

Le reti LAN senza fili (**WLAN**, *Wireless LAN*) si impongono come modello laddove la connessione guidata via cavo è impossibilitata o costosa o inutilizzabile come nel caso di postazioni mobili. Molto spesso le WLAN si integrano a LAN su cavo preesistenti (*wired LAN*), spesso di tipo Ethernet 802.3, per consentire a determinate stazioni di connettersi a una rete cablata classica.

Lo standard IEEE che si occupa delle specifiche per le reti Wireless LAN è l'IEEE 802.11\*.

Nel 1999 l'azienda statunitense Interbrand Inc. ha coniato il termine **Wi-Fi** (si può anche tradurre in *Wireless Fidelity*) che spesso è usato come sinonimo di WLAN 802.11 (anche in questo testo).

Ad oggi (2012) lo standard 802.11\* si è evoluto fino alla versione 802.11n, anche se i vari standard nel tempo sono sostanzialmente considerati interoperabili. In sintesi:

Standard	Anno	Frequenza (GHz)	Bit rate (Mbit/s)
802.11	1997	2,4	1, 2
802.11a	1999	5,2; 5,4; 5,8	6, 9, 12, 18, 24, 36, 48, 54
802.11b	1999	2,4	1, 2, 5.5, 11
802.11g	2003	2,4	802.11a, 1, 2, 5.5, 11
802.11n	2009	2,4; 5,4	802.11g, 125, 144, 300

### Wi-Fi e ISM

La banda **ISM** (*Industrial, Scientific and Medical*) è una porzione dello spettro elettromagnetico che, sia a livello nazionale che internazionale (anche se con qualche differenza) viene riconosciuto come disponibile liberamente e gratuitamente (anche se solo all'interno di aree private).

Altri tipi di frequenze, invece, sono ristrette da norme severe e rilasciate solo a pagamento. Siccome è intrinsecamente impossibile confinare una trasmissione radio all'interno di una precisa area privata, le normative nazionali tendono a liberalizzare l'uso del Wi-Fi a prescindere da questa limitazione.

La comunicazione wireless Wi-Fi avviene in frequenza radio all'interno della banda di frequenze denominata **ISM** (*Industrial, Scientific and Medical*) che ogni paese riserva ad applicazioni di radiocomunicazioni non commerciali, ma per uso industriale, scientifico e medico.

Wi-Fi può usare antenne omnidirezionali o direttive. Le antenne omnidirezionali sono utilizzate per distribuire la connettività in aree private, relativamente poco estese come aziende private ma anche sottoforma di **hotspot** pubblici (esempio, alberghi, aeroporti, uffici pubblici, ecc.) che attraverso una connessione Wi-Fi spesso offrono un accesso a WAN Internet.

Siccome la comunicazione avviene tramite onde radio, essa può essere messa in crisi da particolari condizioni (ostacoli come alberi, pilastri, muri di edifici), cosicché la portata dei dispositivi Wi-Fi non sempre rispetta i valori promessi, che sono circa 30 m per ambienti interni e circa 100 m per ambienti esterni.

Le antenne direttive sono in genere utilizzate per connettività più estese e sono poste in luoghi strategici per superare le barriere fisiche ambientali.

Come tutti i dispositivi che operano su frequenze radio, anche la tecnologia Wi-Fi genera *smog elettromagnetico* potenzialmente pericoloso per la salute. Bisogna ricordare però che la potenza degli apparati più diffusi è inferiore a quella emanata da un telefono cellulare.

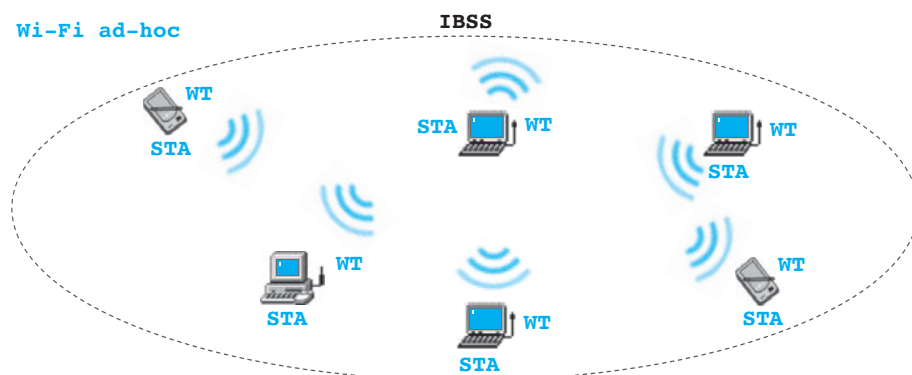
## 1 Modelli e terminologia

Le reti 802.11 possono presentarsi in due configurazioni differenti, tra loro trasparenti e conviventi: modello **ad-hoc**, in cui le stazioni sono paritarie, e modello **infrastructured**, in cui un apparato dedicato coordina il traffico delle stazioni ed è collegato ad una LAN.

In generale le singole stazioni operanti in Wi-Fi sono denominate **STA**, e sono dotate di un modulo di ricestrasmissione denominato **WT** (*Wireless Terminal*).

Organizzare moduli wireless in configurazione **ad-hoc** è la soluzione ideale per connettività temporanea tipo conferenze, sale riunioni, attività di gruppo all'aperto, ecc. La modalità non richiede particolari configurazioni. È studiata dal gruppo di lavoro IETF denominato *MANET* (*Mobile Ad hoc NETWORKS*) e in questo testo non sarà trattata.

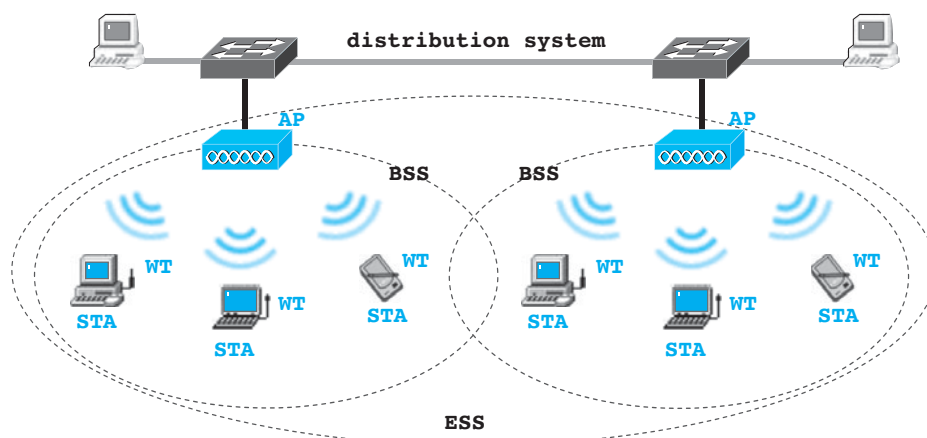
Wi-Fi ad-hoc



La modalità **infrastruttura** (infrastructured) è invece indicata per aggregare su una LAN preesistente e cablata (**distribution system**) uno o più gruppi di stazioni (**BSS**, *Basic Service Set*) attraverso un dispositivo dedicato denominato **Access Point** (**AP**).

Un insieme di BSS che si attestano sullo stesso distribution system è detto **ESS** (*Extended Service Set*).

Wi-Fi infrastructured



Quasi tutti gli Access Point commerciali possono essere configurati per operare in quattro modalità differenti. Quando si utilizza l'AP con un'unica BSS, tipico delle utenze residenziali, si dice modalità **root** (che è quella di fabbrica). Se l'AP deve coordinarsi con altri AP per costituire una ESS, allora va configurato in modalità **bridge**. Se invece lo si volesse usare come ripetitore di segnale per un altro AP, lo si può configurare in modalità **repeater**. Infine, anche se misconosciuta, la modalità **client** consente di trasformare un AP in un modulo WT per una stazione sprovvista di modulo Wi-Fi. In questo caso l'AP va connesso con un cavo Ethernet alla stazione che intende utilizzarlo come modulo WT.

