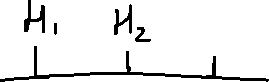
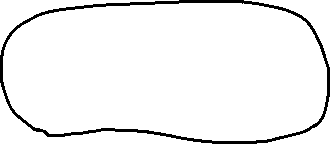
Oggi passiamo ad esaminare i livelli più bassi dei protocolli, il 2 e l’1 (Data Link e livello Fisico).  
Dal punto di vista storico sono stati sviluppati diversi protocolli a questi livelli, ma per il livello 2 al giorno d’oggi solo uno è sopravvissuto: il protocollo Ethernet, standardizzato nella collezione dell’ IEEE 802.3.   
Si tratta di un sistema abbastanza complicato, perché è nato tanti anni fa ed è stato sempre aggiornato senza mai andare a ridefinire le basi del protocollo stesso. Vediamo la prima versione di Ethernet, ormai non più in uso.

Nota: Stiamo parlando di reti locali, la cui copertura dal punto di vista fisico è di poche centinaia di metri.

La prima versione consisteva in una struttura di tipo BUS, che passava da una parte all’altra della rete fisica, quindi dell’edificio in cui si trovava, costituita da un cavo coassiale il cui diametro era di un centimetro abbondante, che serviva a connettere i vari dispositivi alla rete. Per connettere un dispositivo alla rete bisognava attaccare il dispositivo con un morsetto e poi bucare con una vite l’intercettatore interno. Questo permetteva di avere quante NIC connesse si voleva e quindi altrettanti Host.

IL problema di questo tipo di architettura era che c’era un solo cavo condiviso da tutti; quindi, i dispositivi si dovevano coordinare perché si poteva inviare un solo segnale alla volta: l’algoritmo di accesso al mezzo trasmissivo (che permetteva di coordinare i dispositivi) era chiamato MAC. Questo algoritmo derivava da un algoritmo più semplice usato nelle comunicazioni via radio. L’algoritmo, a differenza di altri protocolli, è decentralizzato: ogni host agisce da sé per implementare il meccanismo MAC (senza coordinarsi in modo esplicito con le altre NIC, la comunicazione è implicita).  
Il protocollo viene codificato dalla sigla CSMA/CD (Carrier Sense Multiple Access / Condition Detection). Chi vuole mandare un messaggio, prima di iniziare la trasmissione, ascolta cosa sta succedendo sul canale, in modo da capire se il canale è libero o occupato. Se il canale è occupato, aspetta che termini la comunicazione precedente, in modo da evitare le “collisioni” (il tentativo da parte di più host di mandare messaggi nello stesso momento: i messaggi diventano non più distinguibili e si generano errori), se il canale è libero, invece, si inizia a trasmettere.  
Quando un host comincia a trasmettere bisogna tenere conto della velocità di trasmissione dei segnali e della lunghezza dei cavi. Infatti nel momento in cui un host si accorge che il canale è occupato, in realtà il messaggio è già partito dall’host mittente da una certa quantità di tempo, in particolare, se due host fanno il controllo sulla libertà del canale SIMULTANEAMENTE, il carrier sense non nota messaggi nel canale ed entrambi immettono il proprio messaggio, con grandi probabilità che i due messaggi collidano. Più precisamente, esiste un intervallo iniziale di tempo, chiamato intervallo di vulnerabilità, dopo il quale è assicurata l’assenza delle collisioni, ma prima del quale può capitare che due host pensino che il sistema di trasmissione è libero quando non lo è.

Il Collision Detection è una complicazione a livello hardware che viene introdotta per riconoscere quando ci sono due o più segnali sovrapposti (quindi le NIC sono in grado di distinguere se il messaggio che ricevono è arrivato da una o più sorgenti, e quindi se c’è stata o meno una collisione).



Facciamo un esempio: immaginiamo che h2 voglia inviare un messaggio, ha aspettato l’intervallo di vulnerabilità e decide di effettuare l’invio. Mentre il messaggio di h2 viaggia, si svegliano h1 e h3 e vogliono mandare un messaggio a loro volta, ma poiché hanno il carrier sense aspettano che h2 abbia finito di trasmettere. Ma appena ha finito h2, iniziano entrambi in contemporanea a trasmettere il proprio messaggio, questo creerà inevitabilmente una collisione.  
H2 non può fare niente per fermare la collisione, i due mittenti (H1 e H3) se ne renderanno conto solo quando il messaggio arriverà da loro. Per dedurre che c’è stata una collisione (collision detection) ogni host è anche in modalità ricezione quando invia un messaggio e se gli torna indietro un messaggio diverso da quello che ha inviato allora c’è stata sicuramente una collisione. A questo punto si può solo interrompere la trasmissione, però prima il protocollo prevede di fare un’operazione di “jamming”. Il jamming vuol dire trasmettere un segnale ancora più forte di prima: serve ad assicurarsi che anche dall’altra parte si veda la presenza di una collisione (serve ad aumentare l’affidabilità del collision detection: basta che uno solo dei due host la noti per renderla nota anche all’altro).  
A questo punto però la parte iniziale del messaggio è andata persa a causa della collisione, e quindi si deve ricominciare da zero: si attiva il carrier sense e, quando si sente che non c’è più niente sul canale si può riprendere a trasmettere. Però qui sorge il problema che se entrambi gli host fanno la stessa cosa si ricapiterà in un’altra collisione (perché i loro carrier sense aspetteranno la stessa quantità di tempo): c’è bisogno di differenziare il loro comportamento, senza che i due host si possano mettere d’accordo tra loro. La soluzione adottata per questo problema è una di tipo probabilistico: si usa un generatore di numeri casuali per decidere se (50%) ritrasmettere immediatamente o (50%) aspettare prima di ritrasmettere. La speranza di uscire dalla collisione, in questo caso, deriva dal fatto che la probabilità che i due host facciano la stessa cosa è di 1/(2^n) dove n è il numero di tentativi fatti. Per accelerare un po’ questa situazione si riduce a ogni collisione la probabilità di invio immediato e si aumentano le possibilità da prendere in considerazione: alla prima collisione la scelta è tra inviare immediatamente o aspettare il tempo di vulnerabilità, dalla seconda collisione la scelta diventa tra inviare il messaggio subito o aspettare 1,2 o 3 volte il tempo di vulnerabilità, dalla terza le possibilità raddoppiano di nuovo e così succede dalla quarta, quinta ecc. (questo fino ad arrivare a dieci collisioni: il massimo che si può aspettare è 1024 volte l’intervallo di vulnerabilità).

Nota: nella prima versione la velocità di trasmissione era di 10 Mb/s (Mb = megabit a quanto pare?)

Proviamo a vedere come deve essere strutturata una NIC per supportare questo genere di protocollo. Sicuramente la NIC deve avere un buffer, in grado di contenere almeno un intero messaggio (in modo da poterlo ritrasmettere completamente in caso di collisioni). Chiola ha interrotto il discorso qui, yaaay.

Dal punto di vista di data link, i messaggi devono essere costituiti nel seguente modo. I primi 8 byte sono chiamati “preambolo”, questo serve a gestire l’asincronicità dei clock sulla rete locale: esso è costituito da 64 bit in cui sono alternati i valori 0 e 1, con due 1 consecutivi come terminazione, ciò permette di vedere qual è la reale frequenza di trasmissione del mittente.   
Quello che segue sono due indirizzi: l’indirizzo di destinazione e l’indirizzo mittente, questi sono costituiti da configurazioni di 6 byte. Questi indirizzi, per differenziarli dagli altri che vediamo all’interno degli altri protocolli, vengono chiamati “indirizzi MAC” (abbiamo un overloading del termine MAC: vuol dire dispositivo ad accesso condiviso, ma anche indirizzo da 6 byte). Il motivo per cui si usano 6 bytes per l’indirizzo MAC è perché si è stabilita l’unicità di questi indirizzi a livello mondiale (e non solo a livello locale).  
Per fare ciò ci si affida al costruttore del dispositivo: l’indirizzo del dispositivo è diviso in due parti da 3 byte ciascuno, i primi 3 indicano il costruttore, mentre gli altri 3 indicano l’indirizzo effettivo del dispositivo costruito dal costruttore. Quindi ogni costruttore avrebbe come limite 2^24 indirizzi MAC assegnabili, se non fosse che nel caso li usi tutti può sempre chiedere che gli sia assegnato un altro identificatore da 3 byte.  
Quello che viene dopo l’indirizzo sorgente è un campo chiamato “Tipo” che indica il protocollo di livello 3 che verrà utilizzato (serve a chi riceve il messaggio per sapere a quale protocollo passare il contenuto del payload): se al livello 2 e 1 abbiamo praticamente soltanto ethernet al livello 3 abbiamo quasi solo IP (c’è anche ICMP ma poco cambia).   
Dopodiché abbiamo il payload di dimensione variabili e poi abbiamo una particolarità del protocollo di livello 2 che è l’aggiunta di un “footer” dopo il payload, di dimensione 4 byte, che serve a fare una verifica di integrità tramite l’algoritmo CRC32 -> i messaggi che non superano questa verifica di integrità vengono scartati: anche in questo caso il protocollo è di tipo best effort, perché a seguito della collisione ritrasmette (o meglio, manda il jamming e si aspetta che ci sarà una ritrasmissione).

La dimensione dei messaggi va da un minimo che comprende soltanto l’header e il CRC32 a una dimensione massima, compresi header e footer, di 1500 bytes (che poi sono quelli che causano la frammentazione a livello di network -> l’MTU).

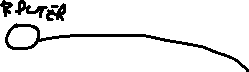
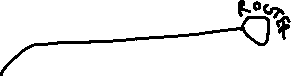
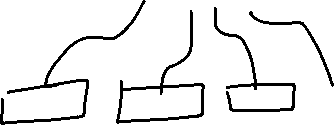
L’indirizzo mittente sarà quello della scheda che ha inviato il messaggio, mentre l’indirizzo destinazione sarà quello della scheda che lo deve ricevere: si noti però che per via del meccanismo del carrier sense tutte le macchine sono sempre in ascolto contemporaneamente, quindi, il modo di funzionamento effettivo del sistema di comunicazione sarebbe quello di broadcast, in cui si mandano messaggi a tutte le macchine. Però, il singolo host è in grado di filtrare i messaggi, confrontandosi con l’indirizzo di destinazione, e decidere se è il caso di passarli ai livelli successivi (protocollo IP, TCP/UDP ecc). Non è da eliminare però la possibilità di effettuare effettivamente un broadcast: è stato introdotto infatti un indirizzo MAC particolare - non associabile a nessuna macchina - chiamato “indirizzo broadcast”, che quando viene riconosciuto dalle NIC nel campo “indirizzo di destinazione” fa sì che accettino sempre il messaggio e lo passino ai protocolli di livello successivo. Tale indirizzo di broadcast è composto da soli 1 (quindi 6 byte = 48 bit tutti impostati a 1).

Problemi di questa versione iniziale di ethernet? La bassa affidabilità/tolleranza ai guasti: innanzitutto l’aggiunta di una connessione fisica sul caso lo modifica e indebolisce (si sta letteralmente piantando una vite nel cavo), può anche capitare che venga interrotta la connettività del cavo, causando la partizione della rete. Oppure se una delle NIC ha un malfunzionamento e inizia a spammare un segnale di jamming quando non ci sono collisioni rende l’intera rete inutilizzabile. Inoltre, per determinare un problema sul cavo, questo va ispezionato tutto a mano.

La versione successiva del protocollo ethernet prevedeva l’utilizzo di cavi più sottili e che permettevano connessioni utilizzando dei “connettori a T”, realizzati apposta. Rimaneva però il problema della partizione e del guasto di una NIC causando uno spam di jamming.

La terza versione ha previsto un cambio completo della tipologia della rete: il BUS è stato eliminato e si è passati a una struttura di interconnessione che prevede un apparato centrale che funziona da centro di raccolta e smistamento (con comunicazione peer to peer tra host e questo apparato centrale), usando all’inizio come apparato trasmissivo il cavo telefonico. Questo nuovo modello imponeva che la lunghezza massima dei singoli cavi che collegavano gli host all’apparato centrale fosse di soli 100 metri, tuttavia se da una parte si riduceva a una riduzione della dimensione della rete, dall’altra si poteva usare un clock a frequenza maggiore, ottenendo una banda di connessione a 100Mb/s.  
Il dispositivo centrale veniva chiamato “Hub” e fungeva da ripetitore e amplificatore dei segnali elettrici: viene simulato un canale di broadcast, ma le accortezze tecnologiche permettono una banda 10 volte superiore rispetto a quella della versione precedente.  
Con evoluzioni successive, il caco elettrico è stato rimpiazzato da connessioni in fibra ottica: ciò permetteva di superare il kilometro di lunghezza ai singoli cavi (e dunque alla distanza tra host e hub), seppur a un costo di realizzazione maggiore.

Con questa situazione, è sempre la NIC dei singoli host che si occupa di verificare l’assenza di segnali sul canale tramite il carrier sense, è sempre quella che si occupa di decidere l’intervallo di vulnerabilità (basandosi sulla distanza tra host). La tolleranza ai guasti è migliore rispetto alle versioni precedenti, perché anche se una NIC si rompe le altre porte di comunicazione (quelle che non portano a quella NIC) possono, in teoria, essere ancora utilizzate.



A livello di Data Link non è cambiato nulla (giusto la banda di trasmissione), è migliorata però la tolleranza ai guasti -> se si taglia un cavo si disconnette soltanto un host anziché partizionare l’intera rete.

L’evoluzione successiva ha previsto la sostituzione dell’Hub con un dispositivo un po’ più complicato, chiamato Switch. Se l’hub è un simulatore di BUS, lo Switch è un apparato che effettua il recapito di messaggi con una tecnica di tipo Store & Forward. Nello Switch possiamo avere quindi un certo numero di porte di comunicazione connesse (tramite cavo telefonico o fibra ottica) con degli host; in questo caso il cavo deve essere doppio (ci devono essere due fili intrecciati) in modo da effettuare operazioni di tipo “duplex”, trasmissione e ricezione devono poter avvenire contemporaneamente. Quando si manda un messaggio di tipo ethernet, questo viene memorizzato all’interno dello switch, che cercherà l’indirizzo MAC del destinatario e invierà verso di questo una copia del messaggio (il messaggio non è più inviato sempre in broadcast, ma su una sola porta di uscita in funzione dell’indirizzo di destinazione. Rimane la possibilità di fare un broadcast con l’indirizzo MAC con tutti 1). Lo Switch avrà quindi almeno un buffer di ricezione e almeno uno di trasmissione.

Se abbiamo una struttura di tipo Switch non c’è più alcuna possibilità di collisione, i canali di comunicazione sono di tipo full duplex (quindi messaggi in trasmissione e ricezione viaggiano su canali diversi) e la comunicazione non è più tra host e host, ma tra host e switch e tra switch e host.   
La gestione di più messaggi contemporaneamente dipende dalla qualità (quindi dal costo) dello Switch: Switch più dispendiosi saranno in grado di gestire più messaggi prima di saturare i buffer. Inoltre, l’impossibilità di collisione fa sì che la velocità della banda di 100 (ormai 200) Mb/s non sia più condivisa tra gli host, ma che ogni host goda di quella velocità di trasmissione a piena potenza: se ci sono 4 host collegati la banda complessiva sarà di 100 (o 200) \* 4 Mb/s.

Il prezzo che si paga per questo aumento della banda è (oltre all’investimento iniziale dei cavi) un aumento della latenza considerