UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Controllo di comportamenti egoistici in reti di telecomunicazioni

Tesi di laurea triennale

Laureanda: Mihaela Calin

Relatore: Prof. Stefano Tomasin

Anno Accademico 2009/2010

Padova, 23 Luglio 2010

Ringraziamenti

"Anche se ciò che puoi fare è soltanto una piccola goccia nel mare,

può darsi che sia proprio quella a dare significato alla tua esistenza…"

(Romano Battaglio)

Vorrei ringraziare al professore per la pazienza e la disponibilità, ai miei amici per avermi aiutata nei momenti difficili della mia vita, all'IKEA per essere stata la fonte finanziaria principale per i miei studi, a Federico e alla Maurilia per aver dedicato del loro tempo prezioso nella correzione della mia tesi, e soprattutto ai miei genitori per avermi regalata la libertà di scegliere e per avermi sostenuta anche quando le mie scelte non sono state le migliori in assoluto. Approfittò di mandare un bacio alla mia nonna e a tutta la famiglia. Vi voglio tanto bene.Grazie.

Indice

Introduzione

Negli ultimi anni, la richiesta di servizi wireless è in continuo accrescimento: con essa aumenta anche l'esigenza di un uso efficiente delle risorse disponibili. L'efficienza totale del sistema dipende in grande misura dal numero di terminali mobili (nodi) contemporaneamente attivi. È fondamentale che il coordinamento di un sistema del genere sia automatico e distribuito, e che ogni terminale regoli le grandezze in gioco, come rate trasmissivo e potenza, non solo in base alle proprie esigenze, ma anche tenendo conto delle condizioni dell'ambiente circostante. Questo vale per qualsiasi dispositivo che sia in grado di instaurare una connessione senza fili. La libertà di movimento che questo tipo di reti concede è il suo principale punto di forza, così come la possibilità di costruire una rete locale ovunque e con costi contenuti. Per tale motivo, questa tesi ha come studio le reti wireless, in particolare il protocollo *Media Access Control* (MAC). Tradizionalmente, i progettisti dei protocolli di rete partono dal presupposto che il loro protocollo sarà rispettato ed eseguito correttamente da tutti nodi della rete. Questa trattazione è una raccolta di documenti di ricerca che analizzano la rete in presenza di nodi che non rispettano il protocollo MAC, a scopo di ottenere maggior beneficio, e propongono delle soluzioni per l'individuazione e la punizione di tali nodi.

Gli argomenti trattati, sono organizzati in questo modo: il Capitolo 1 descrive i principi di funzionamento delle reti wireless; il Capitolo 2 approfondisce il significato di comportamento egoistico nelle reti wireless, specifica come esso può essere realizzato e propone come soluzione un sistema di individuazione di tale comportamento per le reti con infrastruttura; infine, il Capitolo 3 analizza il comportamento egoistico nelle reti *ad-hoc* (senza infrastruttura) utilizzando la teoria dei giochi, e propone una tecnica di rilevazione e punizione dei nodi che esibiscono un comportamento non cooperativo.

Capitolo 1

Lo standard IEEE 802.11

1.1 Introduzione

La fine degli anni '70 vide la comparsa sul mercato statunitense delle *Local Area Network* (LAN). Una LAN è un sistema di comunicazione che permette ad apparecchiature indipendenti di comunicare tra di loro, entro un'area delimitata, utilizzando un canale fisico a velocità elevata e con un basso tasso d'errore.
Con il termine *Wireless LAN* (WLAN), si indica una struttura (rete) in grado di interconnettere apparecchiature elettroniche di diversa natura, libere di muoversi senza i cavi di collegamento entro un'area dell'ordine delle centinaia di metri quadri, e che comunicano fra loro grazie alle tecnologie radio. I nodi della rete wireless sono in grado di comunicare anche con l'esterno, e quindi con altre WLAN o con Internet. Le WLAN possono dunque essere considerate come l'estensione senza fili delle comuni reti LAN.
Rispetto a queste ultime, le WLAN presentano indiscutibili vantaggi legati alla mobilità, flessibilità e facilità d'installazione. Per contro, la particolare natura del mezzo di comunicazione radio pone dei problemi:

- di attenuazione, dovuti alla presenza di ostacoli o al verificarsi di riflessioni indesiderate non prevedibili in fase di progetto che modificano il cammino delle onde elettromagnetiche, noto come *fenomeno dei cammini multipli* o *multipath fading*;
- di interferenza con segnali provenienti da altre sorgenti elettromagnetiche;
- di sicurezza delle comunicazioni (in particolare di autenticità della sorgente e di segretezza dei dati trasmessi);
- di durata finita delle batterie dei dispositivi.

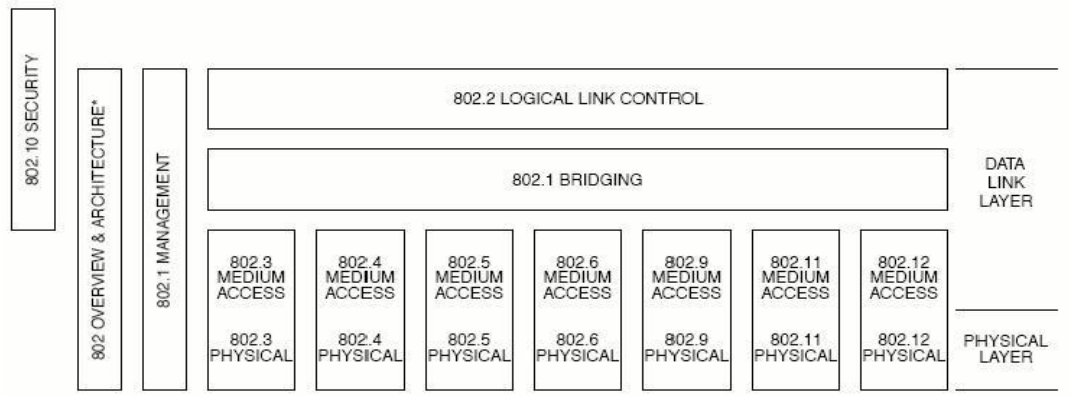


Figura 1.1. Famiglia 802 e relazioni con modello OSI [2]

Inoltre il raggio di copertura di una rete locale senza fili è inferiore rispetto ad una LAN cablata: nel primo caso si possono raggiungere distanze dell'ordine del centinaio di metri, nel secondo invece si può arrivare fino ad una distanza di 2.5 km.

Esistono in commercio e in progetto svariati standard che mirano al mercato delle WLAN, solitamente non compatibili fra loro. Il più diffuso è lo standard IEEE 802.11 [1], definito nel 1997 dall'*Institute of Electrical and Electronics Engineers* (IEEE), operante sia nella banda dei 2.4GHz che dei 5GHz, e raggiungendo rispettivamente velocità di 11Mbps (802.11b) e 54Mbps (802.11a e 802.11g). Lo standard 802.11 fa parte della famiglia 802, la quale implementa una serie di specifiche per le reti locali (LAN). La figura 1.1 mostra le relazioni tra le componenti della famiglia e il modello *Open Systems Interconnection* (OSI – conosciuto meglio come modello ISO/OSI). OSI è il modello di riferimento per le reti, costituito da sette strati (livelli, oppure *layers*) e che realizza una comunicazione per livelli, ovvero, dati due nodi *A* e *B*, il livello *n* del nodo *A* può scambiare informazioni con il livello *n* del nodo *B* ma non con gli altri.

Tutte le reti dello standard IEEE 802 hanno due componenti principali:

- *Medium Access Control* (MAC) - insieme di regole che determinano come accedere al mezzo e trasmettere i dati;
- *Physical* (PHY) - rappresenta in dettaglio come avviene la trasmissione e la ricezione.

1.2 Architettura di rete

Lo standard definisce due diverse topologie architetturali:

- *Independent Basic Service Set* (IBSS) o Ad Hoc Network;
- *Extend Service Set* (ESS) o Infrastructure Mode

L'elemento base é rappresentato dalla *stazione* (nodo), ovvero una qualsiasi unità che contiene le funzionalità del protocollo 802.11. Le stazioni possono essere mobili (palmari), portatili (PC) o stazionarie (*Access Point*). Un insieme di stazioni situate nella stessa area geografica – *Basic Service Area* (BSA) e che sono in grado di comunicare

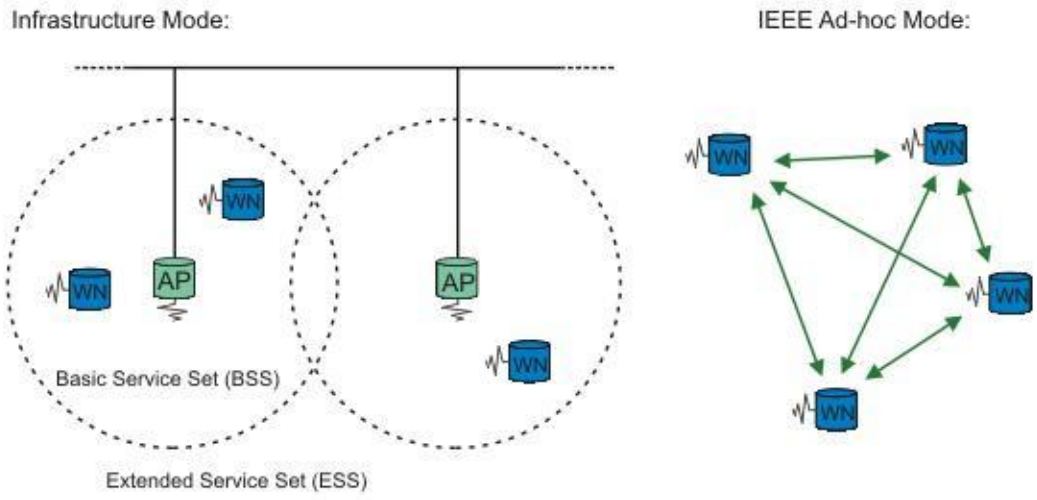


Figura 1.2. Topologie architetturali dello standard 802.11 [3]

facilmente fra loro in maniera diretta, utilizzando la stessa frequenza radio, costituisce un *Basic Service Set* (BSS). Una BSS può essere di tipo *strutturato* o *Ad Hoc:* nel primo caso, all'interno della BSS, vi è un nodo particolare, l'*Access Point* (AP), il cui compito è quello di coordinare, sincronizzare ed identificare la BSS stessa; nel secondo caso non vi è nessun AP e le stazioni comunicano tra loro direttamente, se sono in visibilità radio, o passando attraverso nodi intermedi che hanno il compito di trasmettere il segnale fino a raggiungere il destinatario.

In un IBSS le stazioni comunicano direttamente tra loro (non c'e la necessità di un AP) ed una stazione è raggiungibile solo se situata entro il raggio di copertura.

Un ESS è realizzato da un insieme di BSS strutturate, dove gli AP comunicano tra di loro attraverso il *Distribution System* (DS) per trasportare il traffico da una BSS all'altra, agevolando lo spostamento delle stazioni wireless tra BSS. Nella figura 1.2 vengono illustrate le due topologie architetturali.

1.3 Architettura protocollare

Come detto in precedenza, tutte le reti 802 hanno due componenti [1]:

- il livello fisico (PHY) che rappresenta in dettaglio come avviene la trasmissione e la ricezione;
- il livello di linea (MAC) che corrisponde all'insieme di regole che determinano come accedere al mezzo e spedire i dati.

1.3.1 Livello fisico

Il livello fisico (PHY) è quello più basso previsto dal modello ISO/OSI; esso si interfaccia direttamente con l'hardware ed è collocato logicamente al di sotto del livello MAC; in particolare al livello PHY sono affidati tre compiti:

- rilevamento della portante fisica: *Physical Carrier Sense*, che consente di determinare se il canale è libero oppure è occupato;
- trasmissione sul mezzo fisico, mediante opportuna modulazione del segnale, dei singoli bit che costituiscono il frame proveniente dal livello MAC;
- ricezione, mediante appropriata demodulazione del segnale, dei singoli bit che compongono il frame in arrivo.

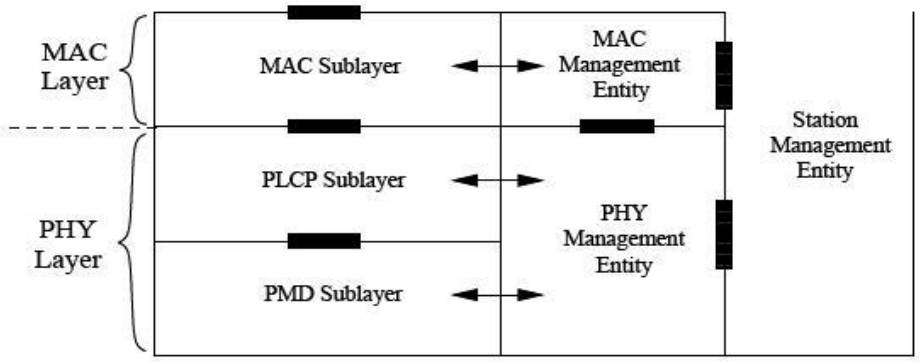


Figura 1.3 Modello di riferimento dei livelli PHY e MAC definiti dal IEEE 802.11 [1]

Con il termine *frame* si identifica un insieme di ottetti con una struttura ben definita e nota a priori sia al nodo trasmittente che a quello ricevente. In seguito, verrà presentata brevemente la struttura di un MAC frame 802.11.

Come illustrato in figura 1.3, l'architettura logica del livello PHY è strutturata in questo modo:

- *Physical Layer Management Entity* (PLME) - costituisce l'interfaccia con il livello MAC sovrastante e gestisce i due seguenti sottolivelli;
- *Physical Medium Dependent* (PMD) - definisce le caratteristiche e la metodologia di trasmissione e ricezione dei dati attraverso il mezzo radio tra due o più stazioni in base alle caratteristiche della specifica tecnologia usata (*Direct Sequence Spread Spectrum* (DSSS), *Frequency-hopping Spread Spectrum* (FHSS), infrarossi, *Orthogonal Frequency-Division Multiplexing* (OFDM), *High Rate Direct Sequence Spread Spectrum* (HR-DSSS));
- *Physical Layer Convergence Procedure* (PLCP) *Sublayer* – esegue il *sensing* a livello fisico attraverso il segnale *Clear Channel Assesment* (CCA) che stabilisce se il canale è libero. Inoltre, PLCP mappa i frame del MAC in frame adatti per la trasmissione e ricezione dei dati e delle informazioni di gestione tra due o più stazioni utilizzando lo specifico sistema PMD.

1.3.2 *Media Access Control (MAC) – Distributed Coordination Function (DCF)*

In una Wireless LAN le stazioni sono obbligate a condividere lo stesso canale (mezzo) trasmissivo. Questo implica il problema delle *collisioni* nel caso avvenga la trasmissione contemporanea da parte di due o più stazioni. Per evitare questo problema, lo standard in questione utilizza il protocollo *Carrier Sensing Multiple Access/Collision Avoidance* (CSMA/CA) con *acknowledgment* (ACK) definendo così il *Distribuited Coordination Function* (DCF). Il DCF costituisce il metodo fondamentale di funzionamento di una rete IEEE 802.11 [1] ed è, quindi, l'unico obbligatoriamente presente in tutte le implementazioni. Questo è un protocollo *distribuito*, nel senso che l'ordine di accesso al canale viene determinato da un algoritmo che lavora simultaneamente su tutte le stazioni che partecipano alla contesa. Il DCF è un protocollo finalizzato a minimizzare per quanto possibile le probabilità di collisione: ciò si ottiene cercando di rendere casuale e differenziato l'istante di accesso al canale da parte di ciascuna stazione che ha dati da trasmettere.

1.3.2.1 Meccanismo base di accesso DCF

Ogni stazione che vuole trasmettere un frame di dati ascolta il canale ed aspetta che finiscano le eventuali trasmissioni correnti; una volta che il canale diventa libero, viene atteso un tempo DCF *Inter-Frame Spacing* (DIFS), e poi ha inizio la contesa; al termine di questa, solo la stazione vincitrice trasmette il proprio frame di dati; terminata la sequenza di trasmissione per un frame dati il canale si libera e la procedura si ripete. Se invece il canale, nel momento in cui la stazione si accinge per la prima volta alla trasmissione di un frame, dovesse risultare subito libero, non c'è bisogno del ricorso al meccanismo di contesa: la trasmissione inizia semplicemente dopo aver constatato la condizione di canale libero per un tempo almeno pari ad un DIFS.

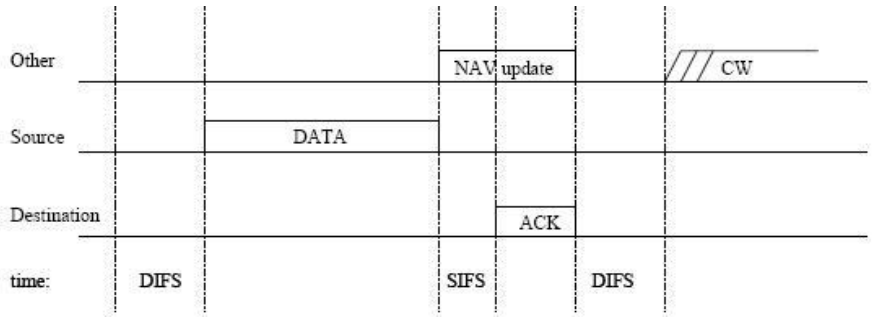


Figura 1.4 Meccanismo di accesso base di tipo CSMA/CA [4]

Più in dettaglio, come mostrato in figura 1.4, la stazione che vince la contesa trasmette il proprio frame di dati; il nodo ricevente attende un tempo *Short Inter-Frame Spacing* (SIFS) dopo la fine della ricezione del frame di dati e genera un frame di *acknowledgment* (ACK) ad indicare che la trasmissione è andata a buon fine; in particolare, il frame di ACK è generato solo se non sono avvenute collisioni e se il controllo degli errori a livello fisico, grazie all'uso del *Cyclic Redundancy Check* (CRC), non indica la presenza di errori nei bit ricevuti: il verificarsi di queste due condizioni implica che il frame è stato ricevuto correttamente.

1.3.2.2 Parametri temporali

Il *Short Inter-Frame Spacing* (SIFS) è il più breve intervallo temporale che può sussistere tra due frame successivi ed indica il tempo necessario al nodo per convertire il suo modo di funzionamento passando dalla fase di trasmissione a quella di ricezione, quindi il tempo necessario a decodificare il pacchetto entrante; la sua durata dipende dal livello fisico utilizzato (Tabella 1).

Il *DCF Inter-Frame Spacing* (*DCF IFS*), usato esclusivamente nella modalità DCF, rappresenta il tempo (con il canale libero) che una stazione deve attendere prima di iniziare la contesa. Il suo valore è costante ed è fornito dallo standard IEEE 802.11 secondo l'espressione:

$$DIFS = SIFS + 2 \times SlotTime.$$

Quando una stazione, ricevendo un pacchetto, rileva degli errori di trasmissione tramite il controllo a ridondanza ciclica (*Cyclic Redundancy Check* (CRC)); il tempo di attesa viene riportato a *Extended IFS (EIFS)*, che è l'interframe più lungo e la sua durata dipende dal livello fisico secondo la seguente espressione:

$$EIFS = SIFS + ACK_{time} + DIFS.$$

Potrebbe accadere che due o più stazioni, a seguito della contesa, inizino le trasmissioni nello stesso istante. Questa situazione non è rilevabile dalle stazioni trasmittenti che, quindi, continueranno a trasmettere il frame di dati fino alla fine. Sarà l'assenza dell'ACK dopo un SIFS ad indicare alla stazione trasmittente l'errata trasmissione del frame di dati e, in tal caso, dopo un DIFS a partire dall'istante in cui il canale è ritornato libero avrà inizio una nuova contesa per la ritrasmissione degli stessi frame.

