A1 Tecnologia IEEE 802.3

Metcalfe's Law

Il co-inventore di Ethernet, Robert Metcalfe, fu assunto nei primi anni '70 dalla Xerox PARC Inc. appena laureatosi al MIT, e dopo aver avuto notizia dei suoi lavori da laureando. Come disse in una intervista, sottopose i suoi calcoli sul progetto di Ethernet a Leonard Kleinrock (inventore delle reti switching, 1962) che li definì «spazzatura». In seguito fu proprio Kleinrock a coniare l'acronimo CSMA/CD.

Una rete Ethernet funzionò la prima volta l'11 novembre del 1973.

Ma nemmeno la Xerox PARC credette nel suo progetto, così decise di fondare una propria azienda per produrre e commercializzare Ethernet (3COM Inc., 1979).

Nello stesso periodo al MIT un autorevole studioso, ma anche consulente IBM, Jerome H. Saltzer, indicò all'intero mondo accademico la sua preferenza per Token Ring come tecnologia per reti su brevi distanze. Le maggiori aziende produttrici di hardware, IBM in testa, decisero quindi di investire in Token Ring, lasciando spazio alla 3COM di Metcalfe di operare sul mercato delle reti LAN, proponendo progressivamente Ethernet come più economica ed efficiente (in *James Pelkey, intervista a Bob Metcalfe, 16 febbraio 1988*).

La tecnologia di rete LAN **IEEE 802.3**, universalmente nota come rete Ethernet, è il tipo di rete locale più popolare e diffuso al mondo. Dopo la sua ideazione e ufficializzazione con il nome **Ethernet** tra il 1973 e il 1976 (Robert Metcalfe e David Boggs della Xerox PARC Inc.), lo IEEE e l'OSI ne descrissero lo standard nel documento IEEE 802.3 del 1983.

La prima realizzazione di IEEE 802.3 usò il cavo coassiale e prese il nome di 10BASE5 con bitrate a 10Mbit/s. In seguito 802.3 si impone sia su rame che su fibra ottica. Nel 1995 l'IEEE diede vita a 802.3u o **Fast Ethernet** denominata 100BASE-TX, a 100 Mbit/s; quindi nel 1999 venne emanato lo standard 802.3ab denominato 1000BASE-T a 1000 Mbit/s (1 Gbit/s, **Giga Ethernet**). Infine nel 2006 fu pubblicato lo standard IEEE 802.an denominato 10GBASE-T o Tera Ethernet a 10 Gbit/s.

L'idea originale di Metcalfe e Boggs prevedeva un formato del pacchetto leggermente diverso da quello ufficializzato da 802.3, ma dal 1997 con lo standard 802.3x, i due tipi di pacchetto possono circolare entrambi senza problemi su una qualsiasi LAN 802.3*.

Nella famiglia di documenti IEEE 802 sono presenti vari standard di rete locale oltre a 802.3: 802.4 Token Bus, 802.5 Token Ring, 802.6 DQDB, 802.8 FDDI e altre, ma anche tecnologie varie connesse alle reti locali, come 802.11 Wi-Fi o 802.16 WiMAX. Alcune di queste tecnologie di rete locale sono oggi cadute in disuso e i relativi documenti resi inattivi o *disbanded* (esempio, Token Bus, Token Ring, DQDB, FDDI).

Di notevole importanza, infine, il documento **IEEE 802.2 LLC** (*Logical Link Control*) che descrive lo strato di rete LAN a contatto con il livello 3 Network allo scopo di uniformare le LAN a prescindere dai vari livelli MAC e livello fisico adottati.

Nella sua prima ideazione Ethernet (d'ora in avanti sinonimo di 802.3) utilizza un modo di accedere al mezzo fisico condiviso da tutte le stazioni denominato **CSMA/CD** (acronimo inglese di *Carrier Sense Multiple Access with Collision Detection*, ovvero accesso multiplo tramite rilevamento della portante con rilevamento delle collisioni), detto anche *MAC di Ethernet*.

Le ragioni della sua grande diffusione sono dovute a:

- primogenitura: è stata la prima tecnologia per reti LAN ad essere realizzata e ingegnerizzata;
- economicità: la sua relativa semplicità consente di utilizzare componentistica hardware relativamente poco costosa;

- efficienza: malgrado alcuni elementi critici della sua architettura, si rivela efficiente per il tipo di traffico generato da un gruppo di utenti omogeneo;
- scalabilità: con l'introduzione degli switch, si possono organizzare LAN anche con migliaia di postazioni;
- è adeguata per veicolare i livelli WAN di Internet, cioè TCP/IP.

1 MAC di Ethernet

Con la sigla **MAC** (*Medium Access Control*) si intende la modalità con cui una rete locale gestisce il mezzo trasmissivo per coordinare trasmissioni e ricezioni. Ogni tipologia di rete locale ha un proprio MAC. Il MAC di Ethernet si chiama CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) ed è un protocollo di accesso multiplo al mezzo derivante da un precedente protocollo denominato ALOHA.

Si tratta di un protocollo totalmente distribuito, senza stazioni privilegiate e intrinsecamente **half duplex**, operante in **broadcast** su topologie logiche a **bus**: ogni stazione quando riceve un pacchetto ne legge sempre l'indirizzo MAC di destinazione e solo se coincide con il proprio ne inoltra una copia al livello superiore.

In trasmissione invece le cose sono un po' più complicate. Poiché i trasmettitori delle stazioni si trovano ad operare in «parallelo» sullo stesso mezzo, è importante evitare che più stazioni trasmettano contemporaneamente, situazione nota con il termine di **collisione**. Tuttavia il protocollo non esclude che ciò possa comunque avvenire e prevede un meccanismo di riconoscimento di tale evento da parte delle stazioni coinvolte, in modo che possano ritentare la trasmissione in un tempo successivo. Un protocollo di questo tipo è detto **a contesa**.

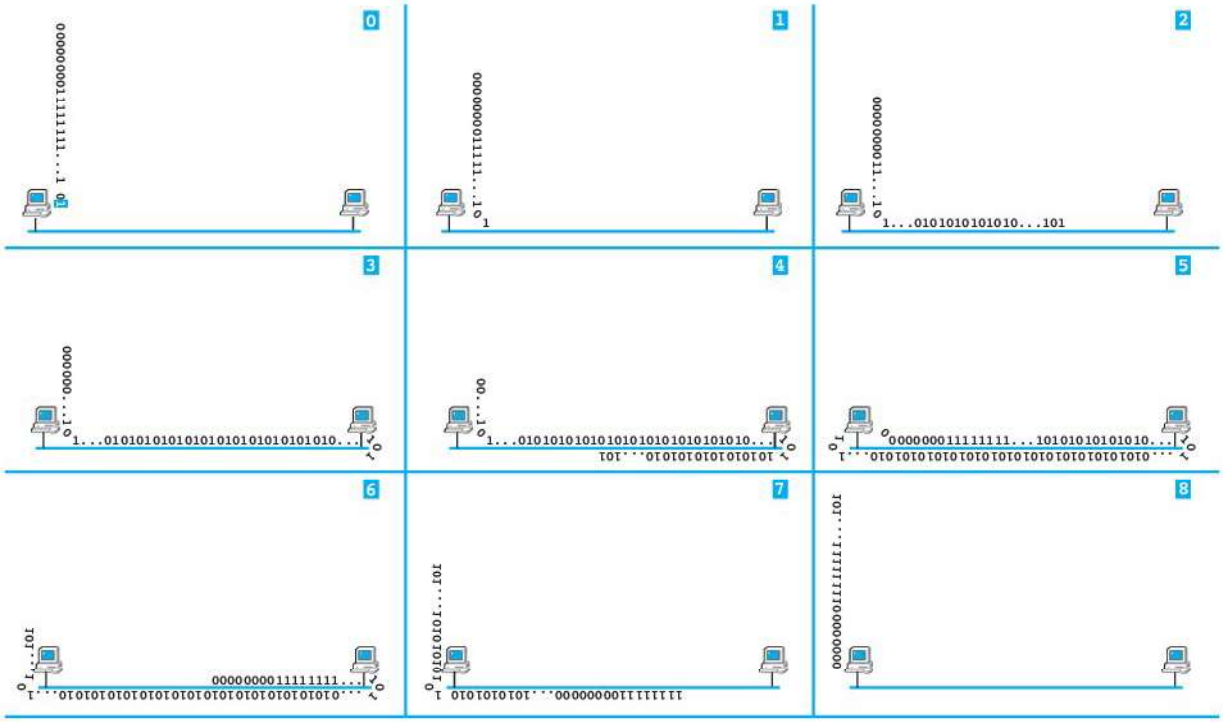
CSMA/CD opera in quattro fasi:

1) **Carrier sense**, ovvero rilevazione del segnale; ogni stazione che deve trasmettere, prima ascolta il bus e decide di trasmettere solo se questo è libero (*idle*). Questa fase è anche detta *listening before talking*.
2) **Multiple access**, ovvero possibilità di accesso multiplo; nonostante aver verificato il canale in stato di idle, è possibile che due o più stazioni, trovando anch'esse il canale libero, decidano di trasmettere.

   Questa situazione è più frequente di quanto non appaia a prima vista. Infatti se una stazione sta effettivamente trasmettendo, è molto probabile che altre stazioni inizino la fase di carrier sense, cioè si pongano in attesa di poter trasmettere. Quando la stazione che sta trasmettendo conclude la trasmissione, le stazioni in attesa vedranno tutte «contemporaneamente» il canale libero e tutte inizieranno a trasmettere (il tempo di propagazione dei bit sul mezzo NON è nullo).
3) **Collision detection**, ovvero determinazione di collisione; mentre una stazione trasmette, contemporaneamente ascolta i segnali sul mezzo trasmissivo, confrontandoli con quelli che emette lei stessa. In caso di ricezione discrepante, viene rilevata una **collisione**. Questa fase è anche detta *listening while talking*.

4) **Backoff**; se viene rilevata una collisione, la stazione intraprende le seguenti azioni:

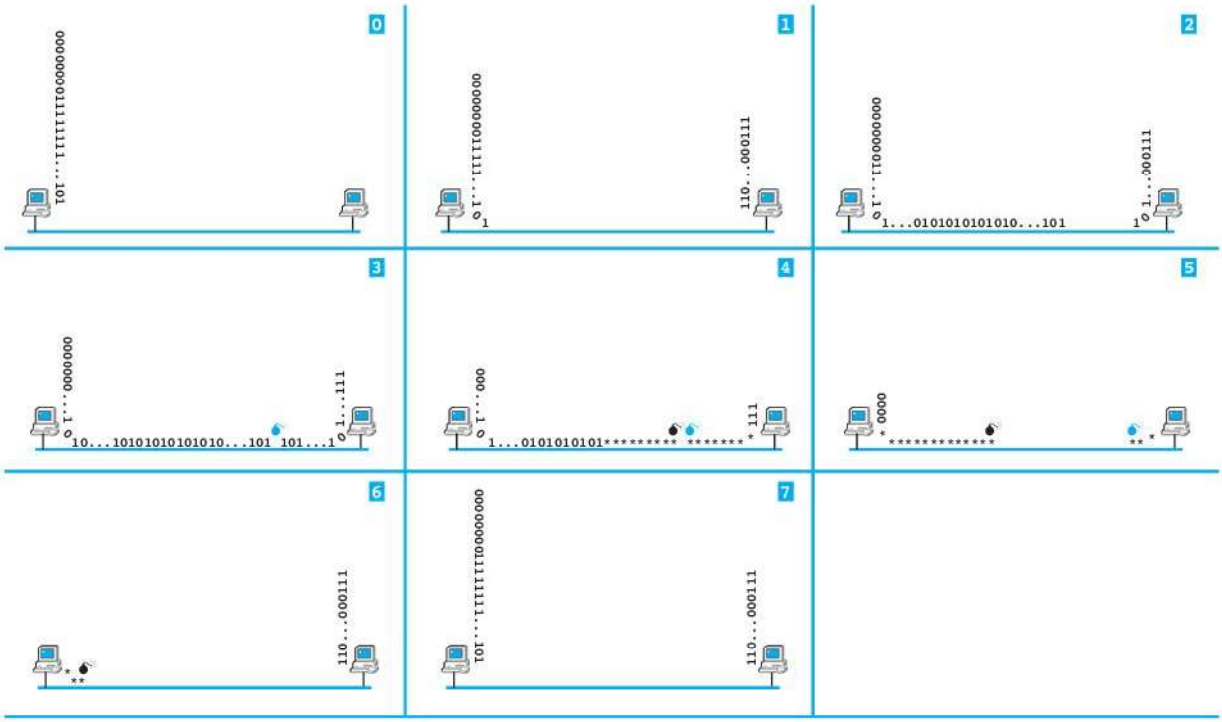
- sospende la trasmissione del pacchetto in corso;
- trasmette una sequenza speciale detta di *jamming*, ovvero di interferenza trasmissiva. Ciò serve per enfatizzare l'avvenuta collisione in modo che anche le altre stazioni percepiscano l'evento. In effetti le altre stazioni riconoscendo la sequenza di jamming in coda al pacchetto (che in questo caso è profondamente corrotto dato che è la sovrapposizione casuale di più pacchetti), scartano tutta la ricezione in corso;
- ripete il tentativo di trasmissione dopo un tempo pseudocasuale calcolato con uno speciale algoritmo e per un numero di volte non superiore a 16. Il ritardo pseudocasuale è necessario per evitare che le stesse stazioni che hanno generato la collisione la causino ancora al secondo tentativo e ai successivi. L'algoritmo di backoff prende il nome di *Truncated Binary Exponential* (TBE).



Risulta altresì evidente che le collisioni non sono altro che l'espediente che consente di accedere correttamente al mezzo; si può sostenere che una certa quantità di banda in CSMA/CD serve appunto per gestire le collisioni e poter così arbitrare il canale.

Così formulato CSMA/CD è un protocollo di accesso con le seguenti caratteristiche:

- Non è deterministico, cioè non è possibile stabilire sistematicamente se e quando una trasmissione andrà a buon fine.
- Non è adatto per attività di tipo *burst*, cioè a raffiche consecutive e continue di trasmissioni, a causa dell'aumentare esponenziale delle collisioni.

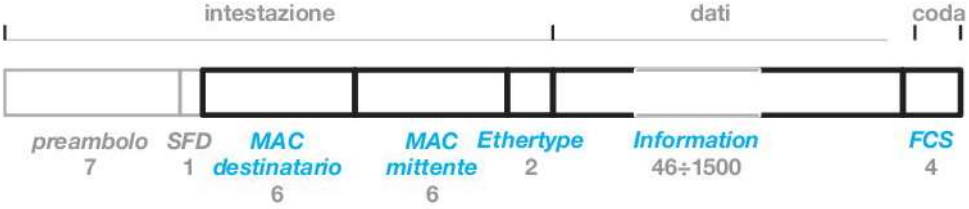


- Non è indicato per applicazioni in tempo reale (real time) dato che ha un tempo di ritrasmissione non predicibile.
- Per la semplicità di realizzazione consente di mantenere bassi i costi sia del cablaggio che dell'hardware di supporto, garantendo comunque una discreta efficienza.