## 2 MAC di IEEE 802.11

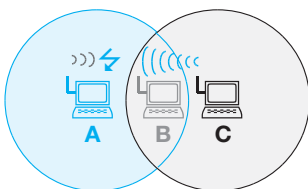
Wi-Fi è strutturalmente half duplex perché un dispositivo o trasmette o riceve. Il mezzo trasmissivo è denominato **etere**, e si può considerarlo condiviso esattamente come per Ethernet, ma in questo caso non è possibile rilevare le collisioni mentre si trasmette, come invece opera CSMA/CD.

Anche se un dispositivo wireless prima di trasmettere può stabilire se l'etere è impegnato, durante la trasmissione non è in grado di stabilire se altri dispositivi stanno a loro volta trasmettendo (collisione).

### 2.1 Stazione nascosta e stazione esposta

La modalità half duplex di Wi-Fi si può tradurre nel principio che, data la singola cella di portata radio di un dispositivo, è necessario che la stazione riceva da un solo trasmettitore alla volta.

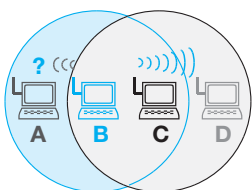
Questo comporta che possono presentarsi due problemi tipici delle comunicazioni radio half duplex, che impediscono del tutto di tentare un approccio con rilevamento di collisioni:



#### Stazione nascosta

In questo caso **B** sta ricevendo da **C**.

**A** vede il canale libero perché il segnale di **C** non gli arriva (**C** è fuori dalla cella di **A**), quindi trasmette, ma viola il principio (nella sua cella c'è una ricezione in corso).



#### Stazione esposta

In questo caso **D** sta ricevendo da **C**.

**B** vuole trasmettere ad **A**, ma non può farlo perché rileva una trasmissione nella sua cella e non sapendo dove si trovano **D** e **A**, non può rischiare. Però **B** avrebbe potuto trasmettere ad **A**, perché non è violato il principio (nessuna stazione sta ricevendo nella sua cella).

I due problemi suddetti vengono superati introducendo una breve fase di negoziazione, da effettuarsi prima di una qualsiasi trasmissione, tramite un protocollo **RTS/CTS** (*Request To Send/Clear To Send*).

In pratica il trasmettitore invita il ricevente (con un pacchetto RTS contenente l'indirizzo MAC del destinatario) a emettere un pacchetto (CTS), così che tutti i vicini del ricevente possano rendersi conto del tentativo di trasmissione in atto.

Qualunque stazione riceva un CTS viene a conoscenza del fatto che una ricezione sarà in corso nella propria cella.

Sarà così in grado di evitare di trasmettere, come nel caso della *stazione nascosta*: infatti A ha ricevuto il CTS di B.

Oppure potrà trasmettere, se necessario, come nel caso della *stazione esposta*: infatti B non ha ricevuto il CTS di D.

## 2.2 CSMA/CA

In Wi-Fi, non potendo rilevare le collisioni, si cerca di evitarle adottando il protocollo **CSMA/CA** (acronimo inglese di *Carrier Sense Multiple Access with collision avoidance*).

Per ottenere lo scopo il CSMA/CA usa la negoziazione RTS/CTS contenente il **NAV** (*Network Allocation Vector*): il pacchetto RTS, oltre all'indirizzo MAC del destinatario, contiene questo valore che indica la durata, in tempo, della trasmissione che si intende attuare. Anche il CTS di risposta contiene lo stesso NAV, in modo tale che qualsiasi stazione che riceve l'RTS o il CTS e non è coinvolta in questa sessione, possa impostare il valore di NAV in un proprio registro interno; questo valore verrà decrementato dalle stazioni in base al trascorrere del tempo, cosicché se una stazione ha il NAV diverso da zero, sa che è in atto una trasmissione nella sua cella. Questa condizione realizza un *Carrier Sense virtuale*: una stazione che vuole trasmettere sa che non può farlo se ha il NAV diverso da zero.

Dotato di Carrier Sense reale (rilevazione se l'etere è in idle, cioè libero) e del Carrier Sense virtuale (se il NAV è uguale a zero), una stazione può attuare il CSMA/CA:

- **Carrier Sense**. La stazione trasmittente cerca di determinare lo stato del mezzo valutando il contenuto di NAV ed ascoltando il mezzo. Il canale è considerato libero, quando sia il carrier sensing virtuale sia quello reale non rilevano attività.
- **Backoff**. Il trasmettitore attende che un proprio contatore interno di backoff vada a 0. Il contatore inizialmente parte da 7 e si decrementa nel tempo solo se il canale rimane libero. Se il canale viene occupato, il contatore rimane al valore che ha raggiunto, ma la stazione ritorna al Carrier Sense.
- **Carrier Sense virtuale**. La stazione emette un RTS. Se entro un intervallo di tempo ben definito (timeout), la stazione non riceve il CTS, deduce che c'è stata collisione. Allora riparte dal backoff, ma il valore iniziale del

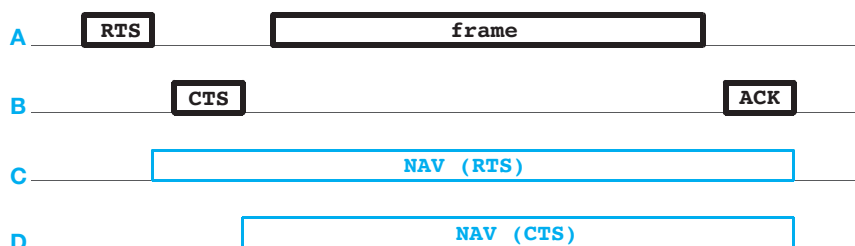
contatore di backoff verrà raddoppiato (esempio, varrà 15). Se invece il CTS viene ricevuto, il trasmettitore spedisce il pacchetto.

- **Data transmission.** Se entro un intervallo di tempo dopo aver spedito il pacchetto (timeout), il trasmettitore non riceve un riscontro dal ricevente (ACK), la procedura viene ripetuta da capo, e il contatore di backoff raddoppiato.

Il CSMA/CA di una trasmissione regolare dalla stazione A alla stazione B, con le stazioni C e D nelle vicinanze, può essere schematizzato dalla figura:

CSMA/CA semplificato

tempo →



## SSID

Il **SSID** (*Service Set Identifier*) è, in effetti, la stringa che identifica un Access Point quando l'utente ricerca la lista dei dispositivi Wi-Fi in una certa area. Anche se generalmente un SSID specifica un Access Point, in effetti non è esattamente identificabile con un solo e determinato AP, dato che è normale che in una ESS vi siano più AP con lo stesso SSID o che un singolo AP possa possedere più di un SSID.

Quando C e D ricevono RTS e CTS, impostano il proprio NAV: ora sanno che il canale sarà occupato, e per quanto tempo.

Si nota che la stazione C è nella cella di A (riceve l'RTS), mentre D no.

A sua volta D è nella cella di B (riceve il CTS).

Un tipico inconveniente del CSMA/CA avviene durante la fase di trasmissione del pacchetto: la collisione viene rilevata solo dopo averlo spedito interamente e aver atteso invano l'ACK: più il pacchetto è grande, più tempo viene perso nel rilevare la collisione. Per questo motivo il MAC di 802.3 **frammenta** i pacchetti lunghi e li ricompatta in ricezione.

Purtroppo il CSMA/CA così descritto comporta numerose collisioni proprio sul pacchetto RTS (e CTS): pur «funzionando», il protocollo impone numerose fasi di backoff ripetute e/o numerose scadenze di timeout nella fase di Carrier Sense virtuale che fanno attendere inutilmente la stazione che deve trasmettere.

Per superare questo problema le stazioni si sincronizzano utilizzando un tempo base denominato **slot time** e suoi derivati: SIFS (<2 slot time), PIFS (SIFS+slot time), DIFS (SIFS+2 slot time), EIFS (SIFS+3 slot time).

L'analisi di queste sincronizzazioni, benché fondamentali per il funzionamento di 802.11, esula dagli obiettivi di questo testo.

## 3 Modalità infrastruttura

Nelle WLAN paritarie (o ad-hoc) è attivo solo il MAC descritto: ogni stazione può tentare di comunicare con ogni altra. Questa modalità è detta **DCF** (*Distributed Coordination Function*).

Nelle WLAN centralizzate (o infrastrutturate) è attiva anche una seconda modalità di comunicazione, detta **PCF** (*Point Coordination Function*).

Questa seconda modalità fu prevista per il traffico real time (esempio, flussi audio/video), ma nel 2005 fu resa obsoleta perché non ritenuta sufficientemente efficace (IEEE 802.11e-2005/2007).

La modalità operativa effettivamente utilizzata nei dispositivi Wi-Fi commerciali è quindi la **DCF** nella quale, in modalità infrastruttura, l'AP funge da **nodo coordinatore** (*point coordinator*).

Le comunicazioni tra le stazioni, infatti, dovranno compiere due passi nella rete senza fili: un primo passo dalla stazione sorgente all'AP e un secondo passo dall'AP alla stazione destinazione.

L'AP si serve dell'invio periodico di uno speciale pacchetto «broadcast» detto **beacon** che contiene varie informazioni generali come per esempio un valore per sincronizzare le stazioni (*timestamp*) e un valore per annunciarsi alle stazioni (**SSID**, *Service Set Identifier*). Il SSID è una stringa ASCII che consente l'identificazione dell'AP da parte dei client Wi-Fi.

### 3.1 Accesso alla rete

Una volta attivato l'AP e collegato alla rete cablata, si pone il problema dell'accesso alla rete wireless delle stazioni (STA). Esso avviene attraverso quattro fasi:

- **Scansione.** È il primo passo che una stazione deve intraprendere per individuare quante e quali reti Wi-Fi sono disponibili nel suo raggio di azione. La stazione può effettuare una scansione passiva, ovvero attendere i vari pacchetti di beacon circolanti nella sua cella e decodificare il SSID in essi contenuto. Oppure può richiedere espressamente la risposta ad uno speciale pacchetto (*probe request*) da parte degli AP disponibili sul suo canale.
- **Autenticazione.** Una volta selezionato il SSID, è necessario che la stazione si autentichi prima di poter operare. L'autenticazione può avvenire in diversi modi, anche se l'algoritmo di autenticazione con chiave condivisa basato su WPA2 è ritenuto accettabile.

Per una procedura di autenticazione più affidabile si può optare per l'utilizzo di servizi di autenticazione basati sullo standard **RADIUS** (*Remote Authentication Dial In User Service*, RFC 2865).

- **Associazione.** Una volta autenticata, la stazione richiede di poter entrare a far parte del BSS relativo all'AP. L'AP in questa fase memorizza in un buffer interno l'indirizzo MAC della stazione in modo da inserirla nel novero degli host raggiungibili dalla rete cablata Ethernet a cui è connesso.
- **Roaming.** Le stazioni mobili, per definizione, potrebbero spostarsi all'interno dell'ESS tra le varie BSS (tra un Access Point e un altro). Wi-Fi garantisce questi passaggi senza l'interruzione del servizio, mediante fasi di disassociazione/associazione determinate dalla circolazione dei pacchetti di beacon.

## 4 Il pacchetto MAC

In una rete Wi-Fi circolano vari tipi di pacchetti: pacchetti di **controllo**, (ACK, RTS e CTS); pacchetti di **gestione**, (beacon, probe request, probe response, association request, association response, ecc.) e pacchetti **dati**.

### WPA2?

**WPA2** (*Wi-Fi-Protected Access*) è lo standard che dal 2004 offre la sicurezza per i dispositivi Wi-Fi: tramite la scelta di una chiave condivisa (**PSK**, *Pre-Shared Key*) e usando l'algoritmo crittografico **AES** (*Advanced Encryption Standard*) garantisce, ad oggi, la protezione delle sessioni di comunicazione radio.

Purtroppo per un lungo periodo di tempo le PSK fornite dai più importanti costruttori di apparati Wi-Fi (Telecom, Fastweb e altri) si potevano dedurre, con opportuni calcoli, dal MAC address del dispositivo e/o dal SSID preimpostati dal costruttore.

Se l'utente non cambiava la chiave segreta fornita con l'apparecchio (la PSK), i dati pubblici del MAC address e del SSID consentivano di ricavarla: la rete Wi-Fi ora diventava utilizzabile da chiunque fosse nel raggio di portata dell'Access Point.

Un algoritmo potenzialmente sicuro (AES, 2012) non può nulla circa una erronea o ingenua scelta della chiave.

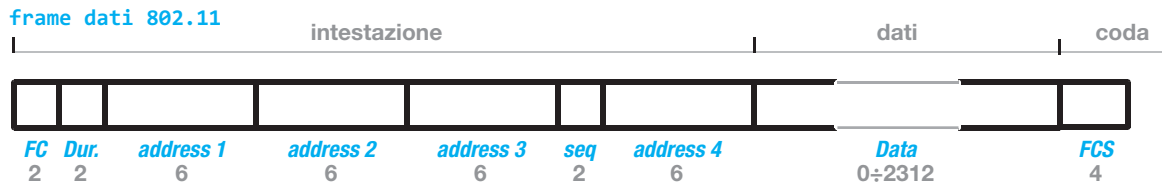
### Radiotap header

Il **radiotap header** è una sequenza di byte che compare prima del frame 802.11, la cui lunghezza è sempre specificata dal terzo e quarto byte (little endian). Dato un pacchetto Wi-Fi in sequenza di ottetti, per raggiungere il pacchetto MAC di 802.11 (cioè il primo campo *Frame Control*) bisogna quindi saltare un numero di byte equivalente alla lunghezza dell'header radiotap.

I byte del radiotap header non circolano effettivamente in rete, ma sono creati dal driver che cattura i pacchetti. In essi è contenuta una sintesi delle informazioni del livello 1 Fisico di 802.11, come la frequenza del canale in uso, il segnale dell'antenna e la qualità del segnale attuale.



Il pacchetto dati di 802.11\* è piuttosto articolato:



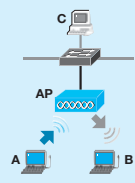
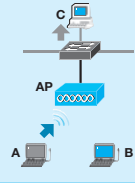
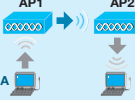
I campi del pacchetto:

- **FC**, *Frame Control*, campo codificato a bit che dispone i parametri dell'attuale trasmissione (da peso 15 a peso 0):
  - *SubType* (4 bit), specifica se RTS o CTS.
  - *Type* (2 bit), indica se pacchetto dati o pacchetto di servizio.
  - *Version* (2 bit), indica il numero di versione del protocollo.
  - *Order* (1 bit), indica se la sequenza in corso è ordinata.
  - *P* (1 bit), indica che il pacchetto è protetto (criptato).
  - *More* (1 bit), indica il frammento di un pacchetto.
  - *Pwr* (1 bit), usato dall'AP per sospendere una stazione.
  - *Retry* (1 bit), specifica se si tratta di un pacchetto ritrasmesso.
  - *MF* (1 bit), specifica se pacchetto frammentato.
  - *ToDS/FromDS* (2 bit), tipo di circolazione.
- **Dur.**, *Duration*, che indica la durata del pacchetto in termini di lunghezza. Serve per impostare il NAV.
- **address1**, **address2**, **address3**, **address4**, indirizzi MAC mittenti e destinatari, 6 ottetti come per 802.3, interpretati in base ai bit *ToDS* e *FromDS* del *Frame Control*.
- **S**, numero di sequenza del frammento.
- **FCS**, firma di integrità calcolata con CRC.

