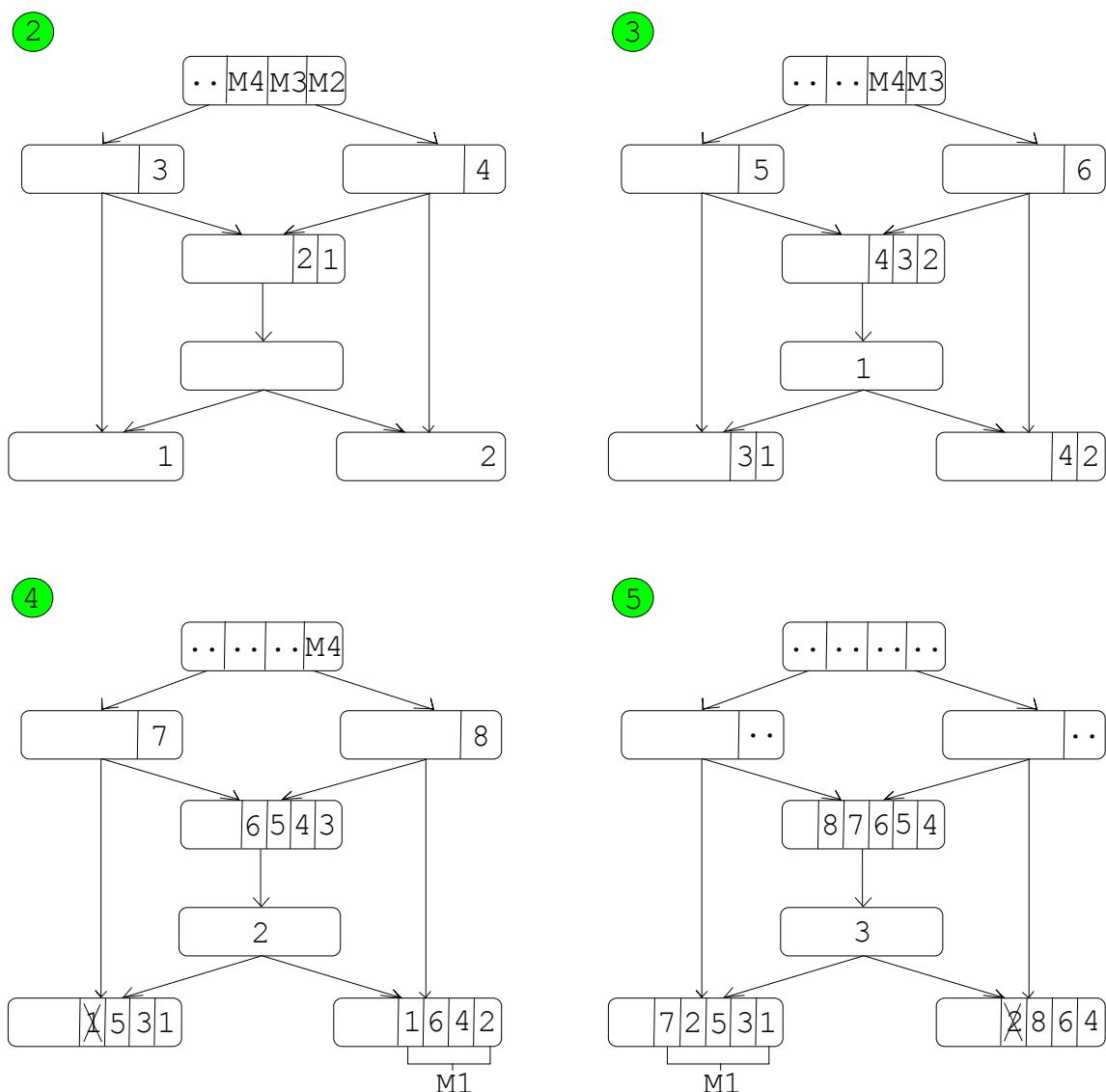


Figura 3.3 Numero di trasmissioni per ricevere un messaggio con rete a farfalla senza NC