2 Il pacchetto MAC

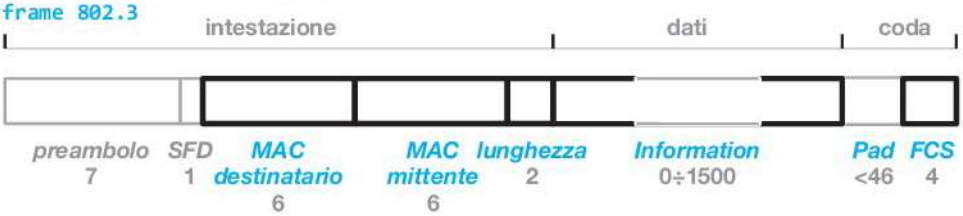
L'evoluzione storica di Ethernet in IEEE 802.3 ha causato la presenza contemporanea di due formati di pacchetto MAC: il più anziano, denominato **frame Ethernet 2** e il formato ufficiale **frame 802.3**:

`frame Ethernet 2`



Entrambi i pacchetti rispettano uno dei parametri progettuali di Ethernet, ovvero che il pacchetto più corto circolante in rete non sia inferiore a **64 ottetti** (byte), esclusi *preambolo* e *SFD*. In totale un frame Ethernet non può essere inferiore a 72 byte (576 bit).

`frame 802.3`



Per questo motivo, dato che la parte costante di entrambi i pacchetti vale 18 ottetti (6+6+2+4), se la parte *dati* è inferiore a 46 (64-18), il MAC la «riempie» automaticamente con tanti byte a zero fino a raggiungere 64 ottetti totali. Inoltre un pacchetto Ethernet (sia Ethernet 2 che 802.3) non può mai superare 1518 byte di ampiezza, altro limite progettuale del protocollo originale dovuto all'**MTU di Ethernet** che vale **1500 ottetti** per il campo *dati*.

Il *preambolo* sincronizza l'inizio del pacchetto. Questa serie di segnali serve per tarare i clock delle stazioni sulla linea.

Il campo *SFD* (*Start Frame Delimiter*) è un ulteriore delimitatore iniziale di pacchetto che contiene una sequenza di bit canonici che consentono all'hardware di sincronizzare i seguenti bit della trama.

Il campo *FCS* ha la funzione di garantire l'integrità del pacchetto. È implementato con CRC.

Si notano quindi i campi dedicati agli indirizzi fisici (MAC Address) che devono essere ampi 6 byte e univoci su scala mondiale: **indirizzo MAC del destinatario** e **indirizzo MAC del mittente**.

PROGRAMMA

Ricavare l'indirizzo MAC (Win32)

Scrivere un programma in linguaggio C per Windows che legge gli indirizzi MAC delle interfacce di rete installate sulla macchina:

LAYOUT

```
NIC 1, indirizzo MAC:   c0 cb 38 a9 fd a8
NIC 2, indirizzo MAC:   68 a3 c4 16 d8 ac
NIC 3, indirizzo MAC:   5c 26 0a 29 be 34
```

Per ottenere gli indirizzi MAC degli adattatori di rete installati, bisogna effettuare una system call al sistema operativo (API di Win32): **GetAdaptersInfo()**.

Affinché **Code::Blocks** possa compilare il seguente codice Win32 per Windows è sufficiente:

1) Creare un progetto con target **Console** (*File-New-Project-Console application*).
2) Collegare la libreria di sistema **libiphlpapi.a** (*Project-Build options-Linker settings-Add*). Basta digitare iphlpapi. La libreria serve per l'API GetAdaptersInfo().

Il codice in linguaggio C potrebbe essere il seguente:

```
00 #include <stdio.h>
01 #include <windows.h>
02 #include <iphlpapi.h> // per GetAdaptersInfo()
03
04 int main(void)
05 {
06   IP_ADAPTER_INFO AdapterInfo[16]; // per 16 MAC address
07   IP_ADAPTER_INFO* pAdapterInfo; // per maneggiare il buffer
08   DWORD dwBufLen;
09   DWORD dwStatus;
10   int i,j;
11   char szMac[80];
12
13   dwBufLen = sizeof(AdapterInfo);          // dimensione del buffer
14   // system call alla API Win32 appropriata
15   dwStatus = GetAdaptersInfo(
16               AdapterInfo,  // [out] conterrà i MAC address
17               &dwBufLen);   // [in] dimensione del buffer
18   if (dwStatus!=NO_ERROR) return 1;
19
20   j=0;
21   pAdapterInfo = AdapterInfo;       // puntatore al buffer
22
```

PROGRAMMA

```
23    while(pAdapterInfo)// se c'è un elemento...
24    {
25      j++;
26      sprintf(szMac,"NIC %d, indirizzo MAC: ",j);
27      for (i=0; i<6; i++)
28      {
29        sprintf(szMac+strlen(szMac)," %02x",pAdapterInfo->Address[i]);
30      }
31
32      printf("\n%s",szMac);
33      pAdapterInfo = pAdapterInfo->Next; // il prossimo...
34    }
35
36    return 0;
37  }
```

In evidenza nelle **righe 15÷17** la chiamata di sistema.

Il sistema operativo restituisce un vettore di strutture IP_ADAPTER_INFO (un elemento per ogni scheda NIC). Un campo della struttura (Address) contiene il vettore di 6 byte contenente l'indirizzo MAC.

Il primo campo della struttura (Next) contiene l'indirizzo del prossimo elemento nel vettore, così si può scorrerlo come una lista concatenata (**righe 23** e **33**).

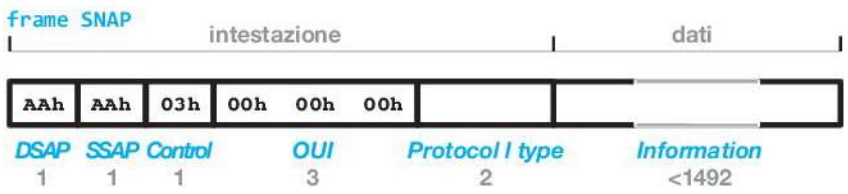
L'unica differenza tra i due formati di pacchetto MAC riguarda il campo *Ethertype/lunghezza* (anche detto **Type/lenght**).

Se questo campo contiene un valore superiore a 1500 (5dch), non si tratta di un pacchetto 802.3 e quindi deve essere un pacchetto Ethernet 2. In questo caso il valore significa il tipo di protocollo di livello superiore trasportato nel campo dati. Le codifiche per *Ethertype* iniziano da 0600h (1536), tra cui si ricordano:

Ethertype	Descrizione	Livello superiore trasportato
0800h	IP, Internet Protocol Version 4 (IPv4)	*3 Network (TCP/IP)*
0806h	Address Resolution Protocol (ARP)	*3 Network (TCP/IP)*
8035h	Reverse Address Resolution Protocol (RARP)	*3 Network (TCP/IP)*
809Bh	AppleTalk (Ethertalk)	
8100h	VLAN-tagged frame (IEEE 802.1Q)	*2, tra MAC e LLC*
814Ch	Simple Network Management Protocol (SNMP)	*3 Network (TCP/IP)*
86DDh	IP, Internet Protocol Version 6 (IPv6)	*3 Network (TCP/IP)*
8808h	MAC Control	*2, tra MAC e LLC*
8809h	LACP	*2, tra MAC e LLC*
8847h	Multi Protocol Label Switch (MPLS)	*3 Network*
8863h	PPPoE	*2, in tunnel*
8870h	Jumbo Frames	*2 MAC*

Quando invece il campo *Type/lenght* è inferiore o uguale a 1500, significa che si tratta della *lunghezza* del campo *dati* e allora la trama è 802.3.