Parameters	*802.11a*	*802.11b (FH)*	*802.11b (DS)*	*802.11b (IR)*	*802.11b (High Rate)*
Slot Time (μs)	9	50	20	8	20
SIFS (μs)	16	28	10	10	10
DIFS (μs)	34	128	50	26	50
EIFS (μs)	92.6	396	364	205 or 193	268 or 364
$CW_{min}(SlotTime)$	15	15	31	63	31
$CW_{max}(SlotTime)$	1023	1023	1023	1023	1023

Tabella 1. Parametri dipendenti dai vari livelli fisici [4]

1.3.2.3 Algoritmo di *backoff* casuale esponenziale

La contesa fra le diverse stazioni avviene secondo un meccanismo probabilistico denominato algoritmo di *backoff* casuale esponenziale, *Exponential Random Backoff Algorithm*: ciascuna stazione estrae un valore intero casuale secondo una distribuzione uniforme nell'intervallo [0, *CW*), dove la *Contention Window* (*CW*) è un parametro definito dallo standard che varia dinamicamente all'interno di ciascuna stazione nell'intervallo [CW_{min}, CW_{max}]. Il valore estratto indica il numero di *SlotTime* (unità temporale dipendente dal particolare livello fisico: Tabella 1) che la stazione dovrà ulteriormente attendere, dopo il DIFS, prima di iniziare a trasmettere il proprio frame di da-

ti. Ogni nodo possiede un proprio timer interno, chiamato *Backoff Timer* che conserva il numero corrente di *slot* ancora da attendere; passato un tempo pari ad un *SlotTime*, se il canale dovesse essere ancora libero e non sono iniziate altre trasmissioni, il timer viene decrementato; solo quando il timer raggiunge il valore zero la stazione avrà vinto la contesa e quindi inizierà la trasmissione. Se invece durante il conteggio alla rovescia, in corrispondenza della fine di uno *slot* temporale, il canale dovesse risultare occupato, il valore del timer non verrà ulteriormente decrementato, ma sarà congelato e conservato, per poi essere utilizzato come valore iniziale nella successiva contesa. In questo modo si crea una sorta di precedenza per le stazioni che da più tempo sono in attesa di trasmettere. Una volta vinta la contesa e trasmesso il frame, viene estratto un nuovo valore di *backoff* da utilizzare nella successiva contesa; questo sarà sempre distribuito uniformemente nell'intervallo [0, *CW*], ma il *CW* sarà variato a seconda che la trasmissione appena conclusa sia andata a buon fine o meno; in particolare, in caso di presenza dell'ACK, il nuovo *CW* verrà settato al valore CW_{min}; in caso di assenza (solitamente causata da una collisione) il *CW* verrà modificato secondo la formula:

$$CW = 2^{2+i} - 1 \qquad 1 \leq i \leq 6$$

dove *i* rappresenta il numero di tentativi di trasmissione di uno stesso frame. L'applicazione di tale formula si traduce in un progressivo raddoppio di *CW*, non superando però mai il valore CW_{max}. I due parametri CW_{min} e CW_{max} (come anche *SlotTime*) sono definiti in funzione del livello fisico utilizzato (Tabella 1).

1.3.2.4 Meccanismo di accesso basato su *Request-to-Send/Clear-to-Send* (RTS/CTS)

Per quanto riguarda i metodi con cui una stazione distingue lo stato di canale occupato da quello di canale libero da trasmissioni, vi è principalmente un *Carrier Sensing* di tipo fisico (in cui fisicamente viene misurata la potenza del segnale ricevuto) a cui viene affiancato un *Virtual Carrier Sensing*: nell'intestazione MAC di ogni tipo di frame tra-

smesso vi è un campo, *Duration Field*, in cui viene specificata (in unità di microsecondi), a partire dalla fine della trasmissione del frame stesso, la durata necessaria a completare la corrente sequenza di scambio di frame. In tal modo, ad esempio, nel *Duration Field* di un frame di dati sarà specificato il tempo occorrente per la trasmissione di un ACK sommato ad un SIFS (che intercorre fra il frame dati e l'ACK stesso). Tutte le stazioni in ascolto che ricevono il frame, aggiornano un proprio registro interno *Network Allocation Vector* (NAV), in funzione del *Duration Field* ricevuto.

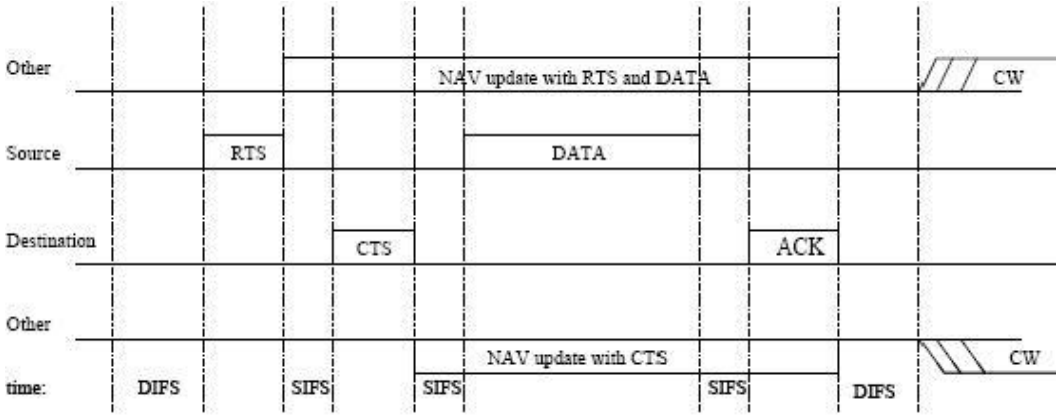


Figura 1.5 Meccanismo di accesso basato su RTS/CTS [4]

Il *virtual carrier sensing* è utile soprattutto nelle situazioni in cui vi sono stazioni non in grado di ascoltare le trasmissioni di tutte le altre; questo è il problema del terminale nascosto (*hidden stations)*: due stazioni non in visibilità radio fra loro (ma che comunque vedono l'AP) possono sentire il canale libero e inviare entrambe un frame, causando una collisione e, quindi, la perdita di entrambi i frame.

La soluzione si è ottenuta con l'uso di due nuovi frame e con l'ausilio del *virtual carrier sensing*; la stazione che vince la contesa, prima di trasmettere il frame di dati, invia una richiesta di invio *Request To Send* (RTS); il frame che contiene l'RTS, che non ha priorità rispetto agli altri frame, include l'indirizzo del destinatario del successivo frame dati e la durata della trasmissione. La durata include anche il tempo necessario per il riscontro.

Ogni nodo che riceve un pacchetto RTS deve considerare il canale occupato per tutta la durata indicata nel campo RTS. Ciò avviene utilizzando il NAV. Se il destinatario del frame RTS riceve correttamente la richiesta, aspetta un intervallo SIFS, dopodiché risponde con un frame di via libera *Clear To Send* (CTS), che contiene anche esso la durata della trasmissione. Tutte le stazioni che ricevono il frame CTS impostano il loro NAV come nel caso precedente, se non lo hanno già fatto. L'insieme delle stazioni che ricevono l'RTS può non coincidere con l'insieme delle stazioni che ricevono il CTS. Alla fine della procedura RTS/CTS, tutte le stazioni entro il raggio di ricezione del trasmettitore e del ricevitore sono informate della trasmissione dati che sta per iniziare e si astengono dall'accedere al mezzo fisico. Dopo un intervallo SIFS, il trasmettitore invia il frame dati. Dopo aver ricevuto correttamente il frame, il ricevitore aspetta un tempo SIFS ed invia il riscontro. Una eventuale collisione potrebbe ancora avvenire, ma tale evenienza è ristretta al solo intervallo in cui viene inviato il frame RTS e al SIFS immediatamente successivo.

Questa modalità, che fa uso dei frame RTS/CTS ha lo svantaggio, soprattutto se i data frame sono di dimensioni molto piccole, di diminuire l'efficienza di utilizzo del canale (riducendo in modo non trascurabile la banda disponibile ed aumentando il ritardo di consegna). È per questo che viene solitamente utilizzata solo quando il frame di dati da trasmettere è sufficientemente grande. Il meccanismo appena descritto viene illustrato nella figura 1.5.

Oltre al DCF, il livello MAC definito dalla versione base dello standard IEEE 802.11 prevede un'altra modalità di funzionamento e di accesso al canale, indicata come *Point Coordination Function* (PCF).

1.3.3 MAC – *Point Coordination Function* (PCF)

Il *Point Coordination Function* (PCF) è un protocollo di accesso con funzione di coordinamento di tipo centralizzato. Si tratta di una modalità opzionale pensata per cercare

di garantire la *Quality of Service* (QoS), ovvero di garantire le prestazioni dell'architettura di rete come l'ampiezza di banda e il ritardo.
Il PCF prevede l'esistenza di un nodo particolare chiamato *Point Coordinator*, solitamente coincidente con l'AP, cui è affidato il compito di assegnare di volta in volta il diritto di trasmissione. Il *point coordinator* effettua l'interrogazione ciclica di tutte le stazioni (*polling*) e solo la stazione che riceve il *poll* può inviare i dati. Poiché la stazione che riceve il *poll* in un dato istante è una sola, non possono verificarsi collisioni.
I due metodi di accesso al canale possono essere presenti entrambi, non contemporaneamente, ma alternandosi temporalmente. Può essere presente solo il DCF, ma non è assolutamente possibile avere solo il PCF. Il periodo temporale in cui viene utilizzato il DCF viene chiamato *Contention Period*, mentre quello in cui viene utilizzato il PCF viene chiamato *Contention Free Period*.

1.4 Frame 802.11

L'unità di trasmissione al livello MAC per un nodo, sia esso una semplice stazione o l'AP, è il *frame* o anche *Mac Protocol Data Unit* (MPDU); come detto precedentemente, un frame è un insieme di ottetti con una struttura ben definita e nota a priori sia al nodo trasmittente che a quello ricevente. All'interno del frame vi sono vari campi che contengono informazioni diverse.
Il tipico frame (di dati) è costruito unendo i contributi provenienti da tutti i livelli dello stack di protocolli ISO/OSI, in particolare per ogni livello il frame elaborato, proveniente dal livello superiore, cui si è aggiunto l'header, costituisce un *Protocol Data Unit* (PDU).
Si possono distinguere tre tipi fondamentali di frame supportati dallo standard 802.11:

- *Data Frame,* usati per il trasporto dei dati sia durante il *Contetion Period,* sia durante il *Contention Free Period*;

- *Control Frame*, usati per il controllo dell'accesso al canale, come i frame *RTS*, *CTS* e i frame *ACK*; sono generati interamente al livello MAC;
- *Management Frame*, impiegati nelle operazioni di associazione con l'Access Point, nelle procedure di autenticazione e per la gestione della sincronizzazione.

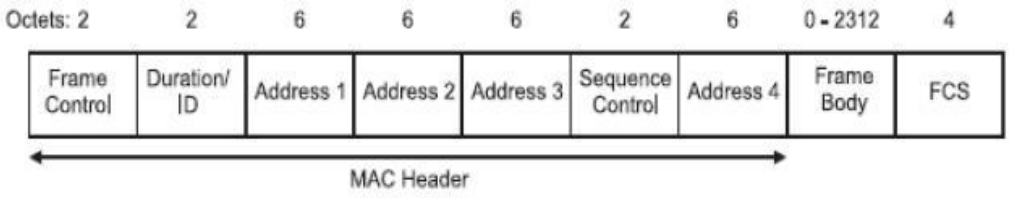


Figura 1.6 Struttura di un MAC frame 802.11 [5]

La struttura generale di un frame 802.11 a livello MAC, riportata in figura 1.6, è la seguente:

- un' intestazione, *MAC Header*, contenente informazioni di controllo, durata (informazione utilizzata dalle stazioni riceventi per aggiornare il NAV), indirizzamento e sequenza (indica l'ordine dei pacchetti e le duplicazioni che si rendono necessarie per la presenza di errori);
- un corpo, *Frame Body*, detto anche *MAC Service Data Unit* (MSDU), contenente informazioni specifiche del tipo di frame e corrispondente al PDU proveniente dal livello superiore;
- un campo di controllo dell'errore (FCS, *Frame Check Sequence*), che contiene un valore a 32 bit calcolato dal livello MAC del nodo mittente in funzione di tutti i precedenti ottetti presenti nel frame utilizzando un polinomio noto come *Cyclic Redundancy Code* (CRC-32); il livello MAC del destinatario riesegue il calcolo, servendosi dello stesso polinomio generatore e controlla se si sono verificati errori in trasmissione.

Capitolo 2

Individuazione del comportamento egoistico negli *hotspot*

Le reti wireless IEEE 802.11 [1] (conosciute anche come *Wi-Fi*) inizialmente erano destinate ad essere utilizzate in luoghi relativamente protetti, come gli uffici aziendali, e di conseguenza, la sicurezza e la garanzia di un accesso equo ha ricevuto scarsa attenzione. Ma negli ultimi anni, si stanno diffondendo sempre più *Access Point* pubblici, chiamati *hotspot*, che offrono accesso wireless a Internet da luoghi pubblici (hotel, aeroporti) utilizzando la tecnologia IEEE 802.11. In questo contesto, far uso improprio del protocollo MAC porta tanti vantaggi in quanto:

- si possono ottenere notevoli guadagni di larghezza di banda, dato che il MAC determina l'accesso al mezzo wireless e la trasmissione dei dati;
- il MAC è nascosto da strati superiori per cui il suo malfunzionamento non può essere rilevato da meccanismi progettati per questi livelli; quindi, può essere combinato con malfunzionamenti degli strati superiori per rafforzarlo;
- è sempre applicabile in ambienti Wi-Fi, perché tutte le stazioni wireless utilizzano lo stesso protocollo MAC IEEE 802.11

Il comportamento egoistico (*selfish misbehavior*) allo strato MAC consiste nel modificare il funzionamento del protocollo IEEE 802.11 cambiando dei parametri definiti dallo standard oppure non rispettando le procedure di comunicazione previste, a scopo di migliorare le proprie prestazioni (*throughput*, latenza, energia, ecc.). Quindi un nodo della rete non addotta un comportamento egoistico se non trae alcun vantaggio. Al contrario, il *malicious misbehavior*, a cui si fa riferimento quando si parla della sicurezza delle reti wireless, è un comportamento a scopo distruttivo in cui si vuole danneggiare il

funzionamento della rete, anche se ciò non porta alcun guadagno di prestazioni per il nodo *malicious*.

Il *selfish misbehavior* comprende gli *host* (stazioni) che si rifiutano di inoltrare i pacchetti delle altre stazioni per risparmiare energia, oppure host che selezionano piccoli valori di *backoff* per ottenere un *throughput* maggiore (questo è il comportamento che verrà analizzato); il *malicious misbehavior* invece include attacchi *denial of service*, oppure il *jamming* del canale wireless, cioè il disturbo provocato intenzionalmente inserendo una sorgente di rumore sul canale a scopo di diminuire o negare la comunicazione tra le stazioni.

In quello che segue verrà utilizzato il termine *cheater* per identificare la stazione che addotta un comportamento egoistico. Le tecniche utilizzate per ottenere un tale comportamento saranno chiamate tecniche *misbehavior*.

2.1 Tecniche misbehavior

Diversi studi [6], [7] hanno dimostrato che circa il 90% del traffico trasportato dalle WLAN usa il *Transfer Control Protocol* (TCP) ed è principalmente di tipo *downlink*, cioè diretto dall'*Access Point* (AP) alle stazioni utenti. Pertanto è importante distinguere le tecniche di *misbehavior* in base al tipo di traffico di destinazione [8].

2.1.1 Traffico *Uplink* (diretto dalle stazioni all'AP)

Un *cheater* può eseguire lo *scrambling* selettivo dei frame trasmessi da parte delle altre stazioni, al fine di aumentare la loro finestra di contesa. Lo *scrambling* è una tecnica simile al *jamming*, ma agisce per periodi di tempo più limitati e riguarda frame specifici o parte di essi. I frame di interesse sono:

1. CTS – in questo caso, il *cheater* sente un frame RTS inviato ad un'altra stazione e provoca intenzionalmente la collisione e la perdita del corrispondente frame

CTS a scopo di impedire il successivo scambio di sequenze lunghe di frame (come detto prima, il meccanismo RTS/CTS viene utilizzato generalmente per frame grandi). Di conseguenza, il canale diventa inattivo dopo il CTS danneggiato, la finestra di contesa della stazione mittente viene raddoppiata, e il *cheater* ottiene maggior possibilità di inviare i suoi dati.

2. ACK e frame DATA – anche se provocare la collisione di questi frame non aiuta a diminuire il tempo di attesa per la trasmissione dei dati, provoca il raddoppio della finestra di contesa della stazione destinataria dell'ACK (vale a dire, della sorgente DATA) e di conseguenza, quest'ultima seleziona valori di *backoff* più grandi e come prima il *cheater* aumenta le sue probabilità di accedere al canale.

Un *cheater* può manipolare i parametri del protocollo, per incrementare la sua larghezza di banda, in questo modo:

1. Quando il canale è inattivo, trasmette dopo il SIFS ma prima del DIFS.
2. Durante l'invio di RTS o frame DATA, imposta il campo *duration* (che si trova nell'intestazione di questi frame) ad un valore elevato, così, le altre stazioni che aggiornano il loro NAV con questo valore, non parteciperanno più alla contesa per tutto questo periodo.
3. Riduce il tempo di *backoff*. Ciò può essere fatto scegliendo una piccola finestra di contesa fissa.

Un *cheater* può anche combinare le varie tecniche sopra elencate oppure adattare il suo comportamento in modo da non poter essere identificato.

2.1.2 Traffico *downlink* (dall'AP alle stazioni)

In questo caso, il *cheater* intende aumentare il volume di traffico inviato a sé attraverso l'AP, quindi, aumentare il numero di pacchetti destinati a esso nella coda dell'AP. Per raggiungere quest'obiettivo, il *cheater* punterà sui protocolli responsabili del riempimento della coda. Si distinguono due tipi di sorgenti che inviano traffico alle stazioni wireless tramite l'AP:

1. Sorgente UDP – utilizza come protocollo di trasporto l'*User Datagram Protocol* (UDP). L'UDP è un protocollo *connection-less*, cioè i dati sono inviati senza alcuna predisposizione preventiva. Attaccare il traffico UDP è inutile in quanto l'UDP non richiede alcun riconoscimento da parte del ricevitore e quindi non può essere influenzato dalle condizioni del canale.
2. Sorgente TCP – utilizza come protocollo di trasporto il *Transfer Control Protocol* (TCP). Il TCP a differenza dell'UDP, è un protocollo basato sulla connessione che offre correzione d'errore e che garantisce la consegna dei dati attraverso ciò che è conosciuto come *controllo di flusso* (determina quando il flusso di dati debba essere fermato, e quando i pacchetti di dati inviati in precedenza debbano essere rimandati a causa di problemi come *collisioni*, assicurando quindi la completa e accurata consegna dei dati). Di conseguenza la velocità di trasmissione del traffico TCP reagisce alle condizioni del canale in quanto TCP utilizza la finestra di congestione e il riconoscimento da parte del ricevitore. Quindi un attacco può essere eseguito sul traffico TCP sfruttando il meccanismo di prevenzione della congestione per ridurre la velocità di trasmissione della sorgente, o eventualmente provocare la chiusura del flusso.

Gli attacchi *downlink* sono relativamente meno intuitivi e richiedono un maggiore "impegno" da parte dei *cheater* per aumentare la loro larghezza di banda, e da parte dell'AP per rilevare i *misbehavior*.

Nella figura 2.1 viene rappresentato il tipico scenario in cui due stazioni mobili M e M_c sono collegate a Internet tramite l'AP. M e M_c scaricano file di grandi dimensioni da due server remoti, S e S_c.