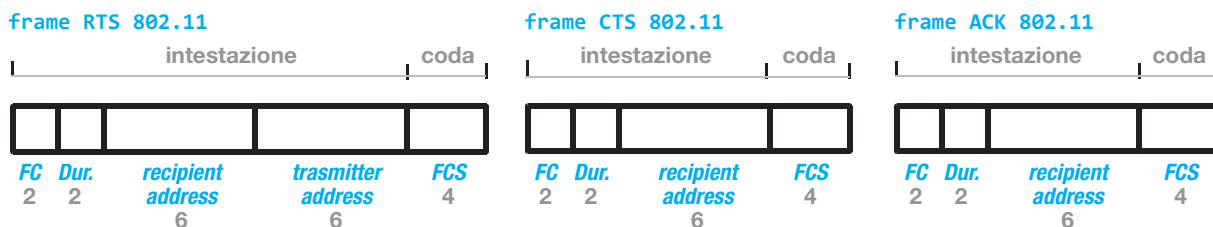
I tipi di circolazione dei pacchetti in Wi-Fi sono determinati dai bit *ToDS/FromDS* del *Frame Control* e seguono il seguente schema:

To DS	From DS	address 1	address 2	address 3	address 4	schema	nota
0	0	B	A	AP <sup>(NB)</sup>	–		solo modo ad-hoc
0	1	B	AP	A	–		downlink
0	1	B	AP	C	–		downlink

NB. In una combinazione **ad-hoc** una delle stazioni, di solito quella che viene avviata per prima, assume il compito di **master** per le altre, ovvero ha l'incarico di orchestrare la comunicazione come se fosse in Access Point, inviando regolarmente i pacchetti di beacon. L'identificazione della rete ad-hoc, ovvero della IBSS viene definita dalla stazione master generando un indirizzo MAC pseudocasuale (il contenuto del campo address 3).

1	0	AP	A	B	-		uplink
1	0	AP	A	C	-		uplink
1	1	AP2	AP1	B	A		interlink

I **pacchetti di controllo** hanno una struttura più semplice, senza parte dati e con un'intestazione breve:



**RTS** (*Type/Subtype* = 01/1011): Come per il pacchetto dati il campo *Frame Control* contiene la codifica a bit descritta; il campo *Duration* serve per impostare i NAV, e conterrà il valore temporale della dimensione del pacchetto che si intende trasmettere sommato al valore temporale di un CTS e un ACK necessari per completare la sessione. *Recipient address* contiene l'indirizzo MAC della stazione destinataria candidata, mentre *transmitter address* è l'indirizzo MAC del mittente.

**CTS** (*Type/Subtype* = 01/1100): Come per il pacchetto dati il campo *Frame Control* contiene la codifica a bit descritta; il campo *Duration* serve per impostare i NAV, e conterrà il valore temporale ricavato dal frame RTS appena ricevuto sottratto del valore temporale di un CTS. *Recipient address* contiene sempre l'*address 2* della trama dati ricevuta. Infatti, se si consulta la tabella precedente, in *address 2* compare sempre il destinatario della trama Wi-Fi.

**ACK** (*Type/Subtype* = 01/1101): Come per il pacchetto dati il campo *Frame Control* contiene la codifica a bit descritta; il campo *Duration* serve per impostare i NAV, ma solo se si tratta del riscontro a un pacchetto frammentato (cioè non l'ultimo di una sequenza). In questo caso è uguale al valore *Duration* contenuto nella trama dati ricevuta meno il tempo equivalente a un ACK. Se invece si tratta del riscontro all'ultimo frammento di un pacchetto (o di un pacchetto unico), situazione deducibile dal campo *More* del *Frame Control*, allora contiene 0. *Recipient address* contiene sempre il destinatario della trama Wi-Fi, come in CTS.





## Trama 802.11

Data una sequenza di byte che rappresenta l'intestazione di una trama 802.11, scrivere un programma in linguaggio C che ne decodifichi il tipo.

### LAYOUT

Frame 802.11: c4 00 68 00 00 0c 41 82 b2 55 55 09 cb 58 0e 00

Frame control=c400h (1100010000000000b): TYPE control; SUBTYPE CTS

Un programma in linguaggio C potrebbe essere:

```
00 #include <stdio.h>
01 #include <string.h>
02
03 #define TYPEMASK      (0x0c) //0ch = 00001100b
04 #define TYPE_MNGMT    (0x00) //00h = 00000000b
05 #define TYPE_CTRL     (0x04) //04h = 00000100b
06
07 #define SUBTYPEMASK   (0xf0) //f0h = 11110000b
08 #define SUBTYPE_RTS   (0xb0) //b0h = 10110000b
09 #define SUBTYPE_CTS   (0xc0) //c0h = 11000000b
10 #define SUBTYPE_ACK   (0xd0) //d0h = 11010000b
11
12 char* DectoszBin(int quale, int contenitore, char* szbinary);
13
14 int main (void)
15 {
16     unsigned char aFrame[14]={0xc4,0x00,0x68,0x00,0x00,0x0c,0x41,0x82,0xb2,0x55,0x55,0x09,0xcb,0x58};
17     char szMsg[80], szBinary[17];
18     int i;
19
20     printf("Frame 802.11: ");
21     for (i=0;i<16;i++) printf("%02x ",aFrame[i]);
22
23     strcpy(szMsg,"TYPE ");
24     if ((TYPEMASK & aFrame[0]) == TYPE_CTRL)
25     {
26         strcat(szMsg,"control; SUBTYPE ");
27         if ((SUBTYPEMASK & aFrame[0]) == SUBTYPE_ACK)
28             strcat(szMsg,"ACK");
29         else
30         {
31             if ((SUBTYPEMASK & aFrame[0]) == SUBTYPE_CTS) strcat(szMsg,"CTS");
32             else strcat(szMsg,"RTS");
33         }
34     }
35     else
36         strcat(szMsg,"management");
37
38     printf("\nFrame control=%02x%02xh (%sb): %s",
39         aFrame[0],aFrame[1],DectoszBin(aFrame[0]*0x100+aFrame[1],16,szBinary),szMsg);
40
41     return 0;
```

```

42 }
43
44 char* DectoszBin(int quale, int contenitore, char* szbinary)
45 {
46     unsigned int mask;
47     int i;
48
49     mask = 0;
50     mask = 1 << (contenitore-1);
51
52     i = 0;
53     while(mask)
54     {
55         if (quale & mask) szbinary[i] = '1'; else szbinary[i] = '0';
56         mask = mask >> 1;
57         i++;
58     }
59     szbinary[i] = 0;
60     return szbinary;
61 }

```

Dopo aver definito le maschere di bit e i valori da confrontare (**righe 3÷10**), si analizza il primo byte del pacchetto che è anche il primo byte del campo *Frame Control*.

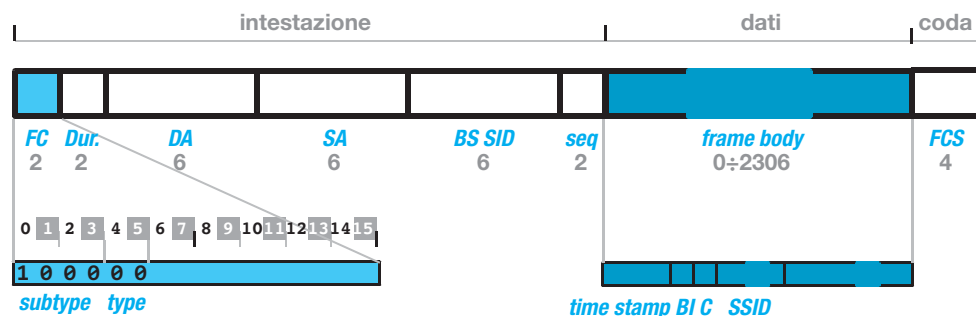
Su questo valore si eseguono i controlli per determinare il *Type* (**righe 24, 35 e 36**) quindi, se si tratta di un frame di controllo, si eseguono i test per determinare il *Subtype* (**righe 27, 31 e 32**).

Il programma fa uso della funzione *DectoszBin()* per stampare il formato binario dell'intero *Frame Control* su 16 bit.

I **pacchetti di gestione** hanno l'intestazione in comune e sono distinti in base al valore della coppia di campi *Type/Subtype* del *Frame Control*.