In questo caso nel campo *dati* è spesso presente una LLC PDU denominata **SNAP Ethernet** (*SubNetwork Access Protocol*), con SAP=aah, che ha il seguente formato:



Quando l'OUI di SNAP vale 0, il campo *Protocol type* ha la stessa funzione di *Ethertype* e assume gli stessi valori codificati dalla tabella precedente.

In altri casi i SAP di LLC possono trasportare trame di altra natura, in base al valore assunto:

LLC SAP	Descrizione
AAh	SNAP
F0h	Netbeui/Netbios
E0h	IPX Novell
06h	IP, Internet Protocol Version 4 (IPv4)
42h	Spanning Tree Protocol (IEEE 802.1D)

Pur essendo posteriore e standardizzato da IEEE, il formato di pacchetto MAC 802.3 è più macchinoso da decodificare e sottrae 8 byte (1+1+1+3+2) all'MTU di Ethernet. Per queste ragioni nelle reti LAN circolano molti più pacchetti di tipo Ethernet 2 anche su LAN recenti, benché questi ultimi non prevedano neppure la presenza dello strato 2 LLC.

ESEMPIO

Frame Ethernet

Date le seguenti trame Ethernet in esadecimale, decodificarne il contenuto:

1) *00 ff fa 0e 65 21 00 11 43 51 fd 99 08 00 45 00 00 3d 5e d1 00 00 80 11 b8 16 c2 02 .. ..*

2) *00 0d 2b 11 f1 01 00 50 fc 23 4b 9c 00 81 F0 f0 da 3a 0e 00 ff ef 16 00 00 00 00 00 .. ..*

3) *00 00 c6 10 a1 31 00 50 fc 23 4b 9c 05 dc aa aa 03 00 00 00 08 00 45 00 00 3d 5e d1 .. ..*

Prima di tutto si controlla se il frame è di tipo Ethernet 2 o 802.3.

Bisogna considerare il tredicesimo e quattordicesimo byte della trama, cioè i primi due dopo gli indirizzi MAC destinatario (6 byte) e mittente (altri 6 byte): se il valore big endian è maggiore di 1500 (**5dch**) allora la trama è Ethernet 2, altrimenti è 802.3.

Nel primo caso basta considerare quei due byte come *Ethertype* e consultare la tabella per sapere che tipo di pacchetto il frame sta trasportando.

Nel secondo caso si individua il tipo di pacchetto consultando i due byte successivi (15mo e 16mo): se sono diversi da **aah**, allora il tipo di pacchetto trasportato è LLC; altrimenti si tratta di una trama SNAP e il tipo di pacchetto trasportato si individua analizzando i successivi quinto e sesto byte in big endian e si consulta la solita tabella (ma solo se l'OUI SNAP vale 0).

I casi proposti:

1) 00 ff fa 0e 65 21 00 11 43 51 fd 99 **08 00** 45 00 00 3d 5e d1 00 00 80 11 b8 16 c2 02 .. ..

13mo e 14mo byte: 08 00, che in big endian = **2048**d, maggiore di 1500 cioè trama Ethernet 2 con *Ethertype* = 0800h che indica pacchetto trasportato di tipo IPv4.

2) 00 0d 2b 11 f1 01 00 50 fc 23 4b 9c **00 81** f0 f0 da 3a 0e 00 ff ef 16 00 00 00 00 00 .. ..

13mo e 14mo byte: 00 81, che in big endian = **129**d, minore di 1500, cioè trama 802.3. Siccome il 15mo e 16mo byte non coincidono con aah aah, non è un pacchetto SNAP, quindi il pacchetto trasportato è un pacchetto LLC con *Protocol type* f0h (17mo byte) che, per inciso, è il *Protocol type* LLC del protocollo Netbios.

3) 00 00 c6 10 a1 31 00 50 fc 23 4b 9c **05 dc** aa aa 03 00 00 00 08 00 45 00 00 3d 5e d1 .. ..

13mo e 14mo byte: 05 dc, che in big endian = **1500**d, la trama è 802.3. I successivi due byte coincidono con **aah**, quindi siamo in presenza di una trama SNAP Ethernet con OUI = 0, come si deduce dai tre byte 00h 00h 00h. Allora i successivi due byte 08h e 00h coincidono con l'*Ethertype* che vale 0800h. Il pacchetto trasportato è quindi un IPv4.

PROGRAMMA

Frame Ethernet

Scrivere un programma in linguaggio C che legge da un file di testo la sequenza esadecimale di un frame Ethernet e ne stampi la natura: Ethernet 2, 802.3 SNAP o 802.3 LLC.

LAYOUT

```
frame Ethernet:00 0d 2b 11 f1 01 00 50 fc 23 4b 9c 00 81 f0 f0 da 3a 0e 00 ff ef 16 00 .. ..
Type/lenght: 0081h (129)
frame Ethernet 802.3 LLC, lunghezza: 129
```

Il codice in linguaggio C potrebbe essere il seguente:

```c
#include <stdio.h>
#include <string.h>  // per strcmp()

// prototipo della funzione
int htoi(char* strnum);

int main (void)
{
    FILE *fp; char szEthFrame[100];
    char szTypeLen[5]; int idx; int v;

    fp = fopen ("frameeth.txt", "r");
    if (fgets (szEthFrame, 100, fp) == NULL ) return 1;
    fclose (fp);
    printf("frame Ethernet:%s .. ..",szEthFrame);

    idx = 13; idx = (idx-1)*3;
    szTypeLen[0] = szEthFrame[idx]; szTypeLen[1] = szEthFrame[idx+1];
    idx = 14; idx = (idx-1)*3;
    szTypeLen[2] = szEthFrame[idx]; szTypeLen[3] = szEthFrame[idx+1]; szTypeLen[4]=0;

    v = htoi(szTypeLen);  // conversione in intero
    printf("\nType/lenght: %sh (%d)",szTypeLen, v);

    if (v>1500)
    {
      printf("\nframe Ethernet 2, Ethertype: %sh ",szTypeLen);
    }
    else
    {
      idx = 15; idx = (idx-1)*3;
      szTypeLen[0]=szEthFrame[idx]; szTypeLen[1]=szEthFrame[idx+1]; szTypeLen[2]=0;
      if (!strcmp(szTypeLen, "aa"))
      {
         printf("\nframe Ethernet 802.3 SNAP, lunghezza: %d ",v);
      }
      else
      {
         printf("\nframe Ethernet 802.3 LLC, lunghezza: %d ",v);
      }
    }

    return 0;
}

int htoi(char* strnum)
{
    int value = 0; char *p;

    p = strnum;
    while(*p )
    {
        if (*p >= '0' && *p <= '9') value = value * 16 + *p - '0';
        else value = value * 16 + *p - 'a' + 10;
        p++;
    }
    return value;
}
```

PROGRAMMA

Il file di testo **frameeth.txt** si trova nella cartella corrente e viene aperto, letto e chiuso con le **righe 11÷13**.