Entrambi i download utilizzano il *File Transfer Protocol* (FTP) – protocollo di trasmissione dati tra *host* basato su TCP. Per aumentare la sua velocità di download, il *cheater* (M_c) può usare le seguenti due tecniche per ridurre il data rate di S e guadagnare più banda per se stesso all'AP (o ad ogni collo di bottiglia tra server e AP):

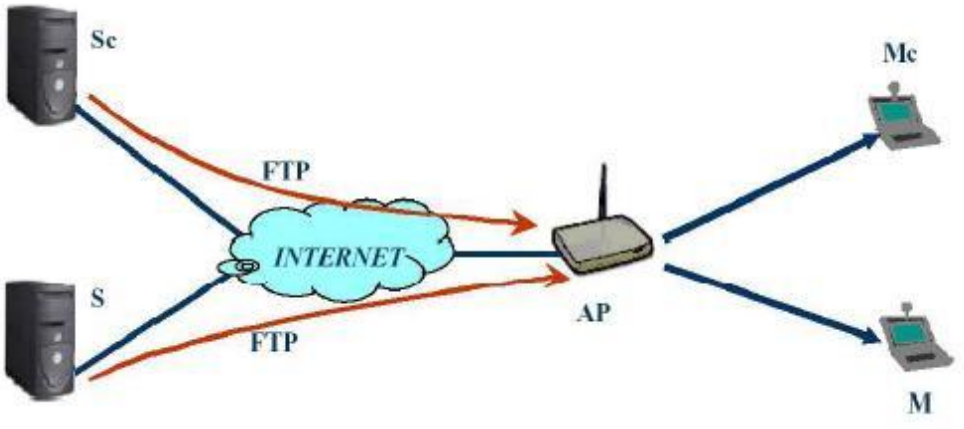


Figura 2.1 Scenario di simulazione [8]

- M_c esegue il *jamming* dei frame TCP-ACK da *M* all'AP, in modo che questi non raggiungano mai il server *S*. Siccome i frame TCP-ACK vengono persi (a causa del *jamming*), *S* diminuisce la sua velocità d'invio dei dati, utilizzando il controllo di congestione TCP, e finisce per interrompere la connessione. Presso l'AP, *M* risulta con una banda decrementata, portando un aumento della velocità di trasmissione dati tra S_c e M_c. In questa tecnica, l'AP può sentire le collisioni (causate dal *jamming*) e riesce a rilevare M_c in base al numero di ritrasmissioni di *M*.
- Un'altra opzione per M_c consiste nel eseguire il *jamming* dei frame dell'AP destinati a *M*, riducendo così la velocità dati di *S* senza essere sentito dall'AP. Siccome i pacchetti di M_c condividono la stessa coda con i pacchetti di *M* presso l'AP, mentre i frame persi a causa del *jamming* vengono ritrasmessi ripetutamente dall'AP a *M*, i pacchetti di M_c vengono trattenuti nella coda, quindi la velocità di trasmissione dati (da S_c) diminuisce allo stesso modo. Per evitare ciò, M_c invia dei frame MAC-ACK "contraffatti", in nome di *M* per i pacchetti persi. In questo modo, evita ritrasmissioni dell'AP, e riduce la velocità di trasmissione dei dati da *S.* Inoltre, come si vedrà nei risultati della simulazione, M_c può anche eseguire il *jamming* solo di una parte dei pacchetti da AP a *M*, risparmiando cosi potenza e rendendo ancora più difficile l'individuazione.

2.2 Una possibile soluzione: DOMINO

Ci sono vari approcci per affrontare le tecniche *misbehavior* presentate in precedenza. In [8] è presentato un sistema di rilevazione: *Detection of Greedy Behavior in the MAC Layer of IEEE 802.11 Public Networks* (DOMINO).

Le caratteristiche principali di DOMINO sono la sua integrazione all'interno o in prossimità dell'AP (senza interferire con le funzioni di quest'ultimo), la compatibilità con le reti esistenti, la facilità di adattarlo alle future versioni dello standard IEEE 802.11 tramite modifiche, e la sua capacità di individuare i *cheater*. Il componente in cui DOMINO deve essere installato è il *controller* dell'*hotspot*, che fornisce controllo degli accessi e può gestire diversi *Access Point* [9]; tuttavia, si assume in seguito, senza perdita di generalità, che il controller è incorporato nell'AP.

Data la varietà e i numeri di attacchi, DOMINO ha sviluppato un'architettura modulare presentata nella figura 2.2.

DOMINO raccoglie periodicamente tracce di traffico, dalle stazioni utente attive, durante brevi intervalli di tempo chiamati *periodi di monitoraggio*. Una serie di test ciascuno con l'obiettivo d'individuare una particolare tecnica di *misbehavior*, determina se il traffico analizzato presenta anomalie di comportamento; queste anomalie possono essere considerate "conseguenze" del corrispondente *misbehavior*. I risultati di questi test vengono poi inseriti in un *Decision Making Component* (DMC) che stabilisce se una data stazione è *cheater* oppure no. Se è così, il controllo viene passato al meccanismo di gestione dei *misbehavior*, che dipende dalla politica del *Wireless Internet Service Provider (WISP)* – fornitore di servizi Internet per reti wireless. Il WISP può decidere come reagire in presenza di utenti *cheater*: può ridurre la qualità del servizio, o addirittura interromperlo secondo la durata e la gravità del *misbehavior*.

L'architettura modulare presenta diversi vantaggi. Ad esempio, i test così come i componenti di decisione possono essere implementati usando algoritmi diversi a seconda della tollerabilità e della complessità richiesta. Un altro vantaggio è che possono essere aggiunti nuovi test per individuare potenziali tecniche di *misbehavior* non ancora scoperte.

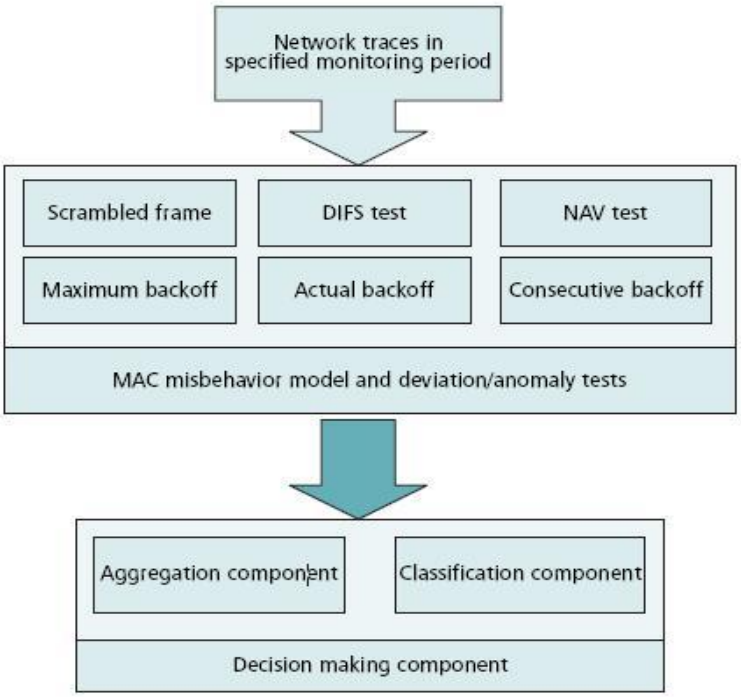


Figura 2.2 Architettura modulare di DOMINO [8]

Ogni test consiste di due componenti: *Deviation Estimation Component* (DEC) e *Anomaly Detection Component* (ADC).

Il DEC è in genere un test statistico che determina la quantità di deviazione del comportamento di una stazione, dedotta dalla traccia del suo traffico, rispetto ad un modello di comportamento previsto (ricavato osservando il comportamento dell'AP o delle altre stazioni attive nel corso di periodo di monitoraggio).

L'ADC utilizza la deviazione misurata dal DEC, per poter giudicare il tipo di comportamento di una stazione.

Il DMC mette insieme le decisioni parziali dei vari test in modo da valutare il comportamento di una data stazione nell'ultimo periodo di monitoraggio. Seguendo l'approccio modulare, il DMC è diviso in due moduli: una componente di aggregazione *Aggregation Component* (AC) e una componente di classificazione dei comportamenti *Behavior Classification Component* (BCC). Ancora una volta, l'implementazione di ciascuna di esse può essere flessibile. Nel modo più semplice, l'AC può essere implementato come

un OR dei risultati booleani dei vari test. Ciò significa che se una stazione cerca di imbrogliare utilizzando uno dei metodi descritti, sarà rilevata come *cheater*. In alternativa, l'AC può fare una somma pesata dei vari risultati dei test; questa somma viene poi normalizzata a 1 e paragonata ad una soglia. I pesi possono essere scelti per indicare sia l'affidabilità di un dato test di rilevamento, sia la gravità del *misbehavior* corrispondente. Ad esempio, un test il cui risultato non può essere influenzato da fattori quali le condizioni del canale, avrebbe un peso maggiore di un test che è più vulnerabile a queste condizioni. Il DEC è specifico per ogni test, mentre nell'ADC, test differenti possono utilizzare implementazioni simili o diverse. Pertanto, nella descrizione di ciascun test, ci si concentrerà sull'algoritmo che sta dietro al corrispondente DEC.
I test riportati di seguito utilizzano la seguente struttura:

if $condition_x$ is true **then**

$output_x := 1$

else

$output_x := 0$

dove *x* indica il numero del test. Tutti i test descritti vengono effettuati su ciascun campione di dati raccolti con successo per una stazione M_i durante l'ultimo periodo di monitoraggio; se viene rilevata una tecnica *misbehavior*, il controllo su M_i viene interrotto in quanto non sono necessarie ulteriori analisi. Per chiarezza, il funzionamento dei test verrà presentato di seguito su un singolo campione di dati.

2.2.1"*Scrambled frames*"

Questo test mira ad individuare le tecniche *misbehavior* che si basano sull'applicazione dello *scrambling* sui frame.
Al fine di aumentare la sua larghezza di banda un *cheater* deve eseguire lo *scrambling* su una percentuale relativamente elevata di frame CTS, ACK, o DATA mandati da altre stazioni. Come risultato, il numero medio delle sue ritrasmissioni sarà inferiore a quello

delle altre stazioni, e può essere rilevato utilizzando il Test1 (in questo test, come anche in quelli che seguono, se la disuguaglianza vale, significa che un attacco - *selfish misbehavior* è probabilmente in corso).

Test 1: Scrambled frames

$condition_1 : num\_rtx(M_i) < \Phi \times E_{j \neq i}[num\_rtx(M_j)]$

In questo test, *num_rtx*(M_i) è il numero di volte in cui la stazione *M* ha ritrasmesso il suo ultimo frame ricevuto con successo dall'AP. ϕ è un parametro di tolleranza, con un valore compreso tra 0 e 1, ed è applicato al numero medio di ritrasmissioni di tutte le "altre" stazioni, $E_{j \neq i}$.

DOMINO è in grado di rilevare una ritrasmissione osservando un numero di sequenza ripetuto nell'intestazione del frame RTS o DATA quando i corrispondenti frame CTS oppure ACK hanno subito lo *scrambling*.

Nel caso dei frame DATA, si potrebbe affermare che l'AP non è in grado di distinguere le ritrasmissioni a causa dello *scrambling*. Tuttavia, il *cheater* non può eseguire lo *scrambling* sull'intestazione di quest'ultimi, altrimenti non potrebbe sapere se un dato frame è destinato a se stesso.

Per ipotesi, il *cheater* è considerato razionale, cioè adotta un comportamento egoistico solo quando trae dei vantaggi; perciò la sua identità può essere individuata dal numero di ritrasmissioni. Il cheater non può modificare questo numero per ingannare, perché il mittente (l'AP in questo caso) reagirà davanti ad un numero di sequenza sbagliato: scartando il frame (se il numero non è maggiore rispetto alla precedente che ha registrato), oppure inviando un frame fuori ordine (se il numero è più grande dell'ultimo valore registrato), a seconda dell'implementazione specifica della scheda wireless. Si ipotizza anche il fatto che l'attaccante non può cambiare l'indirizzo MAC della sua stazione, a causa di un meccanismo di autenticazione (IEEE 802.11i) che impedisce l'utilizzo arbitrario di indirizzi MAC. Un esito errato (falsi positivi) di questo test potrebbe essere causato dalle cattive condizioni del canale che portano alla perdita di frame e quindi alla ritrasmissione. Per evitare questa trappola, l'AP può prendere in considerazione il *Re-*

ceived Signal Strength Indicator (RSSI – misura la potenza del segnale ricevuto) delle stazioni quando vengono rilevati dei malfunzionamenti.

2.2.2 L'individuazione dei parametri di protocollo manipolati

Nei prossimi paragrafi si fa riferimento alle tecniche *misbehavior* che si basano sulla modifica dei parametri del protocollo: ci si concentra soprattutto sulla manipolazione del *backoff* perché è la tecnica più facile da implementare e la più difficile da individuare.

2.2.2.1 "*Shorter than DIFS*"

L'AP può monitorare il periodo di inattività dopo l'ultimo ACK e distinguere ogni stazione che trasmette prima del periodo DIFS richiesto. Dopo aver osservato questo malfunzionamento ripetutamente per diversi frame dalla stessa stazione, l'AP può prendere una decisione affidabile (Test 2).

Test 2: Shorter than DIFS
$condition_2 : idle\_time\_after\_ACK(M_i) < DIFS$

2.2.2.2 "*Oversized NAV*"

Misurando la durata effettiva di una trasmissione (compresi i DATA, ACK, e i facoltativi RTS/CTS) e confrontandola con *duration* - il valore del campo che sta nell'intestazione del frame RTS o DATA, l'AP è in grado di rilevare una stazione che imposta periodicamente il campo *duration* (e quindi il NAV delle stazioni di ascolto) a

valori molto grandi. Nel Test 3, il parametro di tolleranza A (maggiore di 1) assicura che l'AP non sbaglia identificare la stazione cheater.

Test 3 Oversized NAV
$condition_3 : A \times actual\_duration(M_i) < duration(M_i)$

2.2.2.3 *"Backoff manipolation"*

L'identificazione della manipolazione del *backoff* è dovuta all'esecuzione di tre test:*"Actual backoff "*, "*Consective backoff "*, e *"Maximum backoff "*.

2.2.2.3.1 "*Actual backoff"*

Questo test (Test 4) consiste nella misurazione degli *actul backoff*.
Sia *M*, la stazione a cui viene applicata la misurazione dell'*actual backoff*; in genere, le trasmissioni di *M* sono alternate a una o più trasmissioni di altre stazioni (incluso l'AP). Come definito dallo standard, la trasmissione prevede, in aggiunta al frame DATA, anche la presenza dei frame di controllo: RTS, CTS, e ACK, e dei periodi d'inattività SIFS e DIFS. Il valore dell'*actual backoff* misurato è dato dalla somma di tutti gli intervalli d'inattività registrati tra due trasmissioni di *M*, senza includere gli intervalli temporali tra frame successivi, come illustrato nella figura 2.3.
La procedura di esecuzione del test può essere riassunta come segue:

- Se tra due trasmissioni eseguite dalla stazione *M* non ci sono collisioni, si considerano coincidenti il tempo d'inattività di *M* e il *backoff* (nonostante possa accadere che si tratti di un ritardo tra frame di *M*, se quest'ultimo invia pacchetti a bassa velocità). Il valore dell'*actual backoff* si ottiene calcolando la somma, come descritto prima.

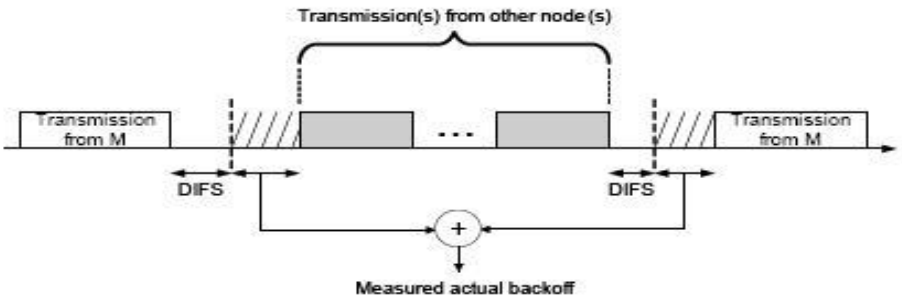


Figura 2.3 Actual Backoff [8]

- Se si verifica una collisione, è difficile conoscere l'identità dei mittenti dei frame implicati nella collisione. In altre parole, risulta problematico sapere per quali, delle stazioni di cui si stanno misurando gli *actual backoff*, si debba aggiornare la somma. Per evitare tale complessità, si stabilisce che le collisioni non vengano prese in considerazione. Nel caso esse si verifichino, infatti, né il *backoff* corrente, né quello successivo vengono misurati per alcuna stazione. Come precisato nella sezione 1.3.2.1, le stazioni che sentono l'intestazione dei frame con CRC errato, a causa di una collisione, ritardano la loro trasmissione di una durata prefissata indicata con EIFS. Quest'ultimo, non interferisce con le misurazioni in quanto le collisioni non vengono contate.

Test 4 Actual backof
$condition_4 : B_{ac}[M_i] < \alpha_{ac} \times B_{acnom}$

Nel Test 4, $B_{ac}[M_i]$ indica l'*actual backoff* medio (osservato dall'AP) della stazione M_i. B_{acnom} rappresenta il valore nominale di *backoff*; esso è uguale al *backoff* medio dell'AP, ipotizzando che ci sia abbastanza traffico da calcolare questo valore. α_{ac} ($0 < \alpha_{ac} \leq 1$) è un parametro espresso in percentuali, configurabile in base alle esigenze di stima del tasso di errore desiderato (identificazione corretta o sbagliata).

Durante le collisioni, non vengono raccolti dati, perciò il test di *actual backoff* misura valori nell'intervallo [0, $CW_{min} - 1$]. A causa del suo meccanismo, questo test non riesce a identificare il *misbehavior* se il *cheater* provoca ritardi tra i frame: *interframe delay* (ad esempio immaginare una sorgente TCP che utilizza il controllo di congestione).

In effetti, il test misura questi ritardi invece dei *backoff* perché somma i periodi d'inattività tra le trasmissioni della stessa sorgente (figura 2.3). La soluzione a questo problema è fornita dal test *Consecutive Backoff*.

2.2.2.3.2 *"Consecutive Backoff"*

Nella figura 2.4 viene illustrato il funzionamento di questo test (Test 5), che opera nel caso di sorgenti con ritardi interframe. In pratica, questa situazione si registra soprattutto nelle sorgenti TCP, dove il ritardo è generalmente dovuto al controllo di congestione.
I test di *actual backoff* non producono valori corretti (come spiegato in precedenza) per questo tipo di sorgenti, e di conseguenza non sono in grado di rilevare i potenziali *cheater*. Ad esempio, si consideri la stazione *M* una sorgente TCP. Si supponga inoltre che sul canale condiviso, la trasmissione di *M* venga alternata con la trasmissione di almeno un'altra stazione, ovvero che tra due frame inviati da *M* e separati da un ritardo dovuto al livello di trasporto, esista almeno un frame appartenente a un'altra stazione. Quindi, se l'AP osserva due frame consecutivi da *M*, può considerare il tempo di inattività tra essi come un *backoff* in aggiunta al DIFS obbligatorio. Tali frame consecutivi risultano dalla contesa sul canale che costringe *M* ad accodare pacchetti al livello MAC, anche se sono stati separati da un ritardo ai livelli superiori. In questa situazione, *M* potrebbe avere dei vantaggi nel manipolare i valori di *backoff*, tali da diminuire la sua coda al livello MAC.
DOMINO è in grado di raccogliere campioni significativi di valori di *backoff* scelti da *M*; questi campioni vengono chiamati *backoff consecutivi*.

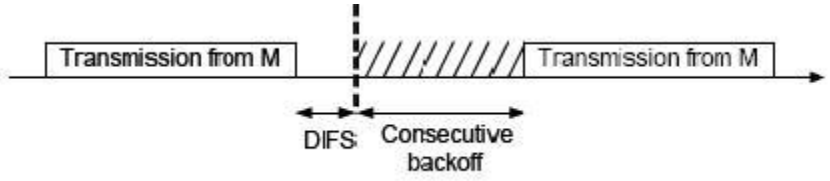


Figura 2.4 Consecutive Backoff [8]

L'assunzione fatta sopra sul livello di trasporto è realistica. Infatti, se il traffico sul canale è sufficientemente lento da invalidare questa ipotesi, e cioè, se *M* potesse inviare frame consecutivi, separati da un ritardo in aggiunta al *backoff* e DIFS, l'attacco sarebbe inutile: perché la riduzione del *backoff,* in questo caso, non influisce sul ritardo al livello superiore. Di conseguenza, rilevare il *misbehavior* non è necessario.

Test 5: Consecutive backoff
$condition_5 : B_{co}[M_i] < \alpha_{co} \times B_{conom}$

Come per il test precedente, la media dei valori raccolti $B_{co}[M_i]$ è confrontata con una percentuale α_{co} ($0 < \alpha_{co} \leq 1$ in generale; α_{co} = 90% nelle simulazioni che saranno descritte tra breve) del valore nominale B_{conom}. Quest'ultimo rappresenta la media dei *backoff* consecutivi all'AP quando sono disponibili sufficienti dati. In caso contrario, viene sostituito con un valore analitico $E[B_{co}]$ (vedi [6]).