Il più importante pacchetto di management è il **beacon**, che è «marchiato» con la sequenza 00/1000 nella coppia *Type/Subtype*:

frame beacon 802.11



Almeno nella modalità infrastruttura, il campo DA (*Destination Address*) contiene l'indirizzo MAC di broadcast ff-ff-ff-ff-ff-ff che impone a tutte le stazioni nel BSS di trattenere il pacchetto.

Si nota che nel campo dati il beacon trasporta il *timestamp* (per sincronizzare le stazioni) e il SSID, di lunghezza variabile ma entro i 32 caratteri Ascii per annunciarsi alle stazioni.

## Frame 802.11

Data la seguente trama 802.11 in esadecimale, decodificala il contenuto:

```
00 00 18 00 8e 58 00 00 10 02 6c 09 a0 00
54 00 00 2b 00 00 9f 61 c9 5c 80 00 00 00
ff ff ff ff ff ff 00 0c 41 82 b2 55 00 0c
41 82 b2 55 50 f8 89 f1 d4 1b 01 00 00 00
64 00 11 04 00 07 49 74 69 73 2e 50 52 .. ..
```

Prima di tutto si individua e si salta l'header del radiotap, per raggiungere il pacchetto 802.11.

La lunghezza del radiotap header si trova nel terzo e quarto byte, in little endian, ovvero  $LE(18h\ 00h) = 0018h = 24$ .

Il 25mo e 26mo byte sono quindi il **Frame Control**: (80h 00h), che in binario valgono 10000000b e 00000000b.

Si nota che i quattro bit MSB del primo byte (**Sub-type**) valgono 1000b e i due successivi (**Type**) valgono 00b: il pacchetto è un **beacon**.

Si nota anche che la circolazione è dedicata ad una rete **ad-hoc**, essendo la coppia **ToDS/ FromDS** uguale a 00b.

Il campo successivo, **Duration**, su due byte, (00h 00h) vale 0: i NAV non saranno impostati.

L'indirizzo di destinazione **DA** è, come spesso succede per i pacchetti beacon, un indirizzo MAC di broadcast (ffh ffh ffh ffh ffh ffh), mentre l'indirizzo mittente **SA** è l'indirizzo MAC dell'Access Point (00h 0ch 41h 82h b2h 55h). In particolare l'OUI 00:0C:41 appartiene a Cisco Inc.

I sei byte successivi sono il **BSSID**, e coincidono con l'indirizzo MAC dell'Access Point. Dopo il campo BSSID si trovano i due byte del **numero di sequenza** dei pacchetti Wi-Fi, (50h f8h). Il valore va letto in little endian, ovvero  $LE(50h\ f8h) = f850h$  in cui i 12 bit MSB sono il numero di sequenza  $f85h = 3973$ , mentre i restanti 4 bit sono il **numero di frammento** 0h=0.

Ora comincia il **frame body** del beacon.

Il primo campo **Timestamp** è rappresentato su 8 byte: (89h f1h d4h 1bh 01h 00h 00h 00h). Va interpretato il little endian, ovvero  $LE(89h\ f1h\ d4h\ 1bh\ 01h\ 00h\ 00h\ 00h) = 000000011bd4f189h = 4761907593$  microsecondi.

Infine, saltando i 4 byte per il **BI** (Beacon Interval) e **C** (Capabilities), abbiamo il SSID: il secondo byte del campo **SSID** indica la lunghezza della stringa (07h)=7 caratteri, ovvero (49h 74h 69h 73h 2eh 50h 52h) = Itis.PR.

## ESERCIZI PER LA VERIFICA ORALE

Saper rispondere ai **requisiti fondamentali** dà una sufficiente garanzia per sentirsi pronti all'interrogazione. Saper anche rispondere ai **requisiti avanzati** dimostra una padronanza eccellente degli argomenti del capitolo.

### Requisiti fondamentali

- 1** Definire la locuzione «banda ISM» e indicare le portate classiche dei dispositivi Wi-Fi.
- 2** Ricordare l'ordine di grandezza del bitrate di una rete Wi-Fi e i principali standard IEEE.
- 3** Riportare le due configurazioni tipiche per una rete Wi-Fi ed elencare la terminologia dei componenti coinvolti.
- 4** Spiegare cosa comporta che una rete Wi-Fi sia di tipo *infrastructure*.
- 5** Indicare il nome del MAC di 802.11.
- 6** Ricordare la modalità trasmissiva strutturale di Wi-Fi e spiegare la differenza con la modalità di Ethernet.
- 7** Illustrare la funzione del NAV nella comunicazione Wi-Fi.
- 8** Spiegare la funzione dell'Access Point nella modalità Wi-Fi *infrastructure*.
- 9** Descrivere il SSID.
- 10** Definire e commentare la nozione di *roaming*.
- 11** Elencare e commentare le tre tipologie di pacchetti circolanti in una rete Wi-Fi.
- 12** Illustrare la funzione del pacchetto di *beacon*.
- 13** Disegnare su un foglio lo schema di una rete Wi-Fi con una BSS e commentare la trasmissione da una STA a un'altra STA e da una STA a un nodo della rete *infrastructure*.

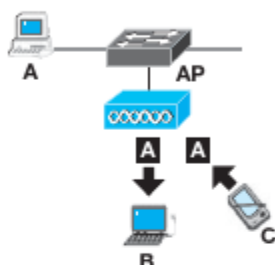
### Requisiti avanzati

- 1** Commentare le terminologie IBSS, ESS e BSS per le reti Wi-Fi.
- 2** Ricordare le quattro configurazioni base per un Access Point.
- 3** Illustrare il problema della stazione nascosta.
- 4** Illustrare il problema della stazione esposta.
- 5** Descrivere il meccanismo RTS/CTS e il *carrier sense* virtuale.
- 6** Spiegare perché è bene che il MAC di 802.11 frammenti i pacchetti troppo ampi.
- 7** Elencare e commentare le fasi di accesso ad una rete Wi-Fi da parte di una stazione STA.
- 8** Descrivere il *radiotap* header e la sua funzione.
- 9** Spiegare la funzione del campo ToDS/FromDS del campo Frame Control di 802.11.
- 10** Elencare e commentare i tre pacchetti di controllo circolanti in Wi-Fi.
- 11** Spiegare per quale motivo il pacchetto di *beacon* deve essere di *broadcast*.
- 12** Descrivere il processo di autenticazione di una stazione su un Access Point.
- 13** Disegnare su un foglio lo schema di una rete Wi-Fi con due BSS (due AP) una stazione fissa sull'*infrastructure* e due stazioni mobili, una su ogni BSS. Mostrare le sei circolazioni possibili.

## ESERCIZI PER LA VERIFICA SCRITTA E DI LABORATORIO

I prerequisiti richiesti per affrontare questi esercizi sono la conoscenza degli operatori bitwise AND, OR, ecc... e le operazioni di maschera dei bit.

- 1** Dato il seguente schema di Wi-Fi infrastruttura, indicare il valore del campo ToDs/FromDs e dei quattro indirizzi sulle tratte indicate.



### Svolgimento

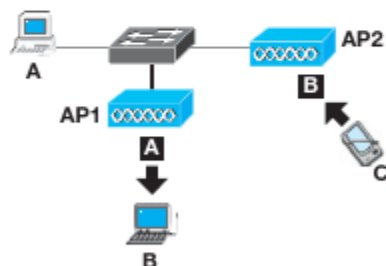
Il primo pacchetto (da A a B) è di tipo downlink, quindi:

ToDS	FromDS	addr1	addr2	addr3	addr4
0	1	MAC B	MAC AP	MAC A	-

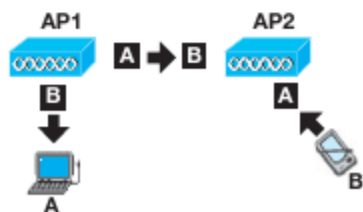
Il secondo pacchetto (da C a A) è di tipo uplink, quindi:

ToDS	FromDS	addr1	addr2	addr3	addr4
1	0	MAC A	MAC AP	MAC C	-

- 2** Dato il seguente schema di Wi-Fi infrastruttura, indicare il valore del campo ToDs/FromDs e dei quattro indirizzi sulle tratte indicate.