Il suo contenuto viene copiato nell'array di caratteri **szEthFrame[ ]**.

Il campo *Type/lenght*, in posizione 13 e 14, viene copiato in una stringa (**righe 16÷19**). Per raggiungerlo bisogna moltiplicare per 3 l'indice 13 e 14 (in una sequenza i byte sono spaziati e occupano tre caratteri).

Una volta trasformato in stringa esadecimale (su quattro caratteri), il campo *Type/lenght* va trasformato in numero decimale per poterlo confrontare con l'*MTU di Ethernet* (1500) per discriminare il tipo di pacchetto.

A tal scopo è stata scritta una funzione **htoi()** che, presa in ingresso una stringa esadecimale, restituisce il numero decimale equivalente. Il corpo della funzione si trova alle **righe 46÷57** e il prototipo a **riga 04**.

Infine, per sapere se il pacchetto è un *802.3 SNAP* o *802.3 LLC* si legge il byte a indice 15 e lo si confronta con la stringa esadecimale "aa" (**righe 30÷32**).

3 Dominio di broadcast

Gli indirizzi fisici delle LAN a 6 byte (MAC address) sono univoci su scala mondiale e memorizzati permanentemente nella ROM della scheda di rete dal produttore secondo lo schema OUI/n. di serie: i primi tre byte sono l'indentificativo univoco su base mondiale assegnato dall'IEEE ad ogni produttore, mentre il n. di serie è deciso autonomamente dal costruttore per individuare la singola scheda commercializzata.

PROGRAMMA

Ricavare l'indirizzo MAC (Java)

Scrivere un programma in linguaggio Java che legge gli indirizzi MAC delle interfacce di rete installate sulla macchina.

LAYOUT

```
NIC 1, indirizzo MAC:   c0 cb 38 a9 fd a8
NIC 2, indirizzo MAC:   68 a3 c4 16 d8 ac
NIC 3, indirizzo MAC:   5c 26 0a 29 be 34
```

Il codice in linguaggio Java potrebbe essere il seguente:

```
00 import java.net.NetworkInterface;
01 import java.net.SocketException;
02 import java.util.Collections;
03 import java.util.Enumeration;
04
05 public class main
06 {
07   public static void main(String[] args)
08   {
09     try
10     {
11       int j=0;
12       Enumeration<NetworkInterface> nets = NetworkInterface.getNetworkInterfaces();
13       for (NetworkInterface AdapterInfo : Collections.list(nets))
14       {
15
16         byte[] mac = AdapterInfo.getHardwareAddress();
17         if ((mac != null) && (mac.length==6))
18         {
19           StringBuilder sbMac = new StringBuilder();
20           j++;
21           sbMac.append(String.format("NIC: %d, indirizzo MAC: ", j));
22           for (int i = 0; i < 6; i++)
23           {
24             sbMac.append(String.format("%02x ", mac[i]));
25           }
26           System.out.println(sbMac.toString());
```

PROGRAMMA

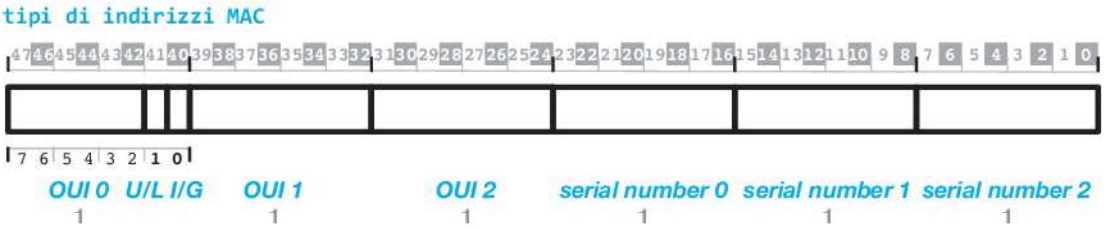
```
27          }
28        }
29      } catch (SocketException e)
30      {
31        System.out.println(e.getMessage());
32      }
33    }
34  }
```

In evidenza alla **riga 12** la chiamata di sistema, sottoforma di metodo statico, *NetworkInterface.getNetworkInterfaces()*.

Il metodo restituisce un vettore (un elemento per ogni scheda NIC) in cui ogni elemento è un oggetto di tipo *NetworkInterface* dal quale invocare il metodo *.getHardwareAddress()* (**riga 16**) che restituisce il vettore contenente i 6 byte dell'indirizzo MAC dell'interfaccia di rete.

Per scandire ogni elemento dell'enumerazione si usa l'interfaccia *Collection* (**riga 13**).

I due bit LSB del primo byte dell'OUI hanno un significato speciale e determinano il **tipo di pacchetto MAC**. Sono sottratti quindi alla codifica OUI, motivo per cui il primo byte di un OUI del produttore dovrebbe essere sempre pari e dovrebbe essere sempre pari il settimo byte di un pacchetto Ethernet:



Gli indirizzi MAC circolanti su Ethernet possono essere quindi di tre tipi:

- **single** (o unicast, I/G=0), se riferito ad un singolo sistema;
- **multicast** (I/G=1), se riferito ad un gruppo di sistemi; solo se la trama è multicast il bit U/L ha significato: 0, gruppo universale; 1, gruppo locale.
- **broadcast** (ff:ff:ff:ff:ff:ff, tutti i byte a ffh), se riferito a tutti i sistemi della LAN.

Quando una scheda LAN riceve un pacchetto effettua una serie di controlli. Prima verifica che il pacchetto sia integro (cioè abbia una FCS corretta) e di dimensioni ammesse. Poi analizza l'indirizzo MAC destinatario e:

- se è broadcast, il pacchetto viene sempre passato al livello superiore;
- se è single, il pacchetto viene passato al livello superiore solo se è uguale all'indirizzo MAC della scheda;
- se è multicast, verifica se il proprio MAC appartiene al gruppo indirizzato, nel qual caso lo passa al livello superiore.

Pacchetti con indirizzi broadcast e multicast *non trasportano mai dati utente* ma solo dati di servizio (esempio, servono per far funzionare il programma arp).

Ethernet big e little endian

L'ordine standard di trasmissione dei bit sul mezzo di Ethernet è bizzarro, benché sia stato poi adottato da tutte le altre tecnologie su LAN.

Rispetto ai byte del pacchetto adotta big endian: i byte vengono spediti da sinistra verso destra. Ma la spedizione dei singoli bit di un byte avviene in modo little endian, da destra verso sinistra, ovvero il primo bit che 'parte' è il bit di peso 0. È per questo motivo che i due bit I/G e U/L hanno peso 0 e 1, così da essere i primi ad essere spediti e quindi ricevuti.

Questa modalità è detta **canonical order**.

ESEMPIO

Tipi di indirizzi MAC

Date le seguenti trame Ethernet in esadecimale, decodificarne il tipo:

1) `aa 00 04 00 e9 7d 08 00 2b 18 1c 22 08 00 45 00 00 3e 3a ff 00 00 1e 11 56 23 82 c0 .. ..`

2) `01 00 5e 7f ff fa 00 80 d3 00 14 6d 08 00 46 00 00 20 58 dd 00 00 01 02 10 f4 c0 a8 .. ..`

3) `ff ff ff ff ff ff 00 11 43 51 fd 99 08 06 45 00 01 48 00 00 40 00 40 11 00 00 00 00 .. ..`

1) Siccome nel primo ottetto dell'indirizzo MAC destinatario (primo byte dell'OUI = aah = 10101010b), il primo bit, di peso zero (I/G), vale **0**, il pacchetto è **single** (unicast).

2) Siccome nel primo ottetto dell'indirizzo MAC destinatario (primo byte dell'OUI = 01h = 00000001b), il primo bit, di peso zero (I/G), vale **1**, il pacchetto è **multicast**. Siccome il bit di peso 1 (U/L) vale **0**, si tratta di un indirizzo multicast assegnato su scala mondiale. In effetti l'OUI **01:00:05** è un l'indirizzo Ethernet multicast assegnato dallo IANA a IPv4, cioè la trama conterrà un pacchetto IP.

3) Siccome l'indirizzo MAC del destinatario vale **ff:ff:ff:ff:ff:ff**, il pacchetto è **broadcast**.

Ora è possibile fornire una definizione di **dominio di broadcast** come quella sezione di rete LAN in cui si trovano tutti quegli host che, in presenza di un pacchetto di broadcast, lo ricevono.

PROGRAMMA

Tipi di indirizzi MAC

Scrivere un programma in linguaggio C che legge da un file di testo tre indirizzi MAC e ne determini il tipo: single, multicast o broadcast.

LAYOUT

```
Indirizzo MAC 0: aa 00 04 00 e9 7d, single
Indirizzo MAC 1: 01 00 5e 7f ff fa, multicast/universal
Indirizzo MAC 2: ff ff ff ff ff ff, broadcast
```

Il codice in linguaggio C potrebbe essere il seguente:

```
00 #include <stdio.h>
01 #include <string.h>
02 #include <stdlib.h> // per malloc()
03 #include "htoi.inc" // funzione htoi()
04
05 #define NUMMAC 3  // quanti indirizzi MAC si analizzano
06 #define DIMMAC 20 // n. caratteri per un indirizzo MAC
07 #define IGMASK (0x01)  //01h = 00000001b, maschera I/G
08 #define ULMASK (0x02)  //02h = 00000010b, maschera U/L
09
10 int main (void)
11 {
12   FILE *fp;
13   int i,j;
14   char *asz[NUMMAC]; // un elemento per indirizzo MAC
15   unsigned char bytMac;
16   char szByte[3]; char szMsg[80];
17
18   fp = fopen ("macaddr.txt", "r");
19   for (i = 0; i < NUMMAC; i++)
20   {
21     char *pc;
22     pc = (char *)malloc(DIMMAC);
23     if (fgets (pc, 100, fp) == NULL ) return 1;
24     *(pc+strlen(pc)-1) = 0;  // via il \n
25     asz[i] = pc;
26   }
27   fclose (fp);
```

PROGRAMMA

```
28
29   for (i = 0; i < NUMMAC; i++)
30   {
31     // il primo byte dell'indirizzo MAC
32     szByte[0]=asz[i][0]; szByte[1]=asz[i][1]; szByte[2]=0;
33     bytMac = (unsigned char) htoi(szByte);
34
35     if ((IGMASK & bytMac) == IGMASK)
36     {
37       int f = 1;
38
39       for (j=0;j<6;j++)
40       {
41         if (asz[i][j*3]!=asz[i][j*3+1])
42         {
43           f = 0;
44           break;
45         }
46       }
47
48       if ((f) && (asz[i][0]=='f'))
49       {
50         strcpy(szMsg,"broadcast");
51       }
52       else
53       {
54         strcpy(szMsg,"multicast");
55         if ((ULMASK & bytMac) == ULMASK)
56           strcat(szMsg," local"); else strcat(szMsg,"/universal");
57       }
58     }
59     else
60     {
61       strcpy(szMsg,"single");
62     }
63
64     printf("\nIndirizzo MAC %d: %s, %s",i, asz[i], szMsg);
65
66   }
67
68   return 0;
69 }
```

Il file di testo **macaddr.txt** deve avere tre righe non più lunghe di 18 caratteri (un indirizzo MAC in Ascii), si trova nella cartella corrente e viene aperto, letto e chiuso con le **righe 18÷27**.

Il suo contenuto viene copiato nel vettore di stringhe **asz [ ]**. Siccome fgets() copia anche un ritorno a capo alla fine della stringa, viene eliminato (**riga 24**).

Quindi per ogni indirizzo MAC (asz[i]), si isola il primo byte Ascii e lo si trasforma in decimale (**righe 32** e **33**).

Ora si possono verificare i due bit di peso 0 e 1, con le relative maschere (riga 35 e 55).

Per determinare se l'indirizzo è broadcast, si controlla che tutte le 6 coppie Ascii siano uguali (**righe 39÷46**), nel qual caso si controlla che un carattere esadecimale coincida con il codice Ascii 'f' (**righe 48÷50**).

NB. *La funzione htoi() è stata inclusa a* ***riga 03****. Il suo codice è presente nel* **PROGRAMMA. Frame Ethernet**.

4 Dominio di collisione

La prima realizzazione commerciale di Ethernet, 10BASE5 su cavo coassiale, ha stabilito alcuni valori base per il CSMA/CD che hanno poi influenzato i sistemi successivi fino ai più recenti. È quindi interessante analizzare le ragioni per cui le scelte di allora hanno generato conseguenze tutt'oggi vincolanti anche per le tecnologie più recenti, per esempio la lunghezza massima dei cavi in rame per le reti 802.3.

Jumbo frame

L'MTU di Ethernet (1500 byte) a volte può rallentare le operazioni di inoltro in rete, per esempio se il protocollo di livello superiore trasportato dalla trama ha un MTU maggiore (esempio, IP). In questi casi il pacchetto di livello superiore di dimensione > 1500 byte deve essere frammentato in spedizione e ricompattato in ricezione, operazioni che sprecano tempo utile (overhead).

Se tutti i dispositivi Ethernet coinvolti sono abilitati a gestire un MTU maggiore di 1500 byte, allora possono essere veicolati in rete pacchetti Ethernet di dimensioni che possono anche arrivare a 9000 byte, i cosiddetti **Jumbo frame**.

I Jumbo frame sono supportati da alcune realizzazioni di Fast Ethernet e da Giga Ethernet in su come standard.

Dato il bitrate originale a 10 Mbit/s, si poneva la questione di quanto dovesse essere lunga una tratta coassiale affinché le collisioni potessero essere ben discriminate. Bisogna considerare che affinché una collisione sia ben determinabile è necessario che il primo bit trasmesso di un pacchetto rientri nella scheda prima che l'ultimo bit sia stato spedito. Solo in questo caso una scheda ha la certezza che un pacchetto sia stato trasmesso correttamente o meno: se i bit che rientrano coicidono ordinatamente con i bit in uscita, il pacchetto è stato spedito correttamente, altrimenti è avvenuta una collisione.

In altri termini, una scheda che trasmette deve avere il completo controllo «elettrico» del mezzo durante tutta la fase di trasmissione.

Considerando che i bit devono percorrere due volte la lunghezza del bus coassiale (andata e ritorno) e che la collisione più sfortunata può avvenire su un estremo (caso in cui la scheda che trasmette e la scheda che causa la collisione sono collocate agli antipodi del bus coassiale), rimane da stabilire che distanza può percorrere il primo bit trasmesso prima che sia trasmesso l'ultimo bit. In altre parole è necessario sapere quanto tempo impiega la spedizione di un pacchetto. Bisogna quindi decidere la dimensione del pacchetto più piccolo trasmissibile, dato che questo è il caso che concede meno tempo al primo bit per circolare sul bus.