2.2.2.3.3 *"Maximum backoff"*

Come precisato nella sezione 1.3.2.2, la contesa tra le stazioni avviene secondo l'algoritmo di backoff esponenziale, il quale prevede che ciascuna stazione selezioni in modo casuale valori di *backoff* nell'intervallo $[0, CW - 1]$ (dove *CW* dipende dal numero di ritrasmissioni). Il massimo valore di *backoff* selezionato su un insieme di frame inviati da una determinata stazione, perciò, dovrebbe essere maggiore uguale a $CW_{min} - 1$, se il numero di campioni è abbastanza grande. DOMINO utilizza questa proprietà per individuare le stazioni il cui *backoff* massimo su un insieme di campioni è più piccolo di un valore di soglia $threshold_{maxbkf}$. Chiaramente, esiste un *trade-off* tra il numero di campioni e la soglia; cioè se si aumenta la soglia (il valore più grande è CW_{min}), bisogna aumentare il numero di campioni di *backoff* per ottenere più valori distinti e quindi per evitare errori (falsi positivi). Nelle simulazioni (presentate di seguito)

viene utilizzata una soglia pari a $CW_{min}/2$, e quindi il test funziona se il valore della finestra di contesa ridotta viene scelto nell'intervallo $[0, CW_{min}/2 - 1]$.
Purtroppo, questo test non è affidabile: basta che un *cheater* riesca a controllare in ogni campione almeno un valore di *backoff* rendendolo maggiore uguale alla soglia; anche le condizioni del canale possono produrre un risultato simile e quindi far fallire il test. Ecco perché, il controllo del massimo *backoff* è soltanto ausiliario per le due prove precedenti.

2.2.3 *"Scrambled TCP packets with forged MAC ACKs"*

Questo tipo di attacco è ancora più difficile da individuare, in quanto l'AP non può sentire le collisioni e il *cheater* invia pacchetti MAC-ACK "contraffatti" (non contengono l'indirizzo della sorgente) corrispondenti ai frame cui è stato applicato lo *scrambling*. In questo caso, DOMINO non può fidarsi del numero di ritrasmissioni per decidere se si tratta di un *misbehavior*. Per affrontare questa tecnica DOMINO ha due meccanismi complementari che implementano i componenti DEC e ADC dell'architettura del sistema nel test. Prima di tutto, DOMINO misura la soglia del flusso di *downlink* (questo è il DEC), poi se viene individuato un ricevitore a cui è destinato più traffico, DOMINO lo sospetta come un potenziale *cheater*. Come sarà spiegato in seguito, la soglia non è una metrica di rilevazione affidabile a causa della diversità degli interessi degli utenti. Di conseguenza, DOMINO utilizza *Dummy Frame Probing* (DFP) per confermare o negare il sospetto. DFP consiste nel mandare frame finti (*dummy*) a stazioni virtuali (non esistenti). Se uno di questi frame viene seguito da un MAC-ACK, è un indizio dell'esistenza di un *cheater* nella rete. L'osservazione a lungo della soglia serve a determinare l'identità del *cheater*. Il DFP associato al confronto con la soglia costituiscono l'ADC.
Un *cheater* intelligente potrebbe crearsi una lista delle stazioni virtuali (ricordandosi di non rispondere con MAC-ACK a tali stazioni) per evitare di rispondere ai frame *dummy*. Per identificare il *cheater* l'AP potrebbe generare degli ACK fasulli (finti), in

questo modo il *cheater* non riuscirebbe a distinguere facilmente tra frame *dummy* e gli altri. Un *cheater* è interessato ad attaccare solo connessioni con alto *throughput*. Di conseguenza, i frame *dummy* dovrebbero essere generati di tanto in tanto a soglie alte, come trappola per i cheater. Il vantaggio dei frame *dummy* è che rappresentano la maggioranza dei test discriminanti: un semplice campione è sufficiente per identificare i sospetti, anche se sono generati durante piccole quantità di tempo (dal 5% al 10%). Perciò l*'overhead* (la spesa) è piccolo.

2.3 Studio delle prestazioni di DOMINO

Lo studio delle prestazioni di DOMINO è stato eseguito utilizzando il simulatore ns-2 [10]. Dato che la tecnica *misbehavior* utilizzante lo *scrambling* è facile da individuare attraverso il numero di ritrasmissioni, viene analizzato in dettaglio solo il meccanismo d'identificazione che riguarda la manipolazione del *backoff*.

2.3.1 Simulazione su traffico uplink

In [6] si presenta la seguente situazione: otto stazioni inviano traffico UDP o TCP all'AP, che a sua volta genera traffico simile a una delle stazioni e lo manda a una stazione ricevitrice addizionale (non mostrata nella figura 2.5). La distanza tra ciascuna stazione e l'AP e di 50m. Tutte le stazioni sono in visibilità radio tra di loro (cioè non si verifica il problema del terminale nascosto).

1. Traffico UDP. Oltre al *cheater*, ci sono sette stazioni che inviano *traffico CBR – Constant Bit Rate* (il tasso nominale è di 500 byte/pacchetto, 200 pacchetti/s), il cheater è anche una sorgente CBR (cioè invia un pacchetto ogni *T* secondi). La tecnica misbehavior consiste nel ridurre la finestra di contesa. In qualsiasi slot di inattività vi è almeno un pacchetto pronto per la trasmissione da parte di una del-

le stazioni concorrenti. Il tempo trascorso tra due trasmissioni dalla stessa stazione (alternato con le trasmissioni di altre stazioni) è quindi dovuto solo al backoff scelto dal protocollo IEEE 802.11.

2. Traffico TCP. Ciascuna delle otto stazioni esegue un'applicazione FTP; una stazione provoca il jamming dei pacchetti TCP e produce i corrispondenti MAC-ACK fasulli. Questo caso mostra l'effetto dei ritardi d'inter-frame (dovuti al controllo di congestione del TCP) sulla misurazione del backoff. Questo è lo scenario più realistico.

In entrambi i casi l'AP genera traffico simile a quello di una stazione, cioè, CBR nel primo caso e FTP nel secondo.

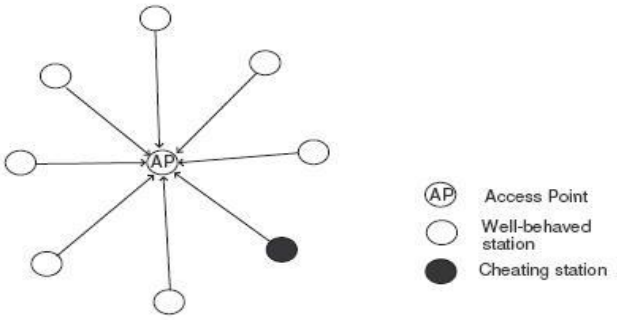


Figura 2.5 Scenario di simulazione[8]

Tenendo conto dell'effetto fading presente nei canali reali, le simulazioni sono state effettuate usando il seguente modello del canale:

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta log\left(\frac{d}{d_0}\right) + X_{dB};$$

$P_r(d)$ rappresenta la potenza media ricevuta alla distanza d, d_0 è una distanza di riferimento, β è l'esponente di perdita di percorso (*path loos)*, mentre X_{dB} è una variabile aleatoria Gaussiana a media nulla e deviazione standard σ_{dB}. È stato utilizzato $\beta = 2$ (per la propagazione nello spazio libero) e $\sigma_{dB} = 4$.

Ogni dieci simulazioni, di durata 110s ciascuna, viene calcolata la media dei risultati. Il periodo di monitoraggio scelto è di 10s, e corrisponde alla decisione ("cheater" piuttosto

che “well-behaved”) da parte dell’AP per ogni stazione. Dunque, ogni punto dei grafici (figura 2.6, figura 2.7) è una media di oltre 100 campioni, con un intervallo di confidenza (intervallo di valori plausibili) del 95%; i primi 10s di ogni simulazione rappresentano un periodo di inizializzazione, in cui le misure non sono prese in considerazione nell'elaborazione dei risultati.
In seguito, il *misbehavior coefficient* rappresenta la quantità di malfunzionamento. Un coefficiente di misbehavior pari a *m* indica che la stazione corrispondente utilizza una finestra di contesa fissa pari a $(1 - m) \times CW_{min}$ e sceglie allora il suo backoff da questa nuova finestra. Quindi, *m* = 0 rappresenta l’assenza di comportamento scorretto, mentre *m* = 1 indica che la stazione trasmette senza alcun *backoff*.

2.3.1.1 Impatto del *misbehavior* sul *throughput*

Prima di presentare le prestazioni di DOMINO, si possono confrontare i valori di *throughput* di una stazione che si comporta bene (well-behaved) e di un cheater in entrambe le situazioni: traffico UDP e TCP.

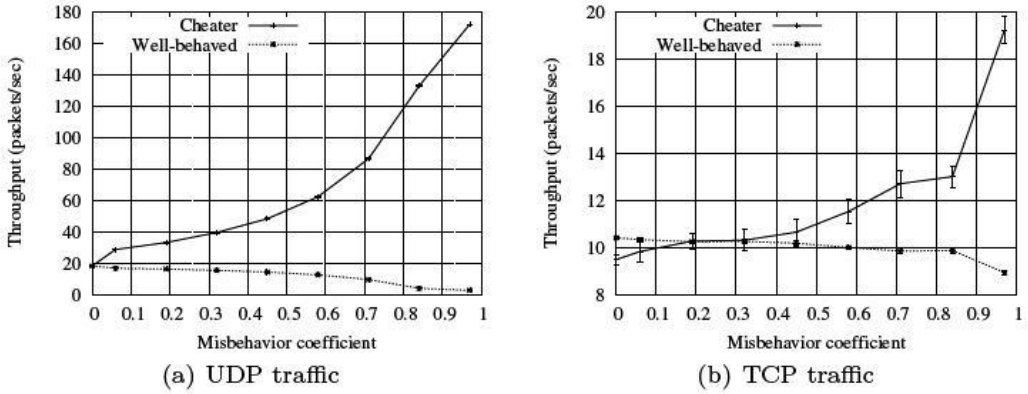


Figura 2.6 Confronto tra il throughput delle stazioni *cheater* e stazioni *well-behaved*[8]

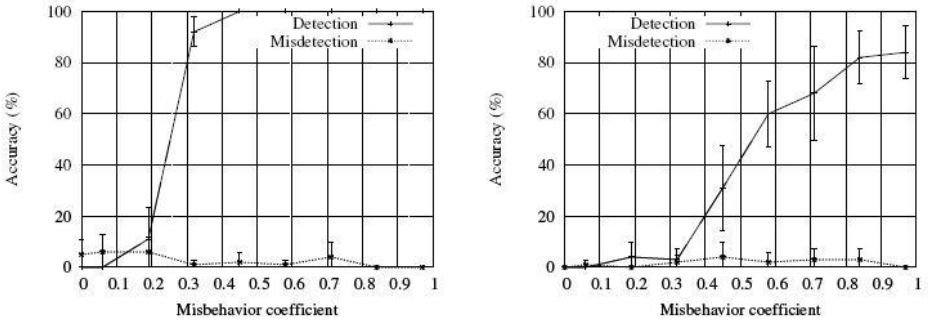
Nella figura 2.6 si può osservare l’andamento del throughput al variare del coefficiente di misbehavior. I dati ottenuti dai test evidenziano come l’incremento del throughput sia

meno significativo nel caso di sorgenti TCP. Ciò è dovuto ai meccanismi di controllo della congestione in TCP.

2.3.1.2 Prestazioni dell'*Actual Backoff*

Dal grafico di simulazione si possono trarre le seguenti osservazioni:

- Nel caso di traffico UDP (figura 2.7(a)), il test ha buone prestazioni, dato che c'è sempre almeno un frame pronto per la trasmissione da parte di ogni stazione. Di conseguenza il tempo d'inattività del canale tra due trasmissioni di una stazione è dovuto solo al meccanismo di backoff (in aggiunta al DIFS).
- Nel caso del traffico TCP, è difficile distinguere se una decisione da parte dell'AP è stata presa in modo corretto (dato che le curve sono praticamente sovrapposti con l'asse *x*, quindi, il corrispondente grafico non fornisce alcun informazione importante, motivo per cui non è stato riportato). La scarsa precisione può essere spiegata dal fatto che l'actual backoff misurato è in realtà il periodo d'inattività (senza includere cicli di trasmissione es: DATA-SIFS-ACK-DIFS, delle altre stazioni) tra due trasmissioni alternate da parte di una stessa stazione, che è uguale in questo caso al ritardo tra le trasmissioni di frame dalla sorgente. Questo ritardo è dovuto ai meccanismi di controllo della congestione del TCP.



(a) *Actual backoff* test in the UDP traffic case. (b) *Consecutive backoff* test in the TCP traffic case.

Figura 2.7 Prestazioni dei test Actual backoff e Consecutive backoff [8]

2.3.1.3 Prestazioni del *Consecutive backoff*

Nel caso di traffico UDP, i risultati del test non danno alcuna informazione (le curve sono sovrapposte con l'asse delle ascisse e pertanto il grafico è stato omesso). Ciò accade perché la media dei backoff consecutivi misurati diminuisce rapidamente con il numero di stazioni.
Nel caso del traffico TCP (figura 2.7(b)), il test dà buoni risultati. Ciò è dovuto alla presenza di altri sorgenti che non permettono alla sorgente con il ritardo interframe (indotto dal controllo di congestione) di trasmettere due frame consecutivi senza accodare il secondo. Vale a dire: il ritardo non influisce sul tempo di inattività tra due trasmissioni consecutive non alternate della sorgente. Altrimenti, se non vi è nessun frame pronto in coda, un'altra sorgente prende il controllo del canale e trasmette almeno un frame tra due frame successivi della prima sorgente.

2.3.1.4 Conclusioni delle simulazioni sul traffico uplink

La descrizione dei test di *actual backoff* e *consecutive backoff*, così come i grafici presentati finora, hanno dimostrato che le prestazioni dei test variano con il tipo del traffico. Un meccanismo completo d'individuazione, quindi è formato della combinazione di entrambi i test.
Poi, fintanto che c'è abbastanza traffico sul canale da soddisfare le richieste del Test 5, soltanto il tipo di traffico del mittente determina quale test funziona al meglio, quindi in scenari misti di traffico (TCP/UDP ad alta velocità), i misbehavior possono anche essere individuati con precisione. Se non c'è traffico sul canale a sufficienza, il misbehavior non produce benefici sostanziali di throughput, quindi la sua individuazione non è necessaria.

2.3.2 Simulazioni su traffico downlink

Con riferimento alla Figura 2.1, [8] presenta la seguente situazione: M_c esegue il jamming dei pacchetti TCP del server *S* per *M*, trasmessi dall'AP, e invia MAC-ACK, in nome di *M*.

M_c può eseguire il jamming anche solo di una percentuale *X* del traffico downlink (quando $X = 1$, M_c esegue il jamming di tutti i frame da AP a *M*). In questo modo M_c risparmia energia e rende più difficile la sua l'individuazione da parte dell'AP. La percentuale di pacchetti, a cui è stato applicato il jamming, può essere distribuita in modo uniforme nel tempo, o applicata a scatti. In quest'ultimo caso, M_c esegue il jamming del canale per *D* secondi ogni periodo *T*, con $D<T$ (quindi $X=D/T$). Questo metodo è conosciuto come "bursty jamming".

Le prestazioni vengono analizzate in base ai throughput dei flussi TCP ricevuti dal cheater (M_c) e quelli del nodo well-behaved (*M*).

Le sorgenti S e S_c iniziano a trasmettere contemporaneamente, utilizzando il protocollo TCP-Reno [10] con pacchetti da 1000 byte. Per assicurarsi che il throughput raggiunga uno stato costante, si presume che il cheater inizia a eseguire il jamming 60s dopo. La media dei risultati viene calcolata dopo oltre 35 simulazioni.

Si distinguono tre fattori che aiutano ad aumentare il throughput di M_c:

- la riduzione del numero di flussi concorrenti che entrano nella coda dell'AP;
- la riduzione del tasso di collisione sul canale wireless (TCP-DATA trasmessi dall'AP con TCP-ACK trasmessi da *M* e M_c);
- la riduzione dei ritardi di accodamento all'AP (i pacchetti che hanno subito il jamming non sono ritrasmessi).

La figura 2.8(a) mostra il throughput di M_c e *M* in funzione della percentuale *X* di frame che hanno subito il jamming. Si osserva che eseguendo il jamming sul 30% dei frame è sufficiente a ridurre il throughput di *M* a zero, e ad aumentare il throughput di M_c alla velocità massima di dati disponibile.

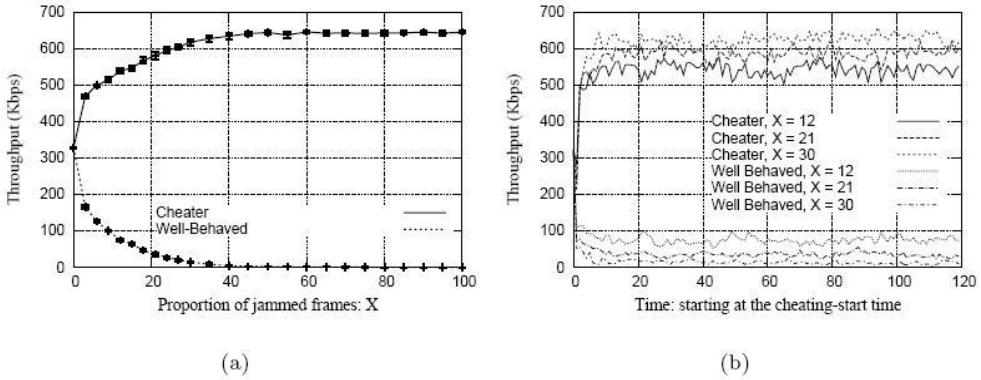


Figura 2.8 Andamento del throughput al variare della percentuale di jamming su traffico down link [8]

L'evoluzione del throughput di M_c ed *M* nel tempo è mostrata nella figura 2.8(b). Si nota che M_c ha un throughput basso quando inizia il jamming dei frame di *M* e produce MAC-ACK fasulli. Più tardi, questo overhead viene ridotto, in quanto *M* riceve throughput diminuito. Perciò, M_c esegue il jamming di meno frame e produce meno MAC-ACK fasulli, aumentando la sua efficienza. Tale periodo transitorio dura meno di 10s.

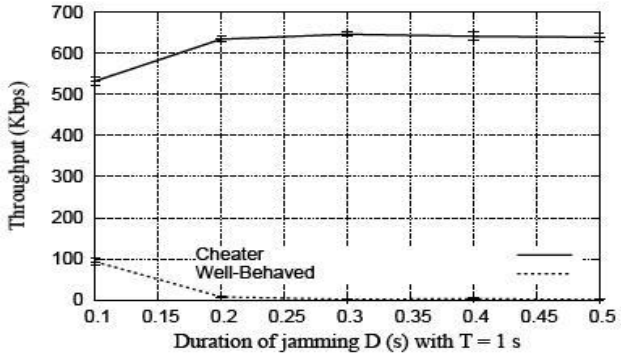


Figura 2.9 Andamento del throughput in presenza di "bursty jamming" su traffico downlink [8]

Nella figura 2.9 è presentata l'evoluzione del throughput quando il cheater applica il "*bursty jamming*". Utilizzando ad esempio T = 1s, [11], [12] la stessa percentuale *X=D/T* porta a un throughput maggiore per M_c e minore per *M*, rispetto al jamming normale (distribuito in maniera uniforme), rendendo così l'attacco ancora più devastante.