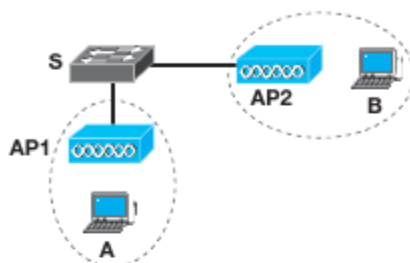


- 3** Dato il seguente schema di Wi-Fi infrastruttura, indicare il valore del campo ToDs/FromDs e dei quattro indirizzi sulle tratte indicate.

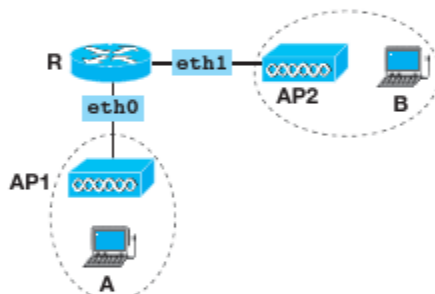


- 4** Dato il seguente schema di Wi-Fi infrastruttura in cui la stazione A spedisce un pacchetto a B,

riportare l'elenco degli indirizzi MAC coinvolti su ogni tratta (4), comprese quelle «wired».



- 5** Dato il seguente schema di Wi-Fi infrastruttura in cui la stazione A spedisce un pacchetto a B, riportare l'elenco degli indirizzi MAC coinvolti su ogni tratta (4), comprese quelle «wired».



- 6** Scrivere un programma che legge da un file di testo la sequenza di un frame 802.11 e ne mostri a schermo le caratteristiche.

Layout:

### OUTPUT

```
Frame 802.11:
00 00 18 00 8e 58 00 00 10 02 6c 09 a0 00 54
00 00 2b 00 00 9f 61 c9 5c 80 01 00 00 ff ff
ff ff ff ff 00 0c 41 82 b2 55 00 0c 41 82 b2
55 50 f8 89 f1 d4 1b 01 00 00 00 64 00 11 04
00 07 43 6f 68 65 72 65 72 01 08 82 84 8b 96
24 30 48 6c 03 01 01 05 04 00 01 00 00 2a 01
02 2f 01 02 30 18
Radio Tap length: 24 (18h)
Frame Control: 80h 01h, beacon, downlink
address 1: ff ff ff ff ff ff
address 2: 00 0c 41 82 b2 55
address 3: 00 0c 41 82 b2 55
```

# Spedire e leggere su LAN: pcap

# A4

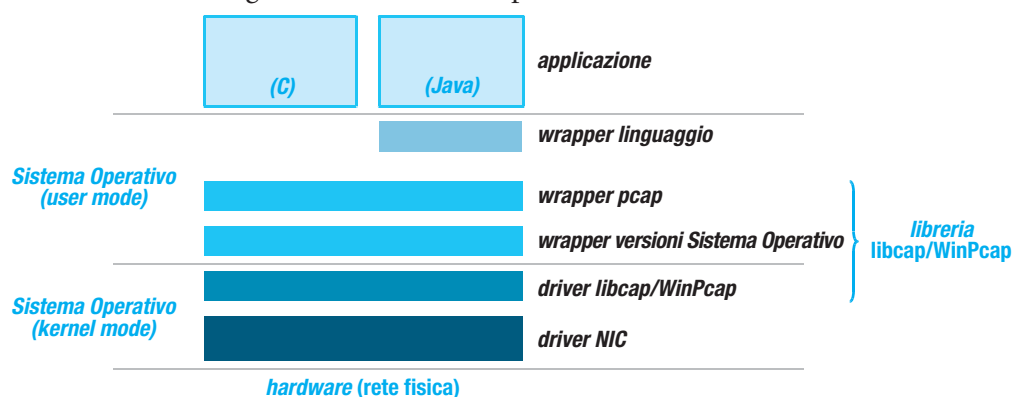
Con **pcap** (*packet capture*) si denomina una serie di funzioni raccolte in una **API** (*Application Programming Interface*) per leggere, analizzare e studiare il traffico di una rete. In pratica si tratta di uno strumento software sottoforma di libreria per scrivere programmi di **sniffing**. In particolare, pcap è realizzato su sistemi Unix-like in una libreria denominata **libpcap**, mentre su sistemi Windows è disponibile un «porting» della stessa, denominata **WinPcap**. Queste due librerie Open Source, pur operanti su sistemi operativi differenti, offrono la stessa interfaccia per il codice che intende utilizzarle.

Sia libpcap che WinPcap sono scritte in linguaggio C, cosicché altri linguaggi come Java e i linguaggi della famiglia .NET possono utilizzarle tramite uno strato software intermedio «adattatore» (wrapper).

Applicando il wrapper, se il linguaggio utilizzato per scrivere i programmi con libpcap/WinPcap è a sua volta portabile, si ottengono applicazioni multi-piattaforma a livello di codice sorgente: spostando il codice del progetto sorgente da una piattaforma all'altra e utilizzando di volta in volta la libreria relativa, la ricompilazione del progetto dà origine alla stessa applicazione senza dovere modificare il codice del programma. Per esempio, è quello che succede se si utilizza il linguaggio Java e, seppur relativamente, anche il linguaggio C.

La libreria WinPcap, nata verso la fine degli anni '90 in Italia come oggetto di una tesi universitaria, ebbe un successo inaspettato a livello mondiale. La libreria diede origine alle applicazioni di analisi di rete più diffuse e famose al mondo, **WinDump** e Ethereal (oggi denominato **Wireshark**), entrambe dipendenti dalla libreria libcap/WinPcap.

La struttura di queste librerie è un ottimo esempio di stratificazione del software, in cui ogni strato uniforma le funzioni dello strato inferiore per fornire un servizio generale allo strato superiore:





Il *driver NIC* è fornito dal sistema operativo, e opera in kernel mode. È il software che legge e scrive i pacchetti sulla scheda di rete (interfaccia fisica o NIC). Per Windows è noto come driver NDIS.

La libreria libpcap/WinPcap aggiunge un driver che comunica con questo per intercettare i pacchetti letti o scritti da e per l'interfaccia di rete. Il driver sarà di un formato tipico Linux se operante per sistemi Unix-like; di tipo *sys* per sistemi Windows.

Questo è il primo strato della libreria: rendere disponibile al livello superiore il traffico reale dei pacchetti. Per WinPcap il nome del driver è **npf.sys**.

Il secondo livello della libreria ha il compito di uniformare le chiamate per le diverse versioni del medesimo sistema operativo. Per esempio le varie distribuzioni Linux (che potrebbero avere driver di rete leggermente diversi) o le versioni Windows tipo XP, 2003/2008, Seven (che hanno driver di rete differenti). Per WinPcap il nome della libreria è **packet.dll**.

Infine il terzo livello della libreria ha il compito di fornire una interfaccia comune per le applicazioni, conforme al progetto originale BPF (Berkeley Packet Filter, 1993) denominato appunto pcap. Per WinPcap il nome della libreria è **wpcap.dll**.

Come si vede nella figura precedente, siccome il terzo livello (pcap) si offre alle applicazioni in linguaggio C, ogni linguaggio differente dal C deve fornire un ulteriore wrapper per «trasportare» quelle stesse chiamate alla propria sintassi.

Le librerie pcap (libpcap/WinPcap) lavorano in modo **promiscuo**: ricevono e spediscono pacchetti con qualsiasi indirizzo MAC. In ricezione leggono tutto il traffico sul mezzo, non solo quello destinato alla scheda di rete su cui stanno operando; in spedizione possono scrivere pacchetti con indirizzo MAC sorgente diverso da quello della scheda su cui operano.

Per scrivere un programma che spedisce e riceve pacchetti su rete LAN è quindi necessario installare la libreria libpcap/WinPcap sul sistema operativo, rispettivamente Linux o Windows. Allo stesso modo, una volta installata la libreria, sarà anche possibile utilizzare il programma tcpdump/WinDump rispettivamente per Linux e Windows e il programma Wireshark (in questo caso ha lo stesso nome su entrambi i sistemi operativi).