Diversamente, la rete senza NC ha bisogno di un ulteriore trasmissione e quindi maggiore consumo di energia per poter inviare il primo blocco M1. Inoltre si evince che per ricevere il secondo messaggio, la rete con NC ha bisogno di una sola trasmissione in più mentre la rete senza NC ha bisogno di due trasmissioni in più. Un altro esempio è dato dalla Figura 3.3 raffigurante una rete a farfalla modificata per l'invio di tre messaggi con NC. ( $M1=[1;2;3]$   $M2=[4;5;6]$   $M3=[7;8;9]$   $M4=[10;11;12]$ ).

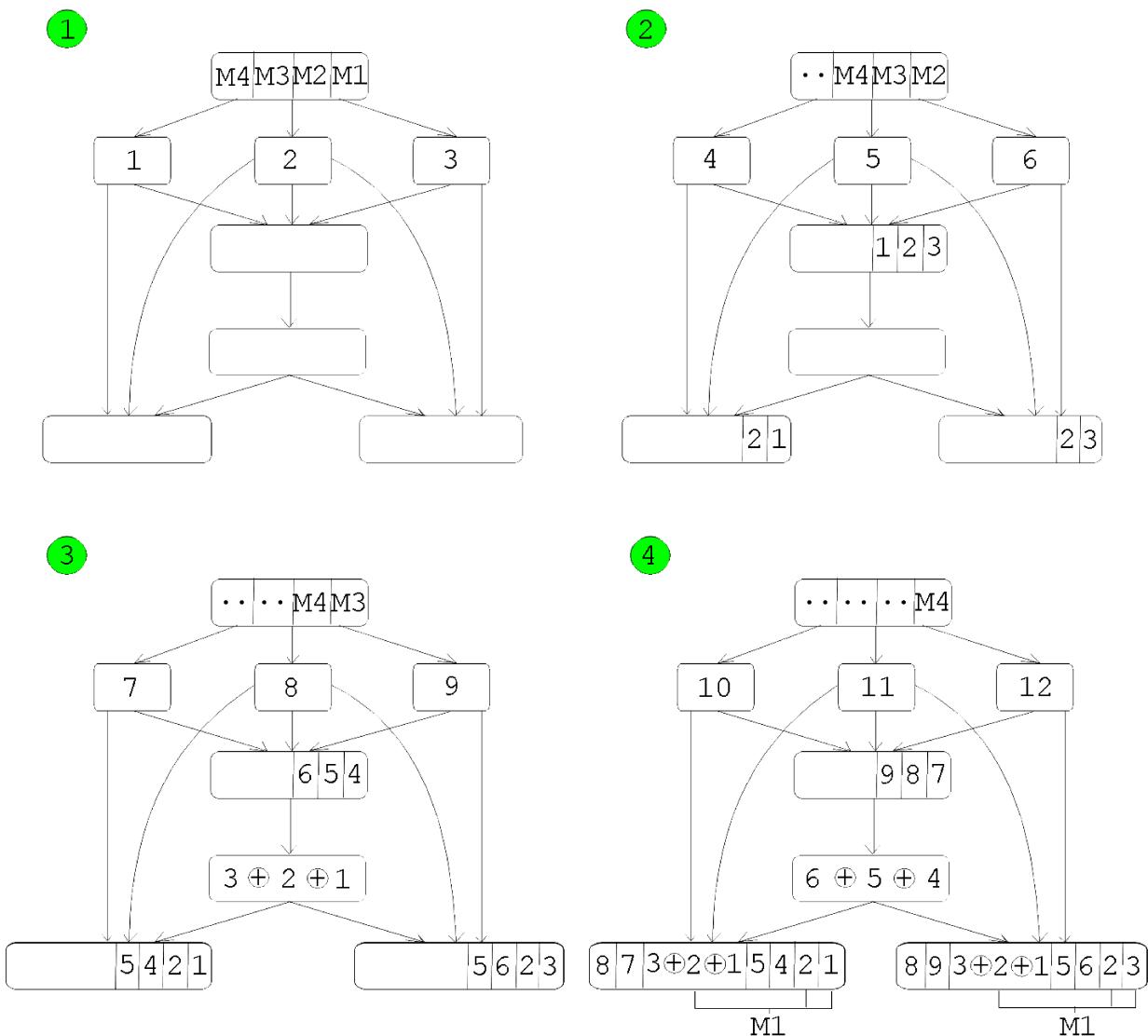


Figura 3.4 Numero di trasmissioni per ricevere un un blocco di tre messaggi con rete a farfalla modificata con NC