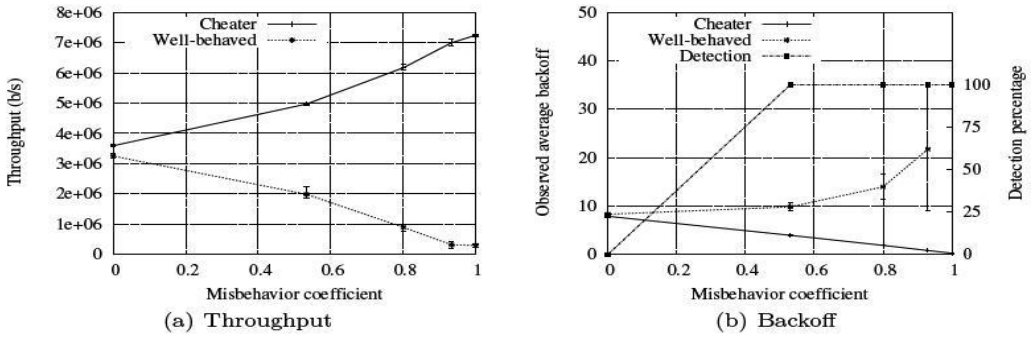


Figura 2.10 Risultati degli esperimenti: (a) Throughput. (b) Backoff. [8]

2.4 Sperimentare DOMINO

Per dimostrare la necessità e l'efficienza del sistema di identificazione proposto, in [8] viene studiata una tecnica misbehavior, basata sulla manipolazione del backoff, e per individuarla si utilizza un prototipo di DOMINO. L'esperimento comprende due stazioni che eseguono l'upload di traffico UDP all'AP e si ripete per diversi valori della finestra di contesa del cheater, tutti del tipo $2^n - 1$, con n un intero adeguato. Precisamente CW_{min} e CW_{max} vengono impostate a 0, 1, 3, 7, e 15 (il valore di default di CW_{min} è 15 mentre quello di CW_{max} è 1023 per le schede wireless utilizzate), che corrispondono ai coefficienti di misbehavior 1, 0.93, 0.8, 0.53, e 0 rispettivamente.

La figura 2.10 presenta il throughput e il backoff risultante delle varie stazioni, al variare del coefficiente di misbehavior.

Nella figura 2.10(a), si osserva come il cheater ottiene un throughput maggiore, a spesa delle stazioni well-behaved, con l'aumentare del misbehavior. I corrispondenti valori di backoff osservati sono illustrati nella figura 2.10(b). Quando la percentuale di misbehavior aumenta, la media dei backoff del cheater diminuisce (così aumentando la possibilità di ottenere il canale e aumentare il suo throughput). Ciò può essere facilmente individuato da DOMINO, come si nota dalla curva d'identificazione. Nel frattempo, la media dei backoff delle stazioni con un buon comportamento aumenta con la percentuale

di misbehavior (dovuto alle collisioni e all'ulteriore incremento della finestra di contesa); così si spiega il loro throughput diminuito.

2.4.1 *Throughput* come metrica d'indentificazione

Nonostante il throughput sembri la metrica più intuitiva per distinguere stazioni che utilizzano una quota maggiore di banda del canale, rispetto ad altre stazioni, esso non può essere utilizzato come metrica. Infatti, se due stazioni hanno differenti velocità di trasmissione dati e ritardi, come il VoIP rapportato alle sorgenti di streaming video, il throughput di quest'ultime sarà naturalmente molto più grande di quello del VoIP. Quindi, non è possibile affidarsi al throughput senza conoscere il tipo di applicazione in esecuzione su ogni stazione (ciò richiederebbe ad ogni stazione di dichiarare all'AP le applicazioni usate al momento della comunicazione, in violazione alla stratificazione protocollare).

Studi sperimentali, [13] e [14] hanno dimostrato che il throughput di una sorgente UDP in una rete wireless è influenzato da molti fattori come: l'overhead dei pacchetti, SNR (rapporto segnale rumore), l'hardware presente nella rete, il driver di periferica, e le implementazioni del protocollo di rete nel sistema operativo. Gli autori di [15] hanno provato che la diminuzione del bit rate di una singola stazione (a causa di un cattivo canale) riduce i bit rate di tutte le altre stazioni a valori vicini a quelli della stazione svantaggiata. Tutti questi fattori portano a grandi differenze di throughput anche tra le stazioni che inviano con la stessa velocità.

Le prestazioni del TCP su reti wireless sono state studiate sperimentalmente in [14]. I risultati mostrano che il TCP associato al protocollo MAC IEEE 802.11 produce una riduzione delle prestazioni. Ciò è dovuto alla finestra di congestione, ai meccanismi di recupero, alla dimensione del pacchetto, e ai valori di timeout del TCP, come pure agli ACK, ai limiti di tentativi di ritrasmissione, e al meccanismo di backoff del MAC IEEE 802.11. Quindi, anche se la correttezza delle reti wireless è stata valutata [9], [16], [17], generalmente usando l'indice di correttezza di Jain [18] (che a sua volta utilizza canale a

banda condivisa) il throughput non è la metrica ottimale per identificare il misbehavior in questo caso.

2.4.2 Il problema del terminale nascosto e DOMINO

Il problema del terminale nascosto può avere un effetto negativo su DOMINO. Ad esempio, se due stazioni *A* e *B* sono in visibilità per l'AP, ma nascoste tra di loro, *A* potrebbe percepire il canale inattivo, mentre l'AP lo sente occupato perché *B* sta trasmettendo. Come risultato, *A* continuerà a decrementare il suo contatore di backoff, trasmettendo poi un frame il cui backoff misurato all'AP appare inferiore al valore reale atteso. Nelle condizioni in cui questa situazione si verifica più volte, il meccanismo di rilevamento inizia erroneamente a sospettare *A*. Nonostante ciò, DOMINO riesce a migliorare le sue prestazioni, tramite la modifica del raggio di ricezione e attraverso la scelta appropriata dei valori di soglia di rilevamento (α_{ac} e α_{co} definite in 2.2.2.3.1 e 2.2.2.3.2).

2.4.3 Scelta dei parametri di rilevamento

La scelta dei parametri di rilevamento influisce sulle prestazioni di DOMINO. Dato che tale scelta dipende dalle condizioni in cui il sistema funziona, i parametri dovrebbero essere impostati durante l'installazione nell'AP. In effetti, gli amministratori di sistema devono eseguire una serie di test. Ad esempio scegliere un valore maggiore per il periodo di monitoraggio non è necessario, ma potrebbe produrre un ritardo nella risposta dei meccanismi di rilevazione. In realtà, gli amministratori potrebbero iniziare con valori di default e poi, aggiustarli in modo appropriato.

2.4.4 Periodo di monitoraggio

Per evitare di sovraccaricare l'AP con operazioni su ogni frame, i dati necessari per il rilevamento sono raccolti durante intervalli di tempo configurabili; alla fine di ogni intervallo, il meccanismo di rilevamento viene messo in esecuzione. Un vantaggio di tale metodo è la capacità di raccogliere più dati statistici e di conseguenza aumentare la precisione. Inoltre, si è mostrato in [16, 19] che l'algoritmo di backoff esponenziale dello standard IEEE 802.11 non è applicabile in maniera corretta per i brevi periodo. Ciò porterebbe a risultati incerti (falsi positivi), se le stazioni hanno un periodo di monitoraggio troppo breve (anche in assenza di misbehavior). In conclusione il periodo di monitoraggio deve essere abbastanza grande da contare sulla correttezza del backoff a lungo termine.

2.5 Sicurezza - *Adaptive cheating*

Con *adaptive cheating* si definisce l'insieme di tecniche misbehavior che sfruttano alcune conoscenze sulla modalità di funzionamento di DOMINO. Per esempio, un cheater potrebbe utilizzare tutte le tecniche descritte in 2.1 scambiandole frequentemente; di conseguenza DOMINO non riuscirebbe più a raccogliere abbastanza dati per individuare il malfunzionamento. Dato che il cheater non conosce i parametri di rilevamento, come ad esempio il periodo di monitoraggio e le soglie, è comunque difficile che si adatti al sistema di rilevamento, al fine di evitare di essere catturato. Se invece gli amministratori di sistema al momento dell'installazione, mantenessero i parametri di rilevamento di default, i cheater potrebbero utilizzare tali valori per adeguare le loro tecniche. Però, come specificato in 2.4.3, condizioni diverse richiedono valori dei parametri diversi, anche se si prendono in considerazione i valori di default. Quindi sarà difficile, per i cheater, di adeguarsi ai vari AP e ai loro parametri.
Un altro modo di ingannare DOMINO consisterebbe nell'utilizzare delle tecniche per disabilitare alcuni test. Ad esempio, un cheater potrebbe creare intenzionalmente colli-

sioni (eseguendo lo scrambling dei frame tramite segnali ad alta potenza) per produrre il fallimento del test *actual backoff*, oppure potrebbe non trasmettere mai due frame consecutivi non-alternati, in modo da produrre il fallimento del test *consecutive backoff*. Queste tecniche, però, aumentano l'overhead del cheater (ad esempio in termini di ritardi inter-frame) che potrebbe non essere compensato da un vantaggio significativo in termini di throughput, rispetto alle altre stazioni.

Capitolo 3

Teoria dei giochi in reti di telecomunicazioni

3.1 Introduzione

La teoria dei giochi, definita nel senso più ampio, è una raccolta di modelli matematici formulati per studiare situazioni di conflitto e cooperazione. L'oggetto di studio di questa disciplina è il *gioco*, definito da ogni situazione in cui:

- Ci sono almeno due *giocatori.* Nel presente lavoro, i giocatori sono rappresentati dai nodi wireless.
- Ogni giocatore ha un certo numero di possibili *strategie*, inteso come l'azione o l'insieme delle azioni che può scegliere di seguire.
- Le strategie scelte da ciascun giocatore determinano *l'esito* del gioco.
- Viene associato ad ogni possibile esito del gioco, un insieme di *payoff*, uno per ogni giocatore. Il payoff rappresenta il valore dell'esito (guadagno) di ogni giocatore.

La teoria dei giochi si basa sull'assunzione di *razionalità* dei giocatori, il che significa che ognuno segue una precisa strategia finalizzata alla massimizzazione del *payoff*. Per ogni giocatore sarà quindi possibile definire una *funzione di utilità*, che descrive le corrispondenze fra le varie strategie ed i guadagni possibili; ovviamente lo scopo del giocatore sarà avvicinarsi quanto più possibile al massimo di tale funzione. In generale, comunque, quando un giocatore sceglie una strategia, gli altri risponderanno con le strategie che giudicano più adeguate alla situazione, e così via finché non si raggiunge un qualche *equilibrio*, cioè una situazione in cui nessuno dei giocatori ha motivo di scegliere una strategia diversa nei passi successivi.

3.1.1 Tipologie di giochi

Si distinguono almeno due tipi di giochi: *cooperativi* e *non cooperativi*. I primi sono quelli in cui i partecipanti collaborano fra loro per raggiungere uno scopo comune; i secondi, invece, rappresentano i casi in cui ognuno vuole ottenere il massimo guadagno possibile e questo obiettivo può entrare in contrasto con quelli degli altri. È evidente che, volendo modellare degli utenti di reti senza fili, ognuno dei quali è interessato solo al buon funzionamento della propria connessione, il gioco sarà sicuramente *non cooperativo*.

In un gioco si distinguono tre elementi essenziali:

- L'insieme dei giocatori (*N*);
- L'insieme delle possibili azioni per ogni giocatore (*S*);
- L'insieme delle funzioni di utilità, una per ogni giocatore, che descrivono la corrispondenza fra le azioni e i guadagni, ovvero i payoff (*U*).

Nel caso in cui ogni giocatore conosca completamente tutte e tre le componenti, il gioco si definisce a *informazione completa*.

3.1.2 Rappresentazione del gioco

Un gioco solitamente viene rappresentato in *forma normale* oppure in *forma estesa*. Nel primo caso i vari giocatori costituiscono le diverse dimensioni di una matrice i cui elementi sono riempiti con i valori delle funzioni di utilità corrispondenti a una certa combinazione di azioni. In forma estesa, invece, il gioco viene visto come un albero in cui ogni livello rappresenta un passo del gioco; si arriva alle foglie quando il gioco termina. In questo senso, la forma estesa appare più adatta a descrivere l'evoluzione nel tempo di un gioco, mentre la forma normale è utile per conoscere i guadagni dei giocatori in corrispondenza delle varie strategie, pur non dando informazioni sull'effettivo evolversi del sistema.

3.1.3 Concetto di strategia

Come detto precedentemente, con *strategia* si può intendere sia una singola azione che un insieme di azioni. Nel caso in cui ad ogni passo, ciascun giocatore sceglie con probabilità unitaria una strategia fra quelle possibili, e assegna probabilità nulla a tutte le altre: *S* è detto insieme delle *strategie pure*. Se invece, ad ogni passo la scelta di una strategia viene fatta con una certa probabilità, si parla di *strategie miste*. L'insieme delle strategie scelte da tutti i giocatori, ad ogni passo, costituisce un *profilo* di strategie, notato con $s = \{s_1, s_2, \dots, s_N\}$, per il quale l'utente *i*-esimo otterrà un guadagno dato da $u_i(s)$.

3.1.4 Concetto di dominanza

Una volta dati tutti gli elementi del gioco, è necessario conoscere degli strumenti per poterlo risolvere, predicendo verso quale equilibrio evolverà, e se tale equilibrio esista o meno. Il metodo più semplice per risolvere un gioco consiste nell'applicare il concetto di *dominanza forte* (*strict dominance*).

Date due strategie s_i ed s_i' del giocatore *i*-esimo, si dice che s_i' è *fortemente dominata* da s_i se:

$$u(s_i', s_{-i}) < u(s_i, s_{-i}), \forall s_{-i} \in S_{-i}$$

dove *–i* indica l'insieme di tutti i giocatori tranne l'*i*-esimo; questi giocatori costituiscono quindi l'insieme degli avversari di *i*; perciò S_{-i} rappresenta l'insieme delle azioni degli avversari di *i*.

In pratica, possono essere eliminate dal gioco quelle azioni che non verranno mai eseguite perché non sarebbero convenienti in termini di guadagno. In questo modo si riduce di molto lo spazio delle strategie e diventa più semplice rilevare l'eventuale presenza di un equilibrio. Nei casi in cui la condizione di dominanza stretta sia troppo restrittiva, si può risolvere il gioco per *dominanza debole* (*weak dominance*); si verifica, cioè, la

disuguaglianza in senso debole, a patto che esista almeno un $s_{-i} \in S_{-i}$ per il quale essa valga in senso stretto. Questi metodi, comunque, aiutano la risoluzione di giochi particolarmente semplici: nella maggior parte dei casi non possono essere usati e bisogna introdurre il concetto di *miglior risposta* (*best response*), che porta poi alla definizione di *equilibrio di Nash*.

3.1.5 Equilibrio di Nash

La miglior risposta $br(s_{-i})$ del giocatore i in conseguenza al profilo s_{-i} è una strategia s_i tale che:

$$br(s_{-i}) = arg \max_{s_i \in S_i} u_i(s_i, s_{-i}).$$

Se si considera ad esempio un gioco con due partecipanti, e le loro due strategie sono migliori risposte l'una per l'altra, allora si intuisce che nessuno dei due avrà alcun interesse a non giocare quella strategia.
Un profilo di strategie costruito in questo modo costituisce un *equilibrio di Nash*. Formalmente, il profilo s^* è un equilibrio di Nash se per *ogni* giocatore vale:

$$u(s_i^*, s_{-i}^*) \geq u(s_i, s_{-i}^*), \forall s_i \in S_i.$$

La soluzione che si trova con questo metodo è la stessa che si troverebbe con i criteri descritti precedentemente. Da notare che potrebbero esserci più strategie per le quali la condizione citata è verificata; in quel caso il gioco ha più equilibri di Nash, e i giocatori devono decidere quale scegliere. Non c'è però alcuna garanzia che, una volta trovato un equilibrio di Nash, esso sia la soluzione più desiderabile per un dato gioco (potrebbe anzi succedere il contrario, e cioè che la soluzione che massimizza l'utilità per tutti non sia un equilibrio di Nash). È possibile misurare l'*efficienza* di un equilibrio e verificare quindi quanto esso sia conveniente, o usare questa informazione per scegliere fra più equilibri. Si introduce a questo scopo il concetto di *Pareto-superiorità* di un profilo di strategie.

Un profilo s è *Pareto-superiore* rispetto a un altro s'se il guadagno del giocatore *i*–esimo aumenta scegliendo s piuttosto che s', ma allo stesso tempo la scelta di s non causa diminuzione nel guadagno di nessuno degli altri giocatori; o meglio:

$$u_i(s_i, s_{-i}) \geq u_i\left(s_i^{'}, s_{-i}^{'}\right),$$

con la condizione verificata in senso stretto per almeno un giocatore. Inoltre, un profilo di strategie s^{po} è *Pareto-ottimo* se non ne esiste uno *Pareto-superiore* a s^{po}. Se esiste una soluzione di questo tipo, un giocatore non può scegliere di fare un'azione che non sia quella che porta a s^{po} senza causare perdite nel guadagno di qualcun altro. In pratica, questo strumento consente, nel caso in cui siano presenti più equilibri di Nash, di eliminare quelli *Pareto-inferiori* rispetto ad altri. Una soluzione può essere *Pareto-ottima* senza essere necessariamente un equilibrio di Nash.

3.1.6 Giochi a orizzonte finito/infinito

Nel caso in cui il gioco non è costituito da un solo passo, (ovvero non è un gioco *statico*), ma si abbiano una serie di passi, in ognuno dei quali ogni giocatore fa la sua azione (giochi *ripetuti*) ed in cui eventualmente la strategia sarà scelta anche in base alle strategie usate dagli altri giocatori nei passi precedenti (giochi *dinamici*), ogni partecipante vorrà massimizzare la propria utilità per la durata totale del gioco, e non solo all'interno di un passo. Se il gioco ha una durata *T*, l'utilità diventa:

$$u_i = \sum_{t=0}^{T} u_i(t, s).$$

Se *T* è un numero finito, allora si parla di gioco a *orizzonte finito*; altrimenti, il gioco viene definito a *orizzonte infinito*. Se si pensa al caso in esame, e quindi a un gioco che vuole modellare delle connessioni wireless di utenti qualsiasi, bisogna considerare un orizzonte infinito, dato che è ragionevole pensare che nessun utente (inteso come il dispositivo di connessione senza fili) sappia quando la connessione verrà chiusa. Spesso si

rappresenta questa situazione con un'utilità nella quale a ogni passo si diminuisce il guadagno degli stadi successivi del gioco; essa può essere definita come:

$$u_i = \sum_{t=0}^{\infty} u_i(t,s)\delta^t, 0 < \delta < 1$$

dove δ è il *fattore di sconto*, ed è un numero solitamente vicino all'unità, dato che è presumibile pensare che in due passi successivi il payoff non vari sensibilmente.

Si vedrà in seguito che i giochi a orizzonte infinito portano i partecipanti a tenere un comportamento collaborativo, perché ciò assicura un certo guadagno a ogni passo, e mette al riparo dal rischio di punizioni da parte degli altri giocatori, che causano perdite maggiori rispetto al guadagno ottenuto deviando dal comportamento precedente. Esiste un teorema che dà giustificazione formale di questo fatto, e può essere enunciato a partire dalla definizione del *valore minmax*, ossia il valore minimo del guadagno a un certo passo che gli avversari del giocatore *i*-esimo possono fargli ottenere tramite punizioni, supponendo che *i* usi come strategie la *best response*:

$$\underline{u_i} = \min_{s_{-i}}[\max_{s_i} u_i(s_i, s_{-i})]$$

Si può dimostrare che questo valore rappresenta, in un gioco ripetuto dall'orizzonte infinito, il minimo payoff che il giocatore *i* ottiene, qualsiasi sia l'equilibrio di Nash a cui si arriva e indipendentemente dal fattore di sconto.

Teorema:

Per ogni vettore di guadagni possibile $\boldsymbol{u} = \{u_i\}_i$, *con* $u_i > \underline{u_i}$, *esiste un fattore di sconto* $\underline{\delta} < 1$ *tale che per tutti i* $\delta \in (\underline{\delta}, 1)$ *esiste un equilibrio di Nash con guadagni pari a* $\boldsymbol{u}$.

In pratica, se il gioco è abbastanza lungo, avviene esattamente quello che intuitivamente era già evidente, e cioè che il guadagno ottenuto deviando anche solo una volta dal comportamento collaborativo viene perduto nei passi successivi del gioco (nei quali si può ambire solo al valore *minmax*), a causa dei provvedimenti punitivi messi in atto dagli altri giocatori.

3.2 Comportamento egoistico in reti ad hoc

Nelle reti wireless, la teoria dei giochi è stata utilizzata per sviluppare protocolli che siano resistenti alle tecniche misbehaviour. In questo contesto si ipotizza che tutti gli utenti siano egoisti e razionali, cioè che scelgano strategie che massimizzano la loro utilità. Inoltre supponiamo che essi comunichino tra di loro in modalità ad hoc (quindi senza alcun access point).

Nell'ottica della teoria dei giochi, ogni nodo è un giocatore, il *throughput* di cui gode è il suo *payoff*, e la grandezza della finestra di contesa rappresenta la strategia del giocatore. I protocolli sono stati progettati in modo da raggiungere uno stato di equilibrio, "equilibrio di Nash", dove un cheater da solo non può trarre alcun vantaggio dalle stazioni che si comportano bene.