Fu stabilito che il pacchetto più piccolo possibile fosse lungo 64 ottetti (byte), che sommati agli 8 ottetti di preambolo e SFD, in totale portano a **72 ottetti** *la dimensione del pacchetto minimo per Ethernet.*

Sapendo che il tempo di trasmissione di un bit (**bit time**) a 10 Mbit/s vale $1/(10 \cdot 10^6)$ secondi, è possibile calcolare il tempo di andata e ritorno del pacchetto minimo (**RTD**, *Round Trip Delay*) per una Ethernet 10BASE5:

$$\begin{aligned} \text{RTD}_{\text{10BASE5}} &= \text{n.ottetti} \times 8 \times \text{bit time} = 72 \times 8 \times 1/(10 \cdot 10^6)\ \text{s} = \\ &= 576 \times 0{,}0000001\ \text{s} = 0{,}0000576\ \text{s} \end{aligned}$$

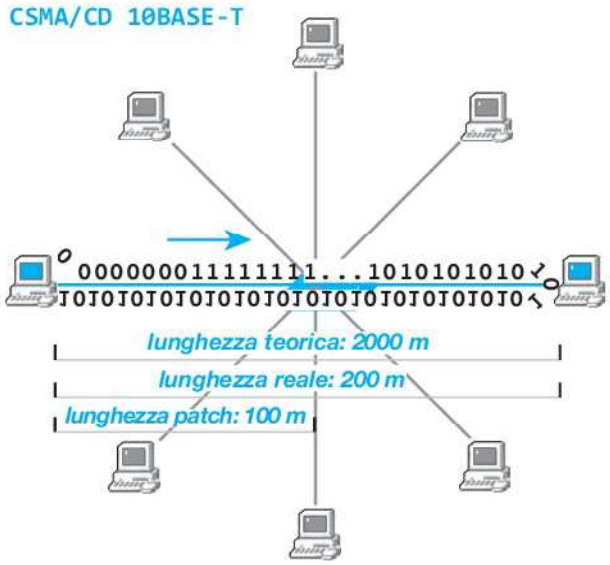
Sapendo che la velocità della luce c su un mezzo in rame viene attenuata di un fattore $k = 0{,}66$, si calcola lo spazio percorso nell'RTD:

$$\begin{aligned} \text{Spazio}(\text{RTD}_{\text{10BASE5}}) &= \text{velocità} \times \text{tempo} = c \cdot k \cdot \text{RTD}_{\text{10BASE5}} = \\ &= 300000\ \text{km/s} \cdot 0{,}66 \times 0{,}0000576\ \text{s} = 11{,}4\ \text{km} \end{aligned}$$

Il bit di un pacchetto di 72 ottetti su bus coassiale con bitrate a 10 Mbit/s percorre circa 11 km prima di rientrare nella scheda che l'ha trasmesso. Ne deriva che la lunghezza massima del cavo non può superare la metà di questa distanza (va divisa per andata e ritorno), cioè 11,4 km / 2 = 5,7 km.

Infine, siccome l'attenuazione del segnale su cavo coassiale vale un fattore 10, la massima tratta di bus ammissibile per rilevare correttamente le collisioni si riduce a circa 500 m.

Per 10BASE-T su doppio doppino con topologia a stella e con centrostella un ripetitore di segnale (*hub*), la massima distanza del mezzo si ha tra due stazioni ai vertici della stella. L'effetto combinato dell'hub (che connette le stazioni e rigenera il segnale) e la fisica del doppino ritorto consente teoricamente un diametro della stella pari a circa 2000 m, ottenibile con cavi di alta qualità. Al tempo la categoria di cavo UTP utilizzato era di categoria 3 o 4, insufficiente per queste distanze, e quindi il raggio massimo consentito per 10BASE-T si ridusse a 100 m.



In ogni caso la scelta di un pacchetto minimo di 72 byte (64+8 di preambolo e SFD) consente a 10BASE5/10BASE-T di ottenere un dimensionamento fisico soddisfacente per una LAN. Tale parametro rimarrà inalterato per molti anni.

Per 100BASE-T (Fast Ethernet), infatti, aumentando il bitrate di un fattore 10 e riducendo l'RTD a un decimo dell'originale, bisognava ridurre ad un decimo il diametro della stella, da 2000 m a 200 m. Con cavi di categoria 5 però si ottengono raggi per la stella di 100 m, pertanto la compatibilità con 10BASE-T rimase valida.

Per 1000BASE-TX (Giga Ethernet), l'RTD si riduce di un altro fattore 10 rispetto a 100BASE-T, pertanto per mantenere invariata la compatibilità per il limite di lunghezza del cavo a 100 m, bisognò aumentare di circa 10 volte la dimensione minima del pacchetto (e quindi dell'RTD), che in Giga Ethernet non vale più 64 byte, ma 512 byte.

Ora è possibile fornire la definizione di **dominio di collisione** come quella sezione di rete LAN in cui, se presente una collisione, tutti gli host su quella sezione di rete la rilevano.

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. Illustrare le nozioni di "half duplex", "broadcast" e "contesa" in riferimento al MAC di Ethernet.
2. Mostrare con un esempio quando si genera una collisione su una rete LAN Ethernet.
3. Spiegare come si distinguono le trame 802.3 dalle trame Ethernet 2.
4. Commentare il significato del campo Ethertype.
5. Spiegare come lo schema degli indirizzi MAC garantisce che siano univoci su scala mondiale.
6. Elencare e commentare i tipi di indirizzi MAC.
7. Commentare l'acronimo RTD.
8. Definire la locuzione «dominio di collisione».
9. Tradurre letteralmente la sigla CSMA/CD e indicare il significato di ogni suo termine.
10. Spiegare per quale motivo un indirizzo broadcast non può mai essere l'indirizzo di un mittente.
11. Definire la locuzione «dominio di broadcast».
12. Disegnare su un foglio lo schema di una rete LAN Ethernet con 5 stazioni. Quindi mettere in evidenza il dominio di broadcast e il dominio di collisione.
13. Elencare le principali tecnologie Ethernet indicandone il bitrate.

Requisiti avanzati

1. Provare a calcolare quanti costruttori di NIC card Ethernet possono essere presenti a livello mondiale.
2. Spiegare perché dopo una collisione una scheda Ethernet non riprova immediatamente a trasmettere.
3. Spiegare per quale ragione una trama 802.3 in taluni casi deve essere «riempita» (pad).
4. Illustrare il pacchetto SNAP.
5. Elencare i tre formati di pacchetto circolanti su Ethernet e motivarne la presenza.
6. Spiegare come si capisce se un indirizzo MAC è multicast o unicast.
7. Commentare il fattore di attenuazione della velocità della luce su un mezzo in rame per quanto riguarda il calcolo dello spazio percorso da un bit.
8. Mostrare perché Giga Ethernet funziona regolarmente su tratti di cavo di lunghezza uguale a quelli utilizzati da Fast Ethernet.
9. Giustificare il fatto per cui il settimo byte di un frame Ethernet dovrebbe essere sempre pari.
10. Riportare la dimensione minima e massima di una trama Ethernet.
11. Spiegare cosa potrebbe succedere se una scheda NIC Ethernet terminasse di spedire prima di ricevere il primo bit trasmesso.
12. Riportare almeno un caso in cui è necessario servirsi di pacchetti di broadcast.
13. Dati due indirizzi MAC su due vettori da sei elementi, scrivere un frammento di codice C per verificare se i due indirizzi sono uguali.

ESERCIZI PER LA VERIFICA SCRITTA

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

Indirizzi MAC e trame Ethernet

1 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
ff ff ff ff ff ff 00 16 ce 6e 8b 24 08 06
00 01 08 00 06 04 00 01 00 16 ce 6e 8b 24
c0 a8 00 72 00 00 00 00 00 00 c0 a8 00 01
```

2 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
00 15 f2 40 76 ef 00 16 ce 6e 8b 24 08 00
45 00 00 30 a7 e3 40 00 80 06 d0 60 c0 a8
00 72 c0 a8 00 c1 04 71 00 15 df b3 b2 fe
00 00 00 00 70 02 40 00 29 63 00 00 02 04
05 b4 01 01 04 02
```

3 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
00 05 5d 21 99 4c 00 03 ff 2a 45 d2 08 00
45 00 00 30 09 30 40 00 80 06 4d e7 c0 a8
00 b7 40 e9 a1 68 04 65 00 50 31 1e 0a c3
00 00 00 00 70 02 ff ff 9e d8 00 00 02 04
05 b4 01 01 04 02
```

4 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
00 a0 cc 3b bf fa 00 00 c0 9f a0 97 08 00
45 10 00 36 5a ac 00 00 40 06 9e b2 c0 a8
00 01 c0 a8 00 02 00 17 06 0e 17 f1 65 fe
99 c5 a1 c4 80 18 43 e0 fd 9e 00 00 01 01
08 0a 00 25 a6 36 00 9c 29 2a 24 20
```

5 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 00 0c cc cc cc 00 e0 1e d5 d5 15 01 1e
aa aa 03 00 00 0c 20 00 01 b4 df f0 00 01
00 06 52 31 00 02 00 11 00 00 00 01 .. ..
```

6 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 00 0c cc cc cc 00 13 c4 12 0f 0d 00 46
aa aa 03 00 00 0c 01 04 01 00 00 13 c4 12
0f 00 02 80 00 00 00 15 00 0d 00 01 .. ..
```

7 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 00 0c cc cc cc 00 13 c4 12 0f 0d 01 69
06 06 00 00 00 0c 20 00 02 b4 ca 1d 00 01
00 06 53 31 00 05 00 c2 43 69 73 63 .. ..
```

8 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 80 c2 00 00 00 00 19 06 ea b8 8c 00 27
42 42 03 00 00 02 02 0e 80 01 00 19 06 ea
b8 80 00 00 00 00 80 01 00 19 06 ea b8 80
80 0c 00 00 14 00 02 00 0f 00 00 00 00 00
00 00 00 00
```

9 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 00 0c cc cc cc c2 01 73 fe 00 00 01 50
aa aa 03 00 00 0c 20 00 02 b4 ea d3 00 01
00 06 52 31 00 05 00 fb 43 69 73 63 .. ..
```

10 Data la seguente sequenza di byte (esadecimali) che rappresentano un frame Ethernet, decodificarne il contenuto.

```
01 00 0c c1 c1 c3 00 e0 1e d5 d5 15 02 0e
e0 e0 04 01 01 02 50 0a 03 00 d1 ff 01 05
04 26 f2 f1 00 0e 00 31 00 00 00 01 .. ..
```

11 Dato il frame dell'Esercizio 1, classificare i tipi di indirizzi MAC del pacchetto.

12 Dato il frame dell'Esercizio 2, classificare i tipi di indirizzi MAC del pacchetto.

13 Dato il frame dell'Esercizio 3, classificare i tipi di indirizzi MAC del pacchetto.

14 Dato il frame dell'Esercizio 8, classificare i tipi di indirizzi MAC del pacchetto.

15 Dato il frame dell'Esercizio 9, classificare i tipi di indirizzi MAC del pacchetto.

16 Dato il frame dell'Esercizio 10, classificare i tipi di indirizzi MAC del pacchetto.

ESERCIZI PER LA VERIFICA DI LABORATORIO

Per compilare gli esercizi proposti è stato usato l'ambiente gratuito multipiattaforma Windows/Linux ***Code::Blocks 10.5*** con **gcc 4.4.1** e ***Eclipse SDK 4.2.0*** con **Jdk 7** per **Java 1.7**.

I testi degli esercizi presentano uno o più **layout** di input/output richiesti; in questo modo sono indicate, a volte, alcune specifiche del programma come l'output, il controllo dell'input o i valori ammissibili.

Indirizzi MAC

1 Scrivere un programma che acquisisce in input un indirizzo MAC in esadecimale, effettuando il controllo dell'input.

Layout:

OUTPUT

```
MAC address byte(1): 0a
MAC address byte(2): ag
MAC address byte(2): af
MAC address byte(3): 98
MAC address byte(4): 11
MAC address byte(5): 100
MAC address byte(5): 10
MAC address byte(6): 32

MAC address = 0a:af:98:11:10:32
```

2 Scrivere un programma che genera casualmente un indirizzo MAC corretto, quindi stabilirne il tipo.

Layout:

OUTPUT 1

```
MAC address casuale: 03:00:0c:cc:cc:cc
Tipo: Multicast, locale
```

OUTPUT 2

```
MAC address casuale: 00:01:2c:c1:c2:90
Tipo: Unicast (single)
```

3 Scrivere un programma che genera casualmente un indirizzo MAC, ma del tipo selezionato in input.

Layout:

OUTPUT

```
Tipo indirizzo MAC (b, m, u): m
MAC address Multicast: 01:f0:1c:aa:bb:13
```

4 Scrivere un programma che, rilevati gli indirizzi MAC del PC, indichi se qualcuno appartiene allo stesso produttore.

Layout:

OUTPUT

```
MAC address rilevati:
NIC 1, indirizzo MAC:  c0 cb 38 a9 fd a8
NIC 2, indirizzo MAC:  68 a3 c4 16 d8 ac
NIC 3, indirizzo MAC:  c0 cb 38 a9 fd 99
NIC 1 e NIC 2 stesso produttore
```

Trame Ethernet

5 Scrivere un programma che letto un frame Ethernet da disco, stampi l'indirizzo MAC destinazione e l'indirizzo MAC mittente.

Layout:

OUTPUT

```
Frame Ethernet su file frameeth.txt.
MAC address destinatario: 00 0d 2b 11 f1 01
MAC address mittente: 00 50 fc 23 4b 9c
```

6 Scrivere un programma che letto un frame da disco, stampi l'indirizzo MAC destinazione e l'indirizzo MAC mittente e ne determini il tipo.

Layout:

OUTPUT

```
Frame Ethernet su file frameethb.txt.
MAC address destinatario: ff ff ff ff ff ff
Tipo: Broadcast

MAC address mittente: 00 16 ce 6e 8b 24
Tipo: Unicast
```

7 Scrivere un programma che crea su disco un frame Ethernet di broadcast come quello dell'Esercizio 1, ma destinato a una interfaccia del proprio PC.

8 Scrivere un programma che letto un frame da disco, verifichi che la lunghezza del frame sia corretta (se possibile).

Nota: nelle sequenze di byte di un frame non viene mai riportato l'FCS, cioè mancano i 4 byte finali.

Layout:

OUTPUT 1

```
Frame Ethernet nel file frameeth.txt;
Formato Ethernet 2, verifica non possibile.
```

OUTPUT 2

```
Frame Ethernet nel file es_scritto_8.txt;
Formato 802.3.
Lunghezza campo Length: 39 (27h)
Padding: (46-39) = 7
Lunghezza del frame letto: 60
Verifica: 39+6+6+2+7 = 60   Ok
```