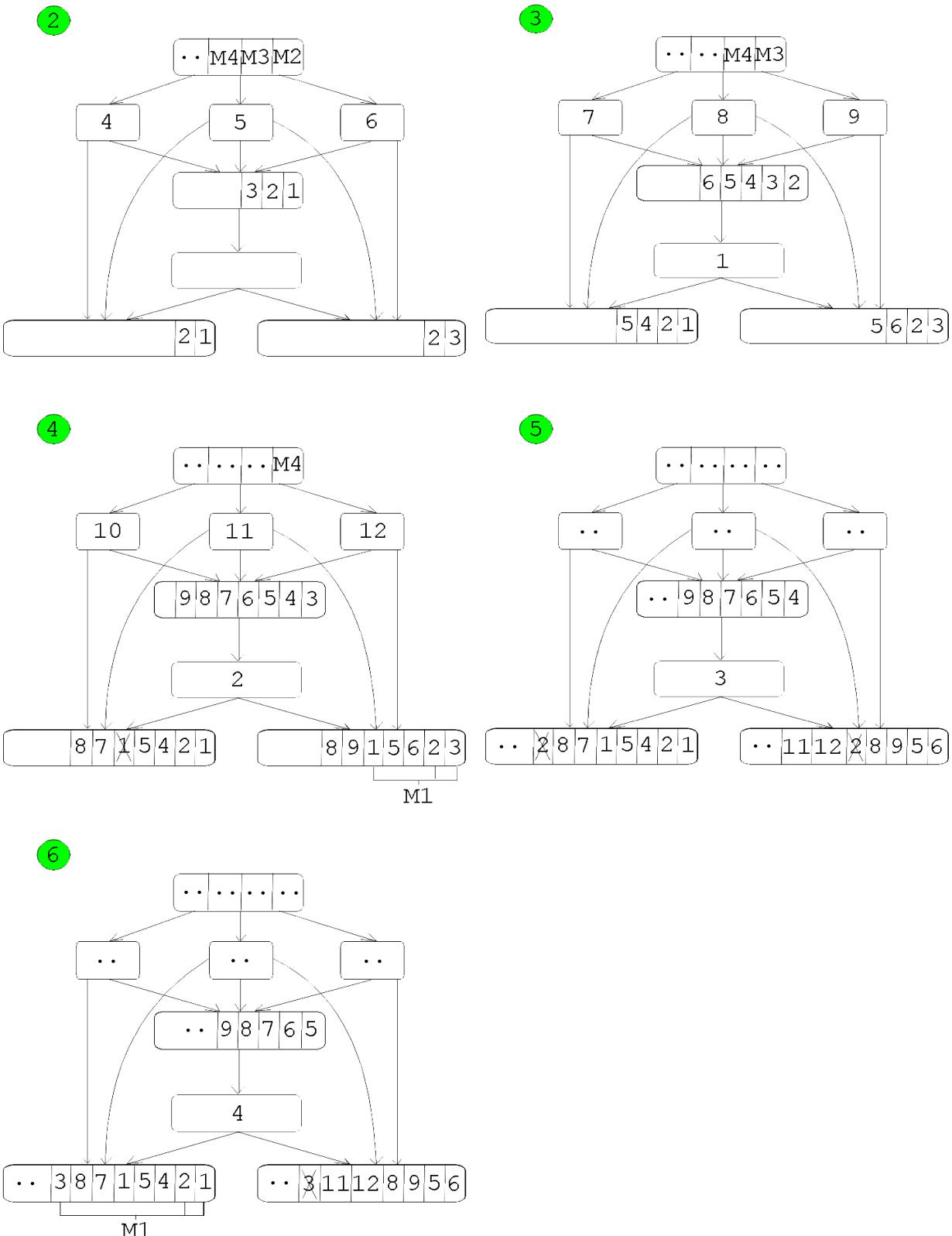


Figura 3.5 Numero di trasmissioni per ricevere un blocco di tre messaggi con rete a farfalla modificata senza NC

In queste due Figure (3.4, 3.5) si illustra un altro esempio di una rete a farfalla modificata per l'invio di tre messaggi nella rete. Anche qui grazie al NC si avrà un guadagno in termini di trasmissioni effettuate e quindi di consumo energetico, infatti utilizzando il classico *store-and-forward*, saranno necessarie dalle 5 alle 6 trasmissioni. Questo succede poiché, visto il nodo **D** (fare riferimento a Figura 2.1) che può ricevere e inviare uno dei tre messaggi, la probabilità che il nodo mandi per primo il messaggio già ricevuto da entrambi i nodi destinatari è di  $1/3$  ed in questo caso ci sarà bisogno di un'ulteriore trasmissione. Quindi a differenza della rete precedente (Figura 3.3) il numero di trasmissioni necessario per ricevere il successivo messaggio può andare dalle 5 alle 6 trasmissioni.

# CAPITOLO 4

## Conclusioni

In questa tesi si è studiato il Network Coding (NC) con i suoi relativi impieghi, benefici, caratteristiche e problemi.

L'obiettivo della tesi è stato creare un simulatore adattabile a una qualunque rete aciclica, direzionale, senza ritardi con capacità dei canali unitaria con l'uso dell'ambiente di sviluppo MATLAB applicando tale tecnica.

Il simulatore permette di inviare un file qualunque attraverso la rete, il quale verrà a destinazione ricavato attraverso le operazioni e le tecniche del RLNC. Grazie al simulatore è infatti stato possibile vedere il funzionamento di una trasmissione con tale tecnica e le relative migliorie che può portare come l'aumento del throughput e la robustezza della rete.

Il *Network Coding* è realmente un'interessante campo di ricerca per diverse applicazioni (Wireless, P2P,...) migliorando di gran lunga la modalità di trasferimento attuale di pacchetti (*store-and-forward*) riducendo anche il consumo di energia.

## Riferimenti

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, R. W. Yeung, “*Network information flow*”, IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, July 2000.
- [2] C. Fragouli, J. Widmer, “*Network coding: an instant primer*” ACM SIGCOMM Computer Communication Review, vol. 36, pp. 63--68, 2006.
- [3] C. Fragouli, E. Soljanin, “*Network Coding Fundamentals*”, Foundations and Trends in Networking, Vol. 2, No. 1, 2007.
- [4] S. Y. R. Li, R. W. Yeung, N. Cai, “*Linear Network Coding*”, IEEE Transactions On Information Theory, Vol. 49, No. 2, February 2003
- [5] P. A. Chou, “*Practical Network Coding for the Internet and Wireless Networks*” Globecom Tutorial, December 3, 2004.
- [6] K. Menger, “Zur allgemeinen Kurventheorie,” Fundamenta Mathematicae, Vol. 10, pp. 95–115, 1927.
- [7] S. Y. R. Li, R. W. Yeung, N. Cai, “*Linear Network Coding*”, IEEE Transactions On Information Theory, Vol. 49, No. 2, February 2003
- [8] <http://cs.nju.edu.cn/wuxb/NC.pdf>
- [9] T. Ho, R. Koetter, M. Medard, D. R. Karger, M. Effros, “*The Benefits of Coding over Routing in a Randomized Setting*”, IEEE International Symposium on Information Theory, 2003.
- [10] S. Katti, D. Katabi, “*Wireless Network Coding: Opportunities and Challenges*”, MILCOM, 2007
- [11] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, J. Crowcroft, “Xors in the air: practical wireless network coding.”, SIGCOMM. Pisa, Italy: ACM, September 2006, pp. 24 –254.
- [12] Tracey Ho,Desmond S-Lun, “*Network Coding:an introduction*”
- [13] S.-Y.R.Li,R.W.Yeung, and N.Cai, “*Linear Network Coding*”, IEEE Trans. IT,Vol. 49, No.2,Feb. 2003.
- [14] ITU-T Rec. G.805(03-2000) *Generic Functional Architectural of Transport Networks*,Ginevra,10 Marzo 2000.

- [15] I.Csiszar, "Linear codes for Sources and Source network: error exponents,universal coding", IEEE Trans. Inf.Theory, Vol.IT-28,No. 4,PP.585-592,Jul. 1982
- [16] T.H.,R.Koetter,M.Medard,D.R.Karger and M.Effros,"The benefits of coding over routing in a randomized settings", in International symposium on information theory(ISIT) 2003.
- [17] P.A. Chou,Y.Wu, and K.Jain, "Pratical network coding",in 41<sup>st</sup> Allerton Conference on communication control and computing, Allerton,IA,US,Oct. 2003.
- [18] Y.Wu,P.Chou, and K.Jain, "A comparison of network coding and tree packing", in international symposium on information theory(ISIT) 2004.
- [19] [http://it.wikipedia.org/wiki/Rete\\_di\\_telecomunicazioni](http://it.wikipedia.org/wiki/Rete_di_telecomunicazioni)
  
- [20] K.V. Rashmi, Nihar B. Shah, and Vijay Kumar, "Network Coding",July 2010.
  
- [21] <http://arxiv.org/abs/1212.2291>
- [22] [http://www.ericsson.com/technology/research\\_papers/wireless\\_access/doc/Multi-User%20ARQ.pdf](http://www.ericsson.com/technology/research_papers/wireless_access/doc/Multi-User%20ARQ.pdf)
- [23] <http://securenetworkcoding.wikidot.com/>
  
- [24] [http://home.eng.iastate.edu/~yuzhen/publications/ZhenYu\\_INFOCOM\\_2008.pdf](http://home.eng.iastate.edu/~yuzhen/publications/ZhenYu_INFOCOM_2008.pdf)
  
- [25] <http://netcod.org/papers/11AcedanskiDMK-final.pdf>
  
- [26] [http://www-bcf.usc.edu/~dimakis/RC\\_Journal.pdf](http://www-bcf.usc.edu/~dimakis/RC_Journal.pdf)
  
- [27] [people.csail.mit.edu/rahul/papers/cope-ton2008.pdf Sigcomm 2006 Form >> XORs in The Air: Practical Wireless Network Coding]
- [28] <http://arena.cse.sc.edu/papers/rocx.secon06.pdf>
  
- [29] <http://www.cs.wisc.edu/~shravan/infocom-07-2.pdf>
  
- [30] [Welcome to IEEE Xplore 2.0: Looking at Large Networks: Coding vs. Queueing](#)
  
- [31] <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4549741>
- [32] A.J. González, D. Rodríguez, J. López, F. I. Rillo, J. Alcober, "Streaming P2P robusto en redes Ad-hoc utilizando información social", Jornadas De Ingeniería Telemática, 2009.

- [33] Z. Li and B. Li. “*Network Coding in Undirected Networks*”, In CISS 04, 2004
- [34] T. Ho and R. Koetter. “*Online incremental network coding for multiple unicasts*”,In DIMACS Working Group on Network Coding, 2005
- [35] <http://it.wikipedia.org/wiki/MATLAB>