Nei prossimi paragrafi vengono studiate varie situazioni. Per prima, viene presa in considerazione una rete con un solo cheater; dopodiché viene analizzato il caso in cui nella rete si hanno numerosi cheater e si andranno ad identificare due famiglie di equilibri di Nash in un gioco statico: una famiglia provoca sempre il collasso della rete mentre, nell'altra, si nota l'esistenza di un solo utente egoista, che ottiene un throughput non nullo. Poiché gli equilibri di Nash del gioco statico (a singolo stadio) sono contemporaneamente molto inefficienti e molto scorretti, viene cercata una soluzione alternativa. Con questo proposito si calcola il punto di giusto funzionamento, cioè il Pareto-ottimo del sistema. Viene poi mostrato come trasformare il punto Pareto-ottimale in un punto di equilibrio di Nash utilizzando la teoria dei giochi ripetuti. Si introduce la nozione di giocatori cooperativi (*cooperative players*), ovvero cheater che tentano di continuare ad operare nel punto di Pareto-ottimo. Poi viene proposta una tecnica di rivelazione e punizione per quei giocatori che esibiscono un comportamento non cooperativo. Infine, viene spiegato come i giocatori possono collettivamente ricercare il punto Pareto-ottimale di funzionamento, anche se non a conoscenza del numero di nodi presenti nella rete.

3.2.1 Gioco CSMA/CA

Sia *N* un numero finito di nodi wireless, disposti a trasmettere i dati a *N* ricevitori. In [22] si realizza un gioco, in cui tutti i nodi utilizzano lo stesso canale radio e condividono lo stesso dominio di collisione, ovvero ogni nodo può sentire qualsiasi altro nodo; questo è per evitare le complicazioni introdotte dal problema del terminale nascosto. Come specificato nel Capitolo1, le stazioni (nodi) utilizzano il protocollo basato su CSMA/CA per risolvere la contesa a livello MAC. Si assume che ogni nodo presenti un meccanismo di autenticazione a livello MAC (l'indirizzo MAC). Infine, si ipotizza che i nodi siano statici e che abbiano sempre pacchetti (delle stesse dimensioni) da inviare. Oltre a *N*, viene considerato un sottoinsieme *P* di nodi mittente, di cardinalità $|P|$, che non rispettano il protocollo IEEE 802.11. Dunque i nodi del sottoinsieme *P* sono cheater. Senza alcuna perdita di generalità, $P = \{1, \dots, |P|\}$.

Ci possono essere tanti modi in cui un nodo può imbrogliare, nel Capitolo 2 sono state presentate alcune tecniche misbehavior. Ad esempio, un cheater $i \in P$ può inizializzare la dimensione della sua finestra di contesa ad un valore inferiore W_i al fine di ottenere un *throughput* maggiore (per evitare di fare confusione, si continuerà a denotare con *CW* la finestra di contesa di un nodo che non è cheater). Inoltre, un cheater non rispetta l'algoritmo di backoff esponenziale [1] e mantiene la dimensione della sua finestra di contesa fissa dopo una collisione, cioè uguale a W_i. Questo modo di imbrogliare è il più facile per i potenziali cheater, perché non richiede di fare modifiche nel funzionamento dello standard IEEE 802.11. Come specificato nella sezione 3.1, la teoria dei giochi si basa sull'assunzione di razionalità dei giocatori, di conseguenza tutti i cheater $i \in P$ tenteranno di massimizzare il loro throughput medio τ_i.

L'insieme delle strategie pure S_i di un dato giocatore *i*, viene definito in questo modo:

$$S_i = \{1, 2, \dots, W_{max}, W_\infty\},$$

dove $W_{max} < \infty$ è il più alto valore finito che può essere assegnato alla finestra di contesa di un giocatore, e il simbolo W_∞ significa che il giocatore *i* non trasmette, (questo è equivalente a $W_i = \infty$); si osservi che l'insieme S_i è finito.

La strategia di ogni giocatore *i* consiste nel fissare la finestra di contesa $W_i \in S_i$ ad un valore specifico. Dato che ogni giocatore *i* tenta di massimizzare il proprio throughput, e dunque la funzione di payoff di un giocatore *i*, u_i sarà uguale al throughput $\tau_i^{(c)}$, o meglio:

$$u_i(W) = \tau_i^{(c)}(W),$$

con $W = (W_1, \dots, W_{|P|}, W_{|P|+1}, \dots, W_N)$.

$W_j \in S_j$, $(j \leq |P|)$ è la strategia scelta dal giocatore $j \in P$, le $W_k, (|P| + 1 \leq k \leq N)$ sono le finestre di contesa dei nodi che si comportano bene (*well-behaved*), e la notazione "(c)" indica un cheater. Nell'analisi fatta, usando la teoria dei giochi, spesso non vengono presi in considerazione i nodi che si comportano bene, quindi spesso si fa riferiremo a *W* come $W = (W_1, \dots, W_{|P|})$. Infine, il gioco CSMA/CA è definito da:

$$G_{CSMA/CA} = \langle N, P, (S_i)_{i \in P}, (\tau_i^{(c)})_{i \in P} \rangle.$$

3.2.2 $G_{CSMA/CA}$ statico - descrizione del payoff $u_i(W)$

Come già spiegato in 3.1.6, un gioco statico è a singolo stadio, ovvero tutti giocatori prendono decisioni (selezionano una strategia) simultaneamente, senza conoscere le strategie scelte dagli altri giocatori.

Prima di definire la funzione di payoff ($u_i(W) = \tau_i^{(c)}(W),\ i \in P$) è necessario capire la relazione tra la finestra di contesa $W = (W_1, \dots, W_{|P|}, \dots, W_N)$ e il payoff risultante $\tau_i^{(c)}(W)$.

Se l'obbiettivo del cheater è quello di massimizzare il suo throughput (e viene considerato il caso in cui ha sempre pacchetti da inviare), sicuramente questo tenterà di utilizzare tutta la capacità del canale (ciò significa che il sistema funzionerà al punto di saturazione).

Ci sono vari modelli di analisi del livello MAC (dello standard 802.11) in condizione di saturazione. Uno dei più conosciuti è il così detto modello di Bianchi [20], pubblicato nel 2000.

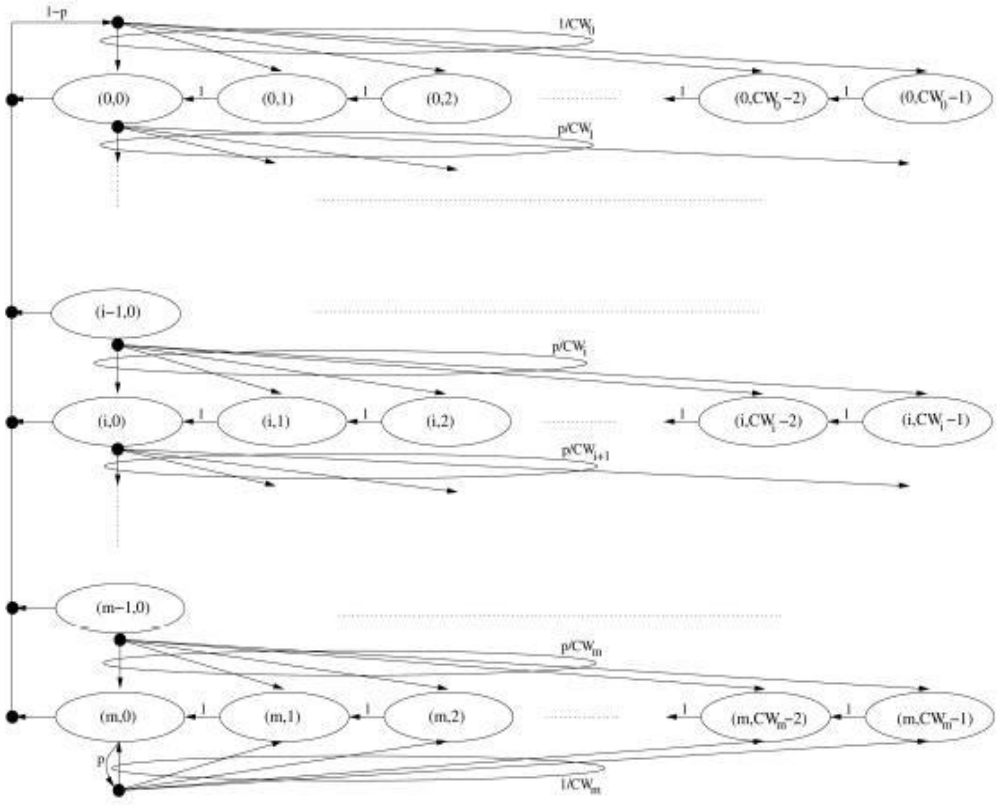


Figura 3.1 Modello di Bianchi [20]

3.2.2.1 Modello di Bianchi

Il principale contributo del modello di Bianchi è dato dalla possibilità di calcolare in forma analitica il throughput di saturazione in un'espressione a forma chiusa. Il modello calcola anche la probabilità di perdita di un pacchetto a causa di una collisione avvenuta durante la trasmissione.

Si assume che il canale sia in condizioni ideali, cioè che non si verifichino né il problema del terminale nascosto, né il problema del nodo esposto. Bianchi utilizza una catena bidimensionale di Markov a *m* stati di backoff, in cui ogni stato rappresenta il timer di backoff di ogni nodo (figura 3.1).

Una catena di Markov è un sistema a stati. Ogni stato è definito da {*s(t)*, *b(t)*}, dove *s(t)* rappresenta la fase di backoff, mentre *b(t)* rappresenta la corrispondente grandezza della finestra di contesa *CW*, ed infine *t* e *t*+1 corrispondono all'inizio di due slot consecutivi. Ogni stazione decrementa il timer di backoff all'inizio di ogni slot. Il timer di backoff viene congelato quando il canale è occupato, perciò l'intervallo di tempo tra *t* e *t*+1 può essere maggiore da quello definito dallo slot nello standard 802.11, visto che può corrispondere sia ad una trasmissione sia ad una collisione di pacchetti.

3.2.2.1.1 Calcolo della probabilità di trasmissione

La catena bidimensionale di Markov a tempo discreto (gli stati di partenza e di arrivo sono associati ad una probabilità) del nodo *i* può essere descritta dalle seguenti equazioni:

$$\begin{cases} P\{i,k|i,k+1\}=1 & k\in(0,2^{i}W_{min}-2) & i\in(0,m) \\ P\{0,k|i,0\}=\frac{1-p}{CW_0} & k\in(0,CW_0-1) & i\in(0,m) \\ P\{i,k|i-1,0\}=\frac{p}{CW_i} & k\in(0,CW_i-1) & i\in(1,m) \\ P\{m,k|m,0\}=\frac{p}{CW_m} & k\in(0,CW_m-1) & \end{cases} \quad (1)$$

tenendo conto che per ogni stato, CW_i è il valore massimo che può avere la finestra di contesa ed è uguale a $2^{i}W_{min}$, con $W_{min}=CW_{min}+1$ (di conseguenza: $CW_{max}=2^{m}W_{min}-1$).

In altre parole, se in uno dei stati (i, 0) avviene una trasmissione corretta, il valore casuale di backoff sarà scelto nell'intervallo (0, CW_0-1) con probabilità $\frac{1-p}{CW_0}$. Questo caso si ritrova negli stati da (0, 0) fino a (0, CW_0-1) nella catena di Markov (figura 3.1). Se

invece, avviene una collisione, ad esempio negli stati $(i-1, 0)$, il valore casuale di backoff sarà scelto nell'intervallo $(0, CW_i - 1)$ con probabilità $\frac{p}{CW_i}$. Nella catena di Markov, questa situazione è rappresentata dagli stati $(i, 0)$ fino a $(i, CW_i - 1)$. Il modello in questione assume che ad ogni tentativo di trasmissione, indifferentemente dal numero di ritrasmissioni avvenute, ogni pacchetto ha una probabilità di collisione uguale a *p*, con *p* costante ed indipendente.

Considerando π la probabilità che una stazione trasmette un pacchetto con successo, *p* si può scrivere come:

$$p = 1 - (1 - \pi)^{N-1} \tag{2}$$

Sia $b_{i,k} = \lim_{t\to\infty} P\{s(t) = i, b(t) = k\}, i \in (0, m), k \in (0, CW_i - 1)$ la distribuzione stazionaria della catena. Una trasmissione avviene quando il contatore del backoff è uguale a zero. Da ciò la probabilità che una stazione trasmette in uno slot scelto casualmente, è data da:

$$\pi = \sum_{i=0}^{m} b_{i,0} \tag{3}$$

Dalla catena di Markov, è facile esprimere $b_{i,0}$ in funzione di *p*:

$$\begin{cases} b_{i,0} = p^i b_{0,0} & 0 < i < m \\ b_{m,0} = \frac{p^m}{1-p} b_{0,0} & \\ b_{i,k} = \frac{CW_i - k}{CW_i} b_{i,0} & 0 < i < m, \qquad 0 < k < CW_i - 1. \end{cases} \tag{4}$$

La prima e la seconda espressione in (4) risultano dal fatto che $b_{i-1,0} \cdot p = b_{i,0}$ per $0 < i < m$ e $b_{m-1,0} \cdot p = (1-p) b_{m,0}$.

La terza equazione si ottiene da:

$$\sum_{i=0}^{m} b_{i,0} = \frac{b_{0,0}}{(1-p)}$$

e prendendo in considerazione la regolarità della catena (per $k \in (1, CW_i - 1)$), vale a dire:

$$b_{i,k} = \frac{CW_i - k}{CW_i} \cdot \begin{cases} (1-p)\sum_{j=0}^{m} b_{j,0} & i = 0 \\ p \cdot b_{i-1,0} & 0 < i < m \\ p \cdot \left(b_{m-1,0} + b_{m,0}\right) & i = m \end{cases} \tag{5}$$

Imponendo la condizione di normalizzazione, insieme all'equazione (4) è possibile ottenere $b_{0,0}$ in funzione di p:

$$\begin{aligned} 1 &= \sum_{i=0}^{m} \sum_{k=0}^{CW_i - 1} b_{i,k} \\ &= \sum_{i=0}^{m} b_{i,0} \sum_{k=0}^{CW_i - 1} \frac{CW_i - k}{CW_i} = \sum_{i=0}^{m} b_{i,0} \frac{CW_i + 1}{2} = \sum_{i=0}^{m} b_{i,0} \frac{2^i W_{min} + 1}{2} = b_{0,0} \frac{W_{min} + 1}{2} \\ &+ \sum_{i=1}^{m-1} \left(b_{0,0} p^i \left(\frac{2^i W_{min} + 1}{2} \right) \right) + \left(\frac{b_{0,0} p^m}{1-p} \right) \left(\frac{2^m W_{min} + 1}{2} \right) \\ &= \frac{b_{0,0}}{2} \left[W_{min} + 1 + \sum_{i=1}^{m-1} \left((2p)^i + p^i \right) + \frac{p^m}{1-p} (2^m W_{min} + 1) \right] \\ &= \frac{b_{0,0}}{2} \left[W_{min} \left(\sum_{i=0}^{m-1} (2p)^i + \frac{(2p)^m}{1-p} \right) + \frac{1}{1-p} \right]. \end{aligned} \tag{6}$$

Quindi $b_{0,0}$ può essere scritto come:

$$b_{0,0} = \frac{2(1-2p)(1-p)}{(1-2p)(W_{min}+1) + pW_{min}(1-(2p)^m)}. \tag{7}$$

Finalmente, considerando le equazioni (3), (4), e (7) la probabilità di accesso al canale π di un nodo in funzione del numero di stati di backoff *m*, del valore minimo della finestra di contesa W_{min}, e della probabilità di collisione *p*, risulta:

$$\begin{aligned} \pi &= \sum_{i=0}^{m} b_{i,0} = \frac{b_{0,0}}{1-p} = \frac{2(1-2p)}{(1-2p)(W_{min}+1) + pW_{min}(1-(2p)^m)} \\ &= \frac{2}{1 + W_{min} + pW_{min} \sum_{k=0}^{m-1} (2p)^k}. \end{aligned} \tag{8}$$

Le equazioni (2) e (8) formano un sistema di due equazioni non lineari, con soluzione unica che può essere risolto numericamente per valori di p e π.

3.2.2.1.2 Calcolo del throughput

Una volta ottenute le probabilità, il throughput di un dato nodo *i* si ottiene da:

$$\tau_i = \frac{E[Payload\ information\ transmitted\ by\ user\ i\ in\ a\ slot\ time]}{E[Duration\ of\ slot\ time]}$$

$$= \frac{P_s^i L}{P^s T^s + P^c T^c + P^{id} T^{id}}. \qquad (9)$$

Il *payload* rappresenta il carico utile, che in questo caso (dei frame) corrisponde alla parte dei dati priva di intestazione e di checksum (quindi al pacchetto gestito al livello di rete).

$P_i^s = \pi_i \prod_{j \neq i}(1 - \pi_j)$ è la probabilità che una stazione *i* trasmette con successo durante un slot casuale;

L è la grandezza media del payload;

$P^s = \sum_{j=1}^{N} P_j^s$;

T^s è il tempo medio necessario per trasmettere un pacchetto di taglia L;

$P^{id} = \prod_{j=1}^{N}(1 - \pi_j)$ è la probabilità che il canale sia inattivo;

T^{id} corrisponde alla durata del periodo d'inattività;

$P^c = 1 - P^{id} - \sum_{j=1}^{N} P_j^s$ è la probabilità di collisione;

T^c il tempo medio consumato dalla collisione.

In fin dei conti, si deve verificare che $P^s + P^c + P^{id} = 1$.

T^c e T^s possono essere calcolati per la trasmissione base (cioè senza pacchetti RTS/CTS) in questo modo:

$$\begin{cases} T^s = H + L + SIFS + \sigma + ACK + DIFS + \sigma \\ T^c = H + L + DIFS + \sigma \end{cases} \quad (10)$$

Dove:

H = tempo necessario per inviare il header del pacchetto

L = tempo necessario per spedire il payload

ACK = acknowledgement

σ = ritardo di propagazione

3.2.2.2 Modello di Bianchi esteso

Per descrivere una rete con nodi cheater, [20] estende il modello di Bianchi utilizzando due catene di Markov distinte. La prima, con $m = 0$ (senza backoff esponenziale, in quanto si suppone che i cheater fissino la loro finestra di contesa), viene utilizzata per calcolare la probabilità di accesso al canale $\pi_i^{(c)}$ dei cheater $i \in P$. La seconda, con $m > 0$, è impiegata per calcolare la probabilità di accesso al canale $\pi_j^{(w)}$ dei nodi che si comportano bene (well-behaved).

Le probabilità condizionate di collisione vengono calcolate considerando le probabilità di accesso al canale di tutte e due le categorie di nodi (cheater e well-behaved).

Come specificato in 3.2.1, il cheater *i* non rispetta l'algoritmo di backoff dell'IEEE 802.11 (caso $m = 0$); di conseguenza, la sua probabilità di accesso al canale secondo [21] è:

$$\pi_i^{(c)} = \frac{2}{W_i + 1}, \quad (11)$$

dove W_i è la grandezza della finestra di contesa del cheater *i*.

Invece la probabilità di accesso al canale $\pi_j^{(w)}$ per i nodi che si comportano bene è:

$$\pi_j^{(w)} = \frac{2}{1 + W_{min} + p^{(w)} W_{min} \sum_{k=0}^{m-1} (2p^{(w)})^k}, \quad (12)$$

con

$$p^{(w)} = 1 - (1 - \pi_j^{(w)})^{N-|P|-1} \prod_{i \in P} (1 - \pi_i^{(c)}). \quad (13)$$

dove (13) è la generalizzazione dell'equazione (2) in presenza di nodi cheater. Da notare che $\pi_j^{(w)}$ è la stessa per tutti nodi well-behaved per cui è possibile scrivere $\pi_j^{(w)} = \pi^{(w)}$.

Utilizzando (9), si ottiene la seguente espressione per il throughput, $\tau_i^{(c)}$, di un cheater *i*:

$$\tau_i^{(c)} = \frac{\pi_i^{(c)} c_i^{(1)}}{\pi_i^{(c)} c_i^{(2)} + c_i^{(3)}}, \tag{14}$$

dove

$$c_i^{(1)} = p_{-i} L \tag{15}$$

$$c_i^{(2)} = p_{-i}\left(T^s - T^{id}\right) - s_{-i}(T^s - T^c) \tag{16}$$

$$c_i^{(3)} = (1 - p_{-i} - s_{-i})T^c + s_{-i}T^s + p_{-i}T^{id}. \tag{17}$$

Sono state usate le seguenti sostituzioni:

$$p_{-i} = \prod_{j \in P \setminus \{i\}} \left(1 - \pi_j^{(c)}\right)\left(1 - \pi^{(w)}\right)^{N-|P|}$$

$$s_{-i} = \sum_{j \in P \setminus \{i\}} \pi_j^{(c)} \prod_{k \in P \setminus \{i,j\}} \left(1 - \pi_k^{(c)}\right)(1 - \pi^{(w)})^{N-|P|}. \tag{18}$$

Da notare che l'unico parametro su cui il cheater ha controllo è il proprio W_i. Al variare di W_i, un nodo cambia la sua probabilità di accesso $\pi_i^{(c)} = f(W_i)$, ma anche la probabilità $\pi^{(w)} = f(W)$ $(W = (W_1, \dots, W_i, \dots, W_N),\ W_i = W_{min},\ i = \{I+1, \dots, N\})$ di tutti nodi well-behaved; ciò segue da (11), (12) e (13). Dunque diminuendo la sua finestra di contesa W_i, un nodo egoista può aumentare il suo throughput.

Ciò è stato provato anche sperimentalmente [22] utilizzando il simulatore ns-2. I parametri di simulazione sono riassunti nella tabella 3.1. Vengono considerati 20 nodi ($N = 20$) mittente, uno dei quali (nodo *X*) è cheater.

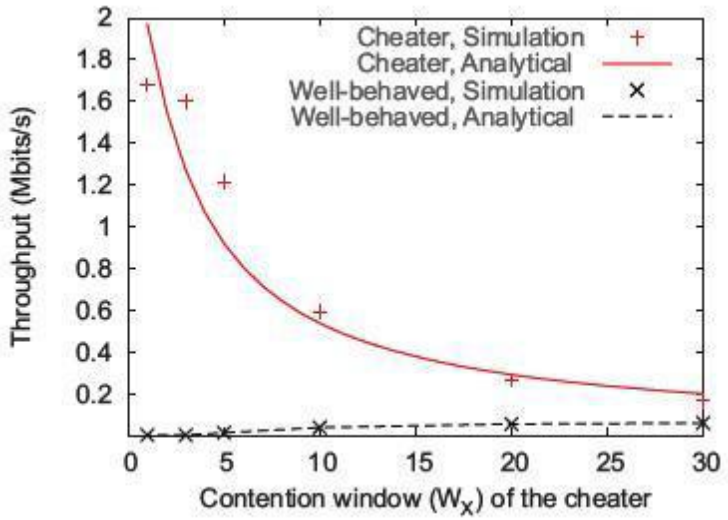


Figura 3.2 Modello di Bianchi esteso – analisi del throughput [22]

I valori dei parametri del protocollo IEEE 802.11 sono stati scelti in accordo allo standard IEEE 802.11b [1]. La durata di ogni simulazione è di 50 secondi e la media dei risultati viene fatta ogni 5 simulazioni.

La figura 3.2 presenta il throughput al variare di W_X. È possibile notare la compatibilità tra i risultati analitici e quelli sperimentali. In effetti il grafico mostra come il througput ottenuto dal cheater aumenta con il diminuire di W_X.

Parameter	Value
Topology	100 m × 100 m, random
Receive range	240 m
Propagation	Free space
MAC	802.11b
Scheme	Basic (No RTS/CTS)
Channel capacity	2 Mbits/s
Traffic sources	CBR/UDP, 1050-byte frames every 5 ms (1.68 Mb/s)

Tabella 3.1 Parametri della simulazione

3.2.3 Soluzione del gioco

Nella trattazione che segue, non vengono considerati i nodi con un buon comportamento (ovvero, si ipotizza $N = |P|$). L'attenzione, piuttosto, si focalizza sull'equilibrio di Nash in strategie pure, che si verifica nel gioco $G_{CSMA/CA}$.

3.2.3.1 Esistenza dell'equilibrio di Nash nel gioco statico

Dalla teoria dei giochi si conosce che in strategia pura ogni giocatore agisce utilizzando la miglior risposta (*best response*) [23]. Qui di seguito, viene studiata l'esistenza dell'equilibrio di Nash, utilizzando quindi il concetto della funzione di miglior risposta del giocatore (vedi 3.1.6).
Vengono introdotte le seguenti notazioni:

$$W_{-i} = (W_1, \dots, W_{i-1}, W_{i+1}, \dots, W_{|P|})$$

$$S_{-i} = \{S_1, \dots, S_{i-1}, S_{i+1}, \dots, S_{|P|}\}$$

dove S_i sono gli insiemi delle strategie pure dei giocatori. Allora la funzione di miglior risposta del giocatore *i* è data da:

$$br_i(W_{-i}) = \{W_i \in S_i : \tau_i^{(c)}(W_i, W_{-i}) \geq \tau_i^{(c)}\left(W_i^{'}, W_{-i}\right) \; per \; ogni \; W_i^{'} \in S_i\}.$$

Dalla teoria dei giochi, un profilo di strategia pura $W^* = (W_1^*, \dots, W_{|P|}^*)$ è un equilibrio di Nash se e soltanto se $W_i^* \in br_i(W_{-i}^*)$ per ogni giocatore $i \in P$.

Lemma 1. *Per ogni profilo di strategia W che costituisce un equilibrio di Nash in* $G_{CSMA/CA}$, $\exists \; i \in P$ *tale che* $W_i = 1$.

Dimostrazione: Supponiamo per assurdo che $W = (W_1, \dots, W_{|P|})$ sia un equilibrio di Nash tale che $W_k > 1, \forall \; k \in P$. Considerato un giocatore *i* e la sua funzione di miglior risposta $br_i(W_{-i})$, e ricordando la dipendenza tra $\tau_i^{(c)}$ e W_i ($W_k > 1 \Rightarrow \pi_k^{(c)} < 1, \forall k \in P$) si ottiene che la condizione:

$$\tau_i^{(c)}(W_i, W_{-i}) \geq \tau_i^{(c)}(W_i', W_{-i}), per\ ogni\ W_i' \in S_i,$$

è soddisfatta solo per $br_i(W_{-i}) = \{1\}$. Per definizione, ad ogni equilibrio di Nash $W_i \in br_i(W_{-i})$, di conseguenza $W_i = 1$. Ciò contraddice l'ipotesi, $W_i > 1$, pertanto la dimostrazione è conclusa.

Teorema 1 *Il gioco $G_{CSMA/CA}$ statico ammette esattamente $(W_{max} + 1)^{|P|} - W_{max}^{|P|}$ equilibri di Nash.*

Dimostrazione: Per ipotesi si suppone che un qualche giocatore $i \in P$ abbia $W_i = 1$. Pertanto la sua probabilità di accesso al canale è $\pi_i^{(c)} = 1$ e di conseguenza per tutti gli altri giocatori $k \in P\backslash\{i\}$, $\tau_k^{(c)} = 0$, qualunque sia $W_k \in S_k$ (equazione (14)). Quindi per un qualsiasi $W_k \in S_k$, è vero che $W_k \in br_k(W_{-k})$, con $k \in P\backslash\{i\}$. Ciò si ottiene indipendentemente dal numero dei giocatori che hanno impostato la loro finestra di contesa all'unità. Associando questo fatto con il Lemma 1, si ottiene la seguente caratterizzazione dell'equilibrio di Nash:

(**Equilibrio di Nash**) Ad ogni equilibrio di Nash di $G_{CSMA/CA}$ esiste almeno un cheater che imposta la sua finestra di contesa ad uno, mentre tutti gli altri giocano una qualunque strategia tra $\{1, \dots, W_{max}, W_\infty\}$.

La dimostrazione si conclude osservando che su un totale di $(W_{max} + 1)^{|P|}$ profili di strategie $W = (W_1, \dots, W_{|P|})$, $(W_j \in \{1, \dots, W_{max}, W_\infty\})$, esattamente $W_{max}^{|P|}$ non contengono alcun elemento unitario.

Si distinguono due famiglie di equilibri di Nash. Per descriverle, si consideri l'insieme $\mathcal{D} = \{i : W_i = 1, i \in P\}$.

Nella prima famiglia: $|\mathcal{D}| = 1$, c'e soltanto un giocatore $i \in P$ che gioca $W_i = 1$ e riceve throughput $\tau_i^{(c)} > 0$, mentre $\tau_k = 0$ per tutti gli altri giocatori $k \in P \setminus \{i\}$.

Nella seconda famiglia: $|\mathcal{D}| > 1$, esiste più di un giocatore $i \in P$ che usa come strategia $W_i = 1$, per cui si ha $\tau_k^{(c)} = 0$ per tutti giocatori $k \in P$.

L'equilibrio di Nash della prima famiglia è anche Pareto-ottimale, perché, per esempio, in un profilo di strategie $W = (1, W_2 = W_\infty, \dots, W_{|P|} = W_\infty)$, tutti giocatori $P\backslash\{1\}$ effet-

tivamente non trasmettono ($W_i = W_\infty = \infty \Rightarrow \pi_i^{(c)} = 0$). Di conseguenza, il giocatore 1 ottiene l'intera capacità del canale per se stesso. Questo, dunque, è un equilibrio scorretto, in quanto è avvantaggiato un solo giocatore. L'equilibrio raggiunto nella seconda famiglia, invece, è conosciuto come *the tragedy of the commons*, perché, come già visto, tutti nodi ottengono come throughput zero e pertanto le risorse della rete sono completamente sprecate.

3.2.3.1 La stabilità (robustezza) dell'equilibrio di Nash in $G_{CSMA/CA}$

Una volta identificato l'equilibrio è possibile studiare le sue proprietà. Interessa particolarmente conoscere se l'equilibrio si mantiene nel caso in cui vengono apportate modifiche alla definizione del gioco, quindi interessa verificare dove il gioco è essenziale e dove non lo è. A questo scopo viene studiata la robustezza dell'equilibrio $W = (W_i = 1)_{i \in P}$.

Sia $\hat{G}_{CSMA/CA}$ un gioco approssimativo all'originale $G_{CSMA/CA}$, definito come segue:

$$\hat{G}_{CSMA/CA} = \langle P, (S_i)_{i \in P}, (\hat{u}_i)_{i \in P} \rangle,$$

con

$$\hat{u}_i(W) = \tau_i^{(c)}(W) - \begin{cases} \epsilon_i, se\ W_i < W_\infty \\ 0, se\ W_i = W_\infty \end{cases}, \forall\ i \in P,$$

dove ϵ_i è una costante infinitesima, positiva ($0 < \epsilon_i \ll 1$) che soddisfa $\tau_i^{(c)}(W) > \epsilon_i$, $\forall\ W$ tale che $\tau_i^{(c)}(W) > 0$. L'esistenza di una tale costante deriva dal fatto che i numeri dei nodi nel sistema è finito ($N < \infty$).

Intuitivamente, il *costo*, espresso con il termine ϵ_i, specifica il fatto che un giocatore preferisce di non trasmettere assolutamente invece di trasmettere senza successo. Essendo un valore infinitesimo, ϵ_i non modifica in modo significativo la funzione di payoff u_i del giocatore *i*.

Dall'analisi dell'equilibrio del gioco $\hat{G}_{CSMA/CA}$ si ottengono i teoremi presentati di seguito (e la cui dimostrazione si ritrova in [21]).

Teorema 2 *Un profilo di strategia W è un equilibrio di Nash del gioco* $\hat{G}_{CSMA/CA}$ *se e soltanto se* $\exists! \, i \in P \; tale \; che \; W_i = 1 \; e \; W_j = W_\infty, \forall j \in P \backslash \{i\}$.

Quindi, con un cambiamento infinitesimo nella funzione di payoff del gioco originale, è possibile costruire un gioco con un insieme diverso di equilibri di Nash. In pratica, l'insieme di equilibri di Nash di $\hat{G}_{CSMA/CA}$ è un piccolo sottoinsieme di quello di $G_{CSMA/CA}$. Poi tutti gli equilibri di Nash in $\hat{G}_{CSMA/CA}$ sono Pareto-ottimali; e ancora di più, il profilo di strategia $(W_i = 1)_{i \in P}$ non è un punto di equilibrio in $\hat{G}_{CSMA/CA}$. Infine, è possibile concludere lo studio della robustezza dell'equilibrio di Nash del gioco originale $G_{CSMA/CA}$, con il seguente teorema:

Teorema 3 *L'equilibrio di Nash* $W = (W_i = 1)_{i \in P}$ *del gioco* $G_{CSMA/CA}$ *è non essenziale (non è robusto), di conseguenza anche il gioco* $G_{CSMA/CA}$ *non è essenziale.*

Il risultato di questo teorema può essere generalizzato: se almeno due giocatori cheater hanno impostato la loro finestra di contesa a 1, nessuno dei due trae benefici dalla rete, perché tutti pacchetti vanno in collisione. Ma se viene inserito anche un minimo costo per la trasmissione, la miglior strategia del cheater (per risparmiare) è semplicemente quella di rifiutarsi di trasmettere.

3.2.4 Soluzione alternativa del gioco

Una soluzione desiderabile per il gioco CSMA/CA dovrebbe contenere le seguenti tre proprietà:

- Unicità – in modo da evitare incertezze da parte dei giocatori, su quale soluzione scegliere.

- Pareto-ottimalità – per quanto riguarda l'allocazione della banda disponibile.
- Correttezza – intesa nel caso in esame come la distribuzione corretta del throughput del sistema (cioè, tutte le stazioni devono avere lo stesso throughput).

Ricordando la situazione di prima con $N = 20$ stazioni collocate nello stesso dominio di collisione, si immagini di avere $|P| = 10$ cheater. Si ipotizza che ogni cheater inizia con una finestra di contesa *W* pari a 1 (seconda famiglia di equilibri di Nash). Osservando che la rete non funziona, questi ultimi, iniziano ad aumentare lentamente e contemporaneamente la loro finestra di contesa, al fine di inviare traffico. Conseguentemente, il loro throughput comincia aumentare velocemente e, assieme ad esso, inizia aumentare lentamente anche il throughput di qualche stazione non cheater. Essendo che il canale è a capacità finita, per un dato valore di *W*, i cheater raggiungono il valore massimo di throughput. Quindi se continuano ad aumentare la loro finestra di contesa, il loro throughput inizia a diminuire, mentre quello delle stazioni well-behaved continua ad aumen-

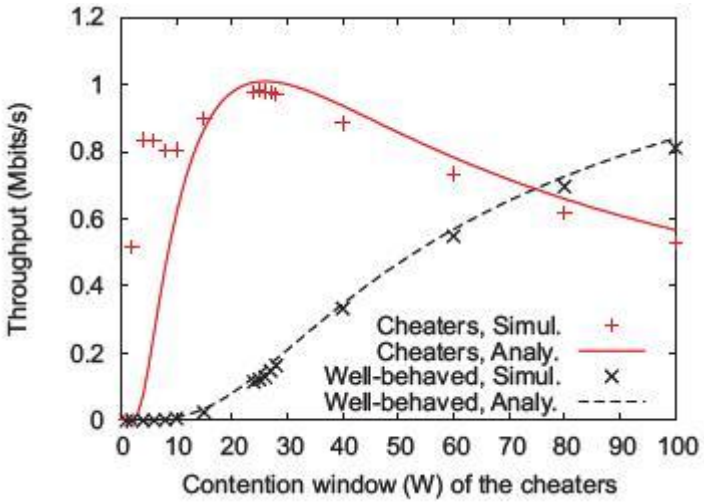


Figura 3.3 Andamento del throughput al variae della finestra di contesa in una rete con numerosi cheater [22]

tare. È possibile notare facilmente questo comportamento nella figura 3.3. Dal grafico si osserva che esiste un unico valore collettivo della finestra di contesa, W^*, che massimizza il throughput del sistema. Quindi, il profilo di strategie $(W_i = W^*)_{i \in P}$ soddisfa tutte le proprietà della soluzione desiderabile del gioco $G_{CSMA/CA}$, è un punto Pareto-

ottimale però $(W_i = W^*)_{i \in P}$ non è un punto di equilibrio di Nash (in quanto, $W_i^* > 1, \forall i \in P$) perciò non è stabile.

3.2.3 $G_{CSMA/CA}$ dinamico

Una volta identificato il punto di Pareto-ottimo, W^*, si cerca di progettare un sistema che permetta ai cheater di convergere a questo punto. A questo scopo si fa utilizzo della teoria dei giochi dinamici e ripetuti [23], [24]. Quindi la scelta delle strategie dei giocatori non avviene più in maniera contemporanea ed è basata sulla conoscenza degli esiti ottenuti ai stati precedenti. Poiché $G_{CSMA/CA}$ viene ripetuto T volte; considerare $T \to \infty$, significa che nessuno dei giocatori sa quando il gioco finisce. Con queste nuove specifiche, la funzione di utilità di ogni giocatore $i \in P$ diventa:

$$u_i^{\infty} = \lim_{T \to \infty} \inf \frac{1}{T} \sum_{t=1}^{T} u_i^t(\pi_i^t, \pi_{-i}^t) \tag{19}$$

dove $u_i^t(\pi_i^t, \pi_{-i}^t)$ rappresenta il payoff del giocatore *i* nella fase *t* e vale:

$$u_i^t(\pi_i^t, \pi_{-i}^t) = \tau^{(c)t}(\pi_i^t, \pi_{-i}^t) - pf_i^t(\pi_i^t, \pi_{-i}^t), \forall i \in P. \tag{20}$$

Invece, $pf_i^t(\pi_i^t, \pi_{-i}^t)$ viene chiamata *funzione di punizione* ed è definita per ogni giocatore $i \in P$.

Questo modello viene utilizzato in [22] per spiegare analiticamente come è possibile trasformare $(W_i = W^*)_{i \in P}$ in un punto di equilibrio di Nash del gioco $G_{CSMA/CA}^{\infty}$.

3.2.3.1 Meccanismo di penalizzazione

Come specificato già nella sezione 1.2, dato che nelle reti ad hoc non è presente l'Access Point, sono i nodi stessi a dover assicurare la correttezza di tutti giocatori; ciò

implica l'esistenza di un monitoraggio e la necessità di reagire (tramite un meccanismo di punizione) ad eventuali deviazioni dalle regole del gioco. Il monitoraggio consiste nel confrontare il proprio throughput con quello delle altre stazioni dopo un certo periodo, T^{obs}. Per quanto riguarda il meccanismo di penalizzazione è importante che esso sia progettato in maniera tale, da non produrre scarse prestazioni a chi lo sta applicando. Ciò può essere assicurato facendo uso del *selective jamming* (*jamming* eseguito su un determinato canale, oppure ad una determinata frequenza). Per esempio, considerati due qualunque giocatori *k* e *i* dell'insieme *P*, e la funzione di punizione definita come:

$$pf_i(\pi_i, \pi_{-i}) = \begin{cases} \tau_i^{(c)}(\pi_i, \pi_{-i}) - \tau_k^{(c)}(\pi_i, \pi_{-i}), se\ \tau_i^{(c)}(\pi_i, \pi_{-i}) > \tau_k^{(c)}(\pi_i, \pi_{-i}) \\ 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad altrimenti \end{cases},$$

quando *i* cerca di imbrogliare allontanandosi dal punto di equilibrio W^*, *k* penalizza *i* eseguendo il jamming selettivo dei pacchetti di quest'ultimo per una breve durata, T^{jam}. Una proprietà fondamentale della funzione di punizione è di assicurare sia al giocatore *k* sia ad *i* lo stesso throughput. Quindi il throughput ricevuto dal giocatore *k* e *i* deve essere lo stesso per tutto il periodo $T^{obs} + T^{jam}$; ovvero $\tau_k T^{obs} + \tau_k T^{jam} = \tau_i T^{obs} + 0T^{jam}$. Di conseguenza:

$$T^{jam} = \left(\frac{\tau_i}{\tau_k} - 1\right) T^{obs}, \qquad (21)$$

con ${\tau_i}/{\tau_k} > 1 + \epsilon, \epsilon$ rappresenta il margine di tolleranza ed è una percentuale di throughput. Per evitare situazioni in cui $T^{jam} = \infty$, in pratica, è possibile definire $T^{jam} = min\{\overline{T^{JAM}}, ({\tau_i}/{\tau_k} - 1)T^{obs}\}$, dove $\overline{T^{JAM}} > 0$ è un valore sufficientemente grande da riportare τ_i vicino a $\tau_k = 0$. In questo modo, viene data l'opportunità alla stazione che sta deviando, *i*, di impostare in modo adeguato la sua finestra di contesa, W_i, e di conseguenza il suo throughput τ_i.

Nella figura 3.4 sono riportati i risultati delle simulazioni eseguite con ns-2 su $N = 20$ stazioni di cui $|P| = 10$. La finestra di contesa del cheater *X* (scelto in modo casuale) è fissata a 10, mentre tutti gli altri hanno la finestra di contesa uguale a 30, coincidente

con il valore Pareto-ottimo (W^*); $T^{obs} = 20$ secondi, mentre $\epsilon = 5\%$. Il cheater X viene identificato dagli altri cheater della rete e penalizzato per la sua deviazione.

Nella figura 3.4(a) è raffigurato il throughput del cheater X sia in presenza che in assenza di punizione. In caso di punizione, il suo throughput raggiunge lo 0. Nella figura 3.4(b) viene illustrato il throughput medio ottenuto da X al variare della finestra di contesa W_X. La media dei risultati è fatta ogni 1000 secondi.

Come si può osservare dal grafico, dopo l'inserzione del meccanismo di identificazione e punizione, il cheater X ottiene throughput massimo solo se adopera il valore Pareto-ottimo. Qualunque deviazione da questo punto, $W_X < W^*$ oppure $W_X > W^*$, lo porta ad ottenere un payoff (throughput) minore. Dunque nessun cheater è incentivato a deviare dal punto Pareto-ottimo e di conseguenza esso diventa un equilibrio di Nash.

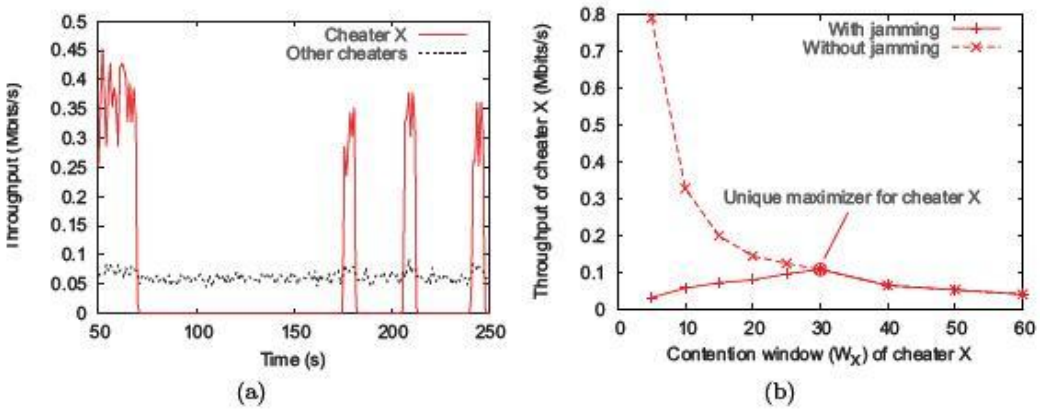


Figura 3.4 Realizzazione del meccanismo di penalizzazione tramite *selective jamming* [22]

3.2.3.2 Meccanismo di rilevazione

Come visto in precedenza, alla base del meccanismo di punizione sta l'abilità dei giocatori nello stimare la differenza dei loro payoff. Ciò è possibile proprio a causa della natura (broadcast) del mezzo di trasmissione wireless, che permette a tutti gli utenti, che si trovano nella zona di copertura di una trasmissione, di udire il messaggio di origine.

Dunque, dopo aver misurato per un periodo T^{obs} il throughput di tutti quanti i giocatori, un nodo *k* decide che un nodo *i* sta deviando, tutte le volte che il throughput del nodo *i* risulta maggiore di quello del nodo *k*, oppure si verifica la relazione:

$$\frac{\tau_i}{\tau_k} > 1 + \epsilon \, .$$

Le prestazioni di tale meccanismo di rilevazione sono state studiate in [22], usando il simulatore ns-2 con le seguenti specifiche: $N = |P| = 30$, la finestra di contesa di un unico nodo *j*, $W_j = variabile$, e le finestre di contesa degli altri nodi $W_k = 30, \forall k \in P \backslash \{j\}$. Nella figura 3.5 sono illustrati i risultati al variare di T^{obs} e ϵ.

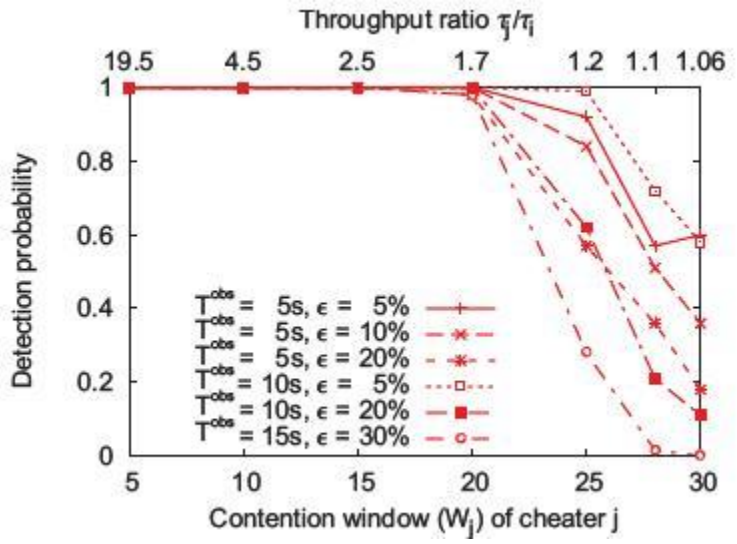


Figura 3.5 Prestazioni del meccanismo di rilevazione [22]

Dal grafico si nota che la scelta dei valori per T^{obs} e ϵ è cruciale nel meccanismo di rilevamento. Poiché per valori di W_j bassi ($W_j \leq 20$), il rapporto τ_j / τ_i è molto maggiore di $1 + \epsilon$, di conseguenza è più semplice individuare la deviazione del nodo *j*.

3.2.3.3 Strategia adattativa

Come specificato in precedenza, nel paragrafo 3.2.3.2, durante il meccanismo di punizione viene data la possibilità al nodo che sta deviando, di adeguare la sua finestra di contesa e di conseguenza il suo throughput. Quindi nel momento in cui un cheater *i* osserva di essere penalizzato (tramite il jamming) durante un qualche periodo Δ, gradualmente esso aumenta la sua finestra di contesa di un numero di passi di valore γ.

Ogni cheater è in grado di capire facilmente se viene punito, osservando il proprio throughput. La scelta di Δ determina l'efficienza del sistema. Ad esempio per valori grandi di Δ, il sistema è inefficiente in quanto non riesce ad individuare i nodi che stanno deviando; mentre con Δ piccoli, la sensibilità del sistema di rilevazione aumenta, obbligando un cheater di ingrandire la sua finestra di contesa senza che questa azione sia necessaria; ciò porta il sistema a convergere a un punto di funzionamento inefficiente.

La scelta di γ genera un trade-off tra il tempo di convergenza ed efficienza del sistema. Se si aumenta la finestra di contesa usando passi grandi, il sistema si stabilizza prima ma il punto di funzionamento può risultare lontano dal punto Pareto-ottimo (W^*), di conseguenza si ottiene un sistema inefficiente; viceversa si arriva allo stesso risultato.

Questa situazione viene simulata in [22] tramite ns-2, usando le seguenti specifiche: $N = 20, |P| = 10, W^{init} = 30$. Si sceglie a caso un nodo *X*, e si fissa $W_X^{init} = 10$. Tutti gli altri nodi del sistema hanno la finestra di contesa uguale a W^{init}. Si prende Δ= 5 secondi e $\gamma = 5$.

Nella figura 3.6(a) è possibile osservare il throughput ottenuto dai cheater nel tempo, mentre nella 3.6(b) è presentata l'evoluzione nel tempo della finestra di contesa del nodo *X*. Si può notare facilmente come, in conformità alla strategia di adattamento, la finestra di contesa di X converge a $W^* = 30$. Dunque si verifica il successo da parte degli altri cheater del sistema, nel portare al punto di equilibrio il cheater che stava deviando.

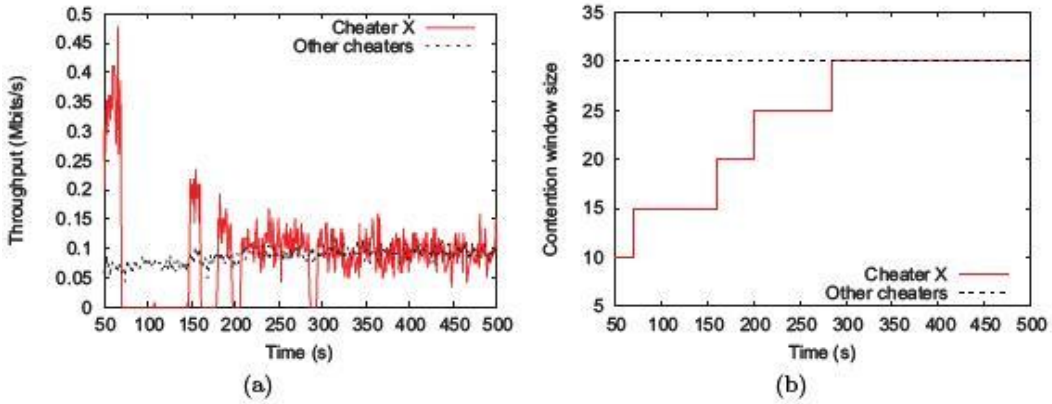


Figura 3.6 Prestazioni del sistema in presenza di strategia adattativa [22]

Utilizzando le stesse specifiche è stato analizzato il caso con tre nodi, *X*, *Y*, *Z*, che deviano nella rete, ovvero: $W_X = 5, W_Y = 10$ e $W_Z = 15$, mentre tutti gli altri nodi possiedono un $W^* = 30$. La figura 3.7 mostra l'evoluzione delle finestre di contesa dei vari cheater nel tempo. Si osserva come, man mano che ognuno di loro viene punito in proporzione al misbehavior, ciascuno inizia a far convergere la propria finestra di contesa a W^*, quindi anche in questo caso si raggiunge il punto di equilibrio del sistema.

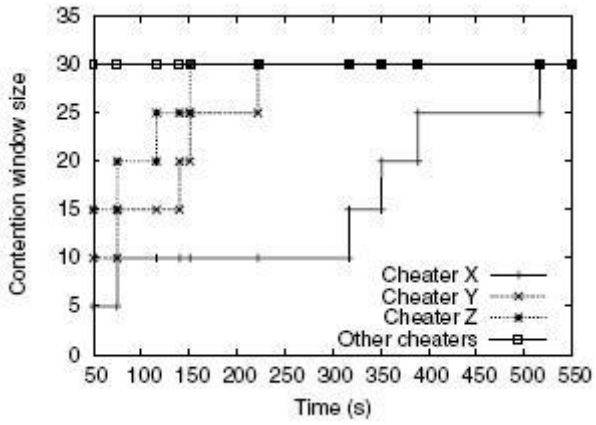


Figura 3.7 Prestazioni del sistema in presenza di più nodi che stanno deviando [22]

Come si può notare, il meccanismo di punizione tramite il jamming selettivo abbinato alla strategia di adattamento produce ottime prestazioni e correttezza nella distribuzione del througput in entrambe le situazioni.

3.2.3.4 Algoritmo per raggiungere il punto di Pareto-ottimo

In questa sezione si spiega com'è possibile indurre i nodi della rete a cooperare in maniera da raggiungere un punto di equilibrio del sistema coincidente con il punto Pareto-ottimo.

Si consideri inizialmente che tutti cheater del sistema abbiano $(W_i = W^{init})_{i \in P}$. Ogni cheater stabilisce un timer casuale (nelle simulazioni è compreso tra 0 e 20 secondi) per aumentare la propria finestra di contesa di un passo γ. Quindi si assume che uno dei cheater, *X*, porti la sua finestra di contesa a $W_X^{init} + \gamma$. In base al meccanismo di rilevazione, *X* decide che tutti gli altri nodi della rete stanno deviando e di conseguenza inizia a penalizzarli. Quando uno dei cheater osserva di essere penalizzato, disabilita il timer e fa uso della strategia di adattamento descritta in 3.2.3.3. Il sistema può raggiunge la stabilità quando tutti cheater hanno $W_i = W_i^{init} + \gamma$. I cheater si accorgono di aver raggiunto un nuovo punto stabile di funzionamento se usufruiscono tutti dello stesso valore di payoff (throughput). A questo punto ogni cheater $i \in P$ confronta il throughput ottenuto usando $W_i = W_i^{init} + \gamma$, con quello precedente quando $W_i = W_i^{init}$. Se esso nota un decremento, la ricerca di W^* termina, altrimenti stabilisce nuovamente un tempo casuale per incrementare la finestra di contesa di γ. Da notare che anche se soltanto un solo cheater osserva un aumento del suo throughput, l'intero equilibrio del sistema si sposta su un altro punto di funzionamento.

Nella figura 3.8 è possibile osservare le prestazioni di tale algoritmo. Nelle simulazioni sono stati usati i seguenti valori: $N = 20, |P| = 7, W_i^{init} = 5$ e la condizione che i cheater continuino la ricerca di W^* solo se osservano un aumento di almeno 10% rispetto al throughput ottenuto all'ultimo punto stabile di funzionamento.

In 3.8(a) viene illustrato un esempio dell'evoluzione delle finestre di contesa di due cheater del sistema. Si osserva come al variare del tempo le finestre di contesa dei cheater convergono a 20. La figura 3.8(b), invece presenta la media dei throughput delle stazioni in base alla variazione delle finestre di contesa.

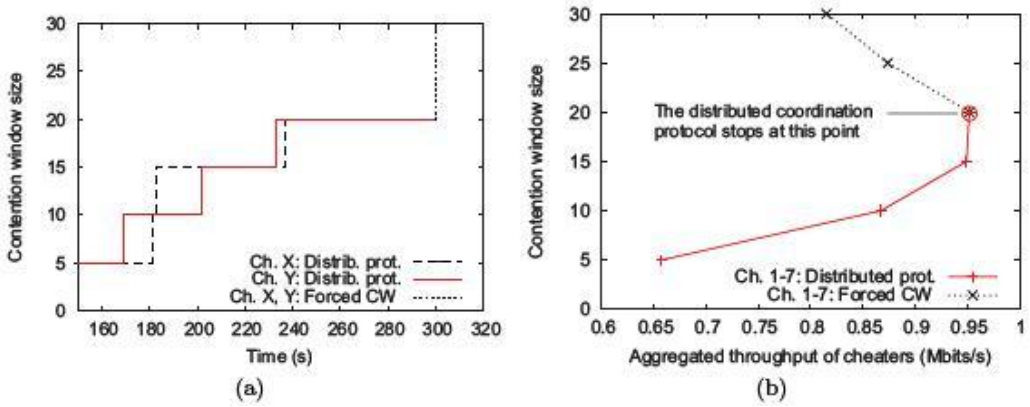


Figura 3.8 Prestazioni dell'algoritmo [22]

Come si può notare il throughput raggiunge il valore massimo in corrispondenza di $W_i = 20$. In pratica, una volta raggiunto questo valore l'algoritmo termina e il sistema continua a funzionare a questo punto di equilibrio che coincide proprio con il punto Pareto-ottimo.

Conclusioni

Il comportamento egoistico al livello MAC in IEEE 802.11 può condurre ad una distribuzione scorretta del throughput: ciò può essere un vero problema negli hotspot pubblici, dove ogni utente paga per l'utilizzo della rete, ed è perciò molto incentivato ad un tale comportamento al fine di aumentare la sua larghezza di banda. Risulta fondamentale, di conseguenza, individuare le stazioni che adottano un comportamento egoistico nella rete, e trovare dei metodi che portino alla cooperazione tra i nodi e ad una distribuzione equa delle risorse della rete.

A questo proposito, si è presentato un sistema di rilevamento, DOMINO, che può essere completamente integrato nell'Access Point. Attraverso i risultati delle simulazioni si sono dimostrate l'efficienza e l'applicabilità di tale sistema in una rete wireless reale con infrastruttura.

Per quanto riguarda le reti ad hoc, utilizzando la teoria dei giochi, è stata proposta una tecnica di rivelazione e punizione per quei "giocatori" che esibiscono un comportamento non cooperativo. In seguito si è proposto anche un algoritmo che stimola la cooperazione tra i nodi della rete.

Tutte le analisi e le simulazioni sono state effettuate con l'assunzione che i nodi della rete utilizzino lo stesso canale radio e che condividano lo stesso dominio di collisione, cioè che ciascun nodo possa sentire qualsiasi altro nodo. Infatti, lo studio del comportamento egoistico nelle reti wireless al livello MAC è tuttora un problema aperto; e ciò è dovuto principalmente alle complicazioni introdotte dal problema del terminale nascosto e alle difficoltà nella scelta dei giusti parametri di monitoraggio.

Bibliografia

[1] LAN/MAN Standards Committee. *ANSI/IEEE Std 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society, 1999.

[2] http://www.learnthat.com/Certification/learn/1370/Free-Network-2005-Training-/page/5/

[3] http://www.docmirror.net/en/linux/howto/networking/OLSR-IPv6-HOWTO/intro.html

[4] Levente Buttyán, Jean-Pierre Hubaux - *Security and cooperation in wireless networks,* 2007.

[5] http://www.cs.wright.edu/~pmateti/InternetSecurity/Lectures/WirelessHacks/Mateti-WirelessHacks_files/image002.jpg

[6] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[7] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proceedings of MOBICOM'02*, pages 107–118, September 2002.

[8] M. Raya, I. Aad, J. P. Hubaux, Alaeddine El Fawal – *DOMINO: Detecting MAC Layer Greedy Behavior in IEEE 802.11 Hotspots, IEEE Trans. Mobile Computing, 2006.*

[9] J. Edney and W. A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley, 2004.

[10] K. Fall and K. Varadhan. *ns notes and documentation*. UC Berkeley, LBL, USC/ISI, Xerox PARC, 2003.

[11] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly. Denial of service resilience in ad hoc networks. In *Proceedings of ACM Mobicom*, Philadelphia, Pennsylvania, USA, 2004.

[12] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proceedings of ACM SIGCOMM 2003*, Karlsruhe, Germany, August 2003.

[13] M.G. Arranz, R. Aguero, L. Munoz, and P. Mahonen. Behavior of UDP-based applications over IEEE 802.11 wireless networks. In *Personal, Indoor and Mobile Radio Communications, 12th IEEE International Symposium on*, volume 2, pages F–72–F–77, Sep/Oct 2001.

[14] G. Xylomenos and G. Polyzos. TCP and UDP performance over a wireless LAN. In *Proceedings of INFOCOM'99*, volume 2, pages 439–46. IEEE, March 1999.

[15] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proceedings of INFOCOM'03*. IEEE, 2003.

[16] C. Barrett, M. Marathe, D. Engelhart, and A. Sivasubramaniam. Analyzing the short-term fairness of IEEE 802.11 in wireless multi-hop radio networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems,10th IEEE International Symposium on*, pages 137–144, 2002.

[17] C. Koksal, H. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless media access protocols. In *Proceedings of ACM SIGMETRICS'00*, June 2000.

[18] R. Ja, The Art of Computer System Performance Analysis. John Wiley and Sons, 1991.

[19] R**.** Kohlas and U. Maurer. Confidence valuation in a public key infrastructure based on uncertain evidence. In *Proceedings of PKC'00*, volume 1751 of *Lecture Notes in computer Science*. Springer-Verlag, 2000.

[20] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal of Selected Areas in Communications*, vol. 18, 2000.

[21] M. Čagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On cheating in CSMA/CA ad hoc networks," Tech. Rep., EPFL, February 2004.

[22] M. Čagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On selfish behavior in CSMA/CA networks". *In Proceedings of the IEEE Conference on Computer Communications (INFOCOM),* Miami, Florida, USA, March 2005.

[23] D. Fudenberg and J. Tirole, *Game Theory*, The MIT Press, 1991.

[24] T. Başar and G.J. Olsder, *Dynamic Noncooperative Game Theory*, SIAM, 1999.