L'installazione delle librerie nei rispettivi sistemi operativi è immediata e non serve configurare nessun altro elemento del sistema. Le librerie sono per poco o per nulla invasive, pertanto se ne consiglia l'installazione (vedi [Appendice. Installazione libpcap/WinPcap per Linux e Windows](#)).

## 1 tcpdump/WinDump

Questo programma è un analizzatore di pacchetti (packet sniffer) a linea di comando, operante allo stesso modo su Linux e Windows nelle rispettive versioni (tcpdump e WinDump). Non ha bisogno di installazione ed è costituito da un solo file eseguibile. Naturalmente il funzionamento di tcpdump/WinDump dipende dalla preventiva installazione sul sistema della libreria libpcap/WinPcap.

Per gli scopi di questo testo servono solo due comandi del programma:

il comando per numerare le interfacce fisiche sulla macchina che si sta utilizzando e il comando per leggere i pacchetti circolanti:

```
WINDOWS
C:\utility\WinDump>WinDump -?
WinDump version 3.9.5, based on tcpdump version 3.9.5
WinPcap version 4.1.2 (packet.dll version 4.1.0.2001), based on libpcap version
1.0 branch 1_0_rel0b (20091008)
Usage: WinDump [-aAdDeflLnNOPqRStuUvxxX] [-B size] [-c count] [-C file_size]
               [-E algo:secret] [-F file] [-i interface] [-M secret]
               [-r file] [-s snaplen] [-T type] [-w file]
               [-W filecount] [-y datalinktype] [-Z user]
               [ expression ]

C:\utility\WinDump>WinDump -D
1.\Device\NPF_{8EA8E280-7B76-41D5-9484-BE8FBD343BE1} (Microsoft)
2.\Device\NPF_{9CBC96A4-1E81-4305-9AC0-26BCBFA68534} (Broadcom half-mini WLAN)
3.\Device\NPF_{9DB73EC6-6FBD-472F-B13A-0070B9CD1EED} (Intel(R) Gigabit Connection)

C:\utility\WinDump>WinDump.exe -i 3 -e -n
WinDump.exe: listening on \Device\NPF_{9DB73EC6-6FBD-472F-B13A-0070B9CD1EED}
23:23:32.746829 00:1f:c6:24:7c:15 > 01:00:5e:40:00:00, ethertype IPv4 (0x0800),
length 157: 192.168.0.1.6771 > 239.192.0.0.6771: UDP, length 115

1 packets captured
1 packets received by filter
0 packets dropped by kernel

C:\utility\WinDump>
```

L'enumerazione sul personal computer dell'esempio conta 3 interfacce fisiche, tra le quali si può notare la NIC per la rete Ethernet cablata, la numero 3.

Ora si può usare il comando per catturare il traffico su questa interfaccia, con l'opzione **-i 3**.

Per interrompere la cattura dei pacchetti, premere Ctrl-c.

Per evitare di dover catturare qualsiasi pacchetto circolante, è possibile specificare un **filtro**, tra doppie virgolette, come ultimo parametro.

Per esempio "**ether host 5c:26:0a:29:be:34**" impone di leggere solo i pacchetti provenienti dalla scheda con indirizzo 5c:26:0a:29:be:34; oppure, specificando il filtro **arp**, verranno catturate solo le trame Ethernet che trasportano l'*Ethertype* 0806h di ARP.

Il comando funziona anche con la redirectione dell'output, con la sintassi **> nomefile** al termine del comando.

Molto utile l'opzione **-w [nomefile].cap**, che invece di mostrare l'output sullo schermo, salva su disco il file **[nomefile].cap**, il formato proprietario di Wireshark, affinché possa essere letto e decodificato nella sua interezza.

## 2 Spedire un pacchetto sulla LAN (Win32)

Sotto il sistema operativo Windows è possibile scrivere un programma C per Win32 che spedisca pacchetti utente direttamente su LAN Ethernet

utilizzando la libreria WinPcap installata e usando l'SDK Open Source denominato **WpdPack** fornito gratuitamente (cfr. [Appendice. Installazione libpcap/WinPcap per Linux e Windows](#)).

In questo caso sarà scritto un progetto per l'ambiente di sviluppo gratuito cross-platform **Code::Blocks 10.5** corredato da compilatore **gcc** per Win32.

L'SDK WpdPack non si installa, basta copiarlo in una cartella qualsiasi del disco. È costituito da due directory fondamentali, *Include* e *Lib*. La prima contiene i file di intestazione *.h* (*header*), necessari per compilare le due librerie statiche presenti nella cartella *Lib*, le citate *wpcap.lib* e *packet.lib*.

Le due librerie sono fornite anche in formato gcc, con i file *libwpcap.a* e *libpacket.a*.

Ora si crea un progetto con Code::Blocks di tipo Console Application, esempio *sendeth.cbp*.

Per comodità si ipotizza che la cartella del progetto *sendeth* sia stata creata in base al seguente schema:



Per consentire di compilare e linkare le librerie statiche che accederanno al driver *npf.sys*, il progetto va configurato così:

- 1) Avviato Code::Blocks e caricato il progetto, selezionarlo nella finestra *Management*, quindi agire sul menu:

*Project-Build options-Search directories-Compiler-Add*, e digitare:  
..<\..\Include.

In questo modo il compilatore cercherà i file di intestazione aggiuntivi proprio nella cartella *Include* dell'SDK WpdPack (osservare lo schema).

- 2) Agire di nuovo sul menu:

*Project-Build options-Search directories-Linker-Add*, e digitare:  
..\..\Lib.

In questo modo il linker cercherà le librerie aggiuntive proprio nella cartella *Lib* dell'SDK WpdPack (osservare lo schema).

- 3) Agire di nuovo sul menu:

*Project-Build options-Linker settings-Add*, e digitare:  
*wpcap*.

In questo modo il compilatore cercherà le funzioni invocate nel codice sorgente anche all'interno di questa libreria. Non è necessario fornire il percorso della libreria dato che è già stato fornito nei passi precedenti.



## Spedire un frame su Ethernet (Win32)

Scrivere un programma in linguaggio C per Win32 che spedisce un pacchetto Ethernet su una interfaccia di rete verso un destinatario qualsiasi.

Il codice prima enumera le interfacce fisiche presenti sul personal computer (**righe 20÷39**) e consente la scelta di una di esse.

Quindi attraverso il nome del driver corrispondente (alla scheda selezionata) lo apre, crea i 12 byte degli indirizzi MAC destinazione e sorgente, crea un vettore di dati (praticamente a caso, il pacchetto) e spedisce la trama.

Affinché **Code::Blocks** possa compilare il seguente codice Win32 per Windows è necessario importare, oltre alle librerie di WinPcap, la libreria di sistema iphlpapi (per l'API GetAdaptersInfo()): dal menu di Code::Blocks (*Project-Build options-Linker settings-Add*), digitare **iphlpapi**.

### OUTPUT

```
NIC 1: Driver alla rete LAN Bluetooth - Miniport
Nome: \Device\NPF_{6F059EA3-B726-454A-9992-A8A349010E2D}

NIC 2: Dell Wireless 1397 WLAN Mini-Card
Nome: \Device\NPF_{FC10EC63-E8D7-4714-8044-8EF72C54EA6F}

NIC 3: Intel(R) 82567LM Gigabit Network Connection
Nome: \Device\NPF_{C2541EC8-DA50-4F03-B94D-1602C8CCEDE4}

Scegliere una interfaccia: 3
```

Il codice potrebbe essere (in evidenza gli elementi della libreria WinPcap):

```
00 #include <windows.h>
01 #include <conio.h>      // per getche()
02 #include <iphlpapi.h>   // per GetAdaptersInfo()
03 #include <pcap.h>       // per la libreria WinPcap
04
05 #define NUMNIC 16 // quante schede NIC possibili
06 #define DIMNIC 80 // n. caratteri per il nome del driver
07
08 Int main(void)
09 {
10     IP_ADAPTER_INFO AdapterInfo[NUMNIC];
11     IP_ADAPTER_INFO* pAdapterInfo;
12     DWORD dw;
13     char *aszNome[NUMNIC]; // nomi dei driver di scheda
14     int i,j;
15     char ch;
16     char errbuf[PCAP_ERRBUF_SIZE+1];
17     pcap_t *fp;
18     u_char pkt[100];
19
20     dw = sizeof(AdapterInfo);
21     if (GetAdaptersInfo(AdapterInfo,&dw)!=NO_ERROR) return 1;
22
23     J=0;
24     pAdapterInfo = AdapterInfo;
25     while(pAdapterInfo)
26     {
27         if (pAdapterInfo->Type==MIB_IF_TYPE_ETHERNET)
28         {
29             aszNome[j] = (char *)malloc(DIMNIC);
30             sprintf(aszNome[j], "\\Device\\NPF_%s", pAdapterInfo->AdapterName);
31             printf("\nNIC %d: %s\nNome: %s\n", j+1, pAdapterInfo->Description, aszNome[j]);
32             j++;
33         }
34         pAdapterInfo = pAdapterInfo->Next; // il prossimo...
35     }
```

```

36
37 printf("\nScegliere una interfaccia: ");
38 ch = getche();
39 if ((ch - '0') > j) return 2;
40
41 j = ch - '0' - 1; // indice dell'interfaccia selezionata
42 if ((fp = pcap_open_live(aszNome[j], // apertura dell'interfaccia di rete
43                          65536,      // dimensione del pacchetto da catturare (non serve)
44                          1,          // impostazione del modo promiscuo
45                          1000,       // timeout di lettura (non serve)
46                          errbuf,     // stringa per eventuale errore
47                          ) == NULL)
48 {
49     fprintf(stderr, "\nImpossibile aprire l'interfaccia %s \n", aszNome[j]); return 3;
50 }
51 // Indirizzo MAC destinazione
52 pkt[0]=0x01; pkt[1]=0x01; pkt[2]=0x01; pkt[3]=0x01; pkt[4]=0x01; pkt[5]=0x01;
53 // Indirizzo MAC mittente
54 pkt[6]=0x02; pkt[7]=0x02; pkt[8]=0x02; pkt[9]=0x02; pkt[10]=0x02; pkt[11]=0x02;
55
56 // pacchetto da 100 byte, praticamente casuale...
57 for(i=12; i<100; i++)
58 {
59     pkt[i]= (u_char)i;
60 }
61
62 // Spedizione del pacchetto sull'interfaccia specificata
63 if (pcap_sendpacket(fp, pkt, 100) != 0)
64 {
65     fprintf(stderr, "\nErrore di spedizione: %s\n", pcap_geterr(fp)); return 4;
66 }
67
68 pcap_close(fp);
69 return 0;
70 }

```

Si noti l'impostazione del driver in modo promiscuo (**riga 44**).

Questo consente di spedire un pacchetto con indirizzi MAC destinazione e sorgente a proprio piacimento. In questo codice si sono scelti, rispettivamente 01: 01: 01: 01: 01: 01 e 02: 02: 02: 02: 02: 02.

In questo modo per attendere il pacchetto su un'altra macchina, basterà usare WinDump con un filtro sul mittente pari a 02: 02: 02: 02: 02: 02, evitando così di visualizzare decine di pacchetti circolanti.

Avviata una sessione di WinDump su una macchina qualsiasi sulla stessa LAN (ma nello stesso dominio di broadcast), si potrà verificare la ricezione del pacchetto:

```

WINDOWS
C:\utility\WinDump>windump -D
1.\Device\NPF_{8EA8E280-7B76-41D5-9484-BE8FBD343BE1} (Microsoft)
2.\Device\NPF_{9CBC96A4-1E81-4305-9AC0-26BCBFA68534} (Microsoft)
3.\Device\NPF_{9DB73EC6-6FBD-472F-B13A-0070B9CD1EED} (Intel(R) 82577LM Gigabit Net)

C:\utility\WinDump>windump -i 3 -n -e "ether host 02:02:02:02:02:02"
windump: listening on \Device\NPF_{9DB73EC6-6FBD-472F-B13A-0070B9CD1EED}
02:33:18.552128
02:02:02:02:02:02 > 01:01:01:01:01:01, ethertype Unknown (0x0c0d), length 100:
    0x0000:  0e0f 1011 1213 1415 1617 1819 1a1b 1c1d  .....
    0x0010:  1e1f 2021 2223 2425 2627 2829 2a2b 2c2d  ...!"#$%&'()*+,-
    0x0020:  2e2f 3031 3233 3435 3637 3839 3a3b 3c3d  ./0123456789:;<=
    0x0030:  3e3f 4041 4243 4445 4647 4849 4a4b 4c4d  >?@ABCDEFGHIJKLM
    0x0040:  4e4f 5051 5253 5455 5657 5859 5a5b 5c5d  NOPQRSTUVWXYZ[\
    0x0050:  5e5f                                     ^
                                                _

1 packets captured
1086 packets received by filter
0 packets dropped by kernel
C:\utility\WinDump>

```

Dopo aver elencato le interfacce sulla macchina prescelta per attendere il pacchetto (con il comando **WinDump -D**), si stabilisce che l'interfaccia su cui esso arriverà è la 3.

Ora si può ordinare al programma di attendere i pacchetti circolanti su questa interfaccia, usando un filtro sul mittente per evitare di catturare anche altri pacchetti. Dato che l'indirizzo MAC sorgente del frame spedito è 02: 02: 02: 02: 02, il comando risulta essere:

```
WinDump -i 3 -n -e "ether host 02:02:02:02:02:02"
```

Nel pacchetto ricevuto si notano i due indirizzi MAC mittente e destinatario e l'Ethertype «casuale» e non riconosciuto 0c0dh, quindi i successivi  $100 - 12 - 2 = 86$  byte del pacchetto spedito dal programma.

### 3 Spedire un pacchetto sulla LAN (Java)

Con Eclipse SDK è possibile scrivere un programma Java che spedisca pacchetti utente direttamente su LAN Ethernet utilizzando i driver fisici (esempio, npf.sys per Win32).

In questo caso sarà necessario aggiungere al progetto il wrapper delle librerie libpcap/WinPcap, denominato **jNetPcap API**, scaricabile dalla pagina [www.jnetpcap.com](http://www.jnetpcap.com) sotto licenza *Lgpl*.

Per l'installazione del wrapper libpcap/WinPcap decomprimere il file compresso scaricato in una cartella a scelta. Va quindi creata una libreria utente che conterrà il wrapper affinché un progetto possa utilizzarlo. La seguente procedura è valida sia per Windows che per Linux (jNetPcap è multiplatforma).

I passi per Eclipse:

#### 1) Creazione libreria che conterrà il wrapper.

menu *Window-Preferences-Java-Build Path-User Libraries-New* e assegnare un nome (esempio, *Jnetpcap*, come il nome del wrapper).

Ora si inserisce nella libreria il wrapper agendo su *Add External JARs*, selezionando il file *jnetpcap.jar* che si trova nella directory decompressa in precedenza. Espandere il ramo e cliccare su *Native Library-Edit*, selezionare la cartella radice del wrapper. Ora Eclipse ha configurato il wrapper e un progetto potrà utilizzarlo con il relativo *import org.jnetpcap.Pcap;* e *import org.jnetpcap.PcapIf;*

#### 2) Impostare il wrapper per un progetto.

Per inserire la libreria in un progetto Java, durante l'impostazione delle librerie di progetto includere anche la libreria utente: *Add Library-User library*, indicando il nome deciso al passo precedente.

Se si intende usare jNetPcap in un solo progetto, è sufficiente aggiungere il file *jnetpcap.jar* (contenuto nel file compresso scaricato dal sito) nel project build path del progetto.

Dopo aver selezionato *Project-Propertiers-Java Build Path-Configure Build Path* si deve aggiungere il file jar come *Add External JARs*.

Impostare quindi la library location del file jar come *Native library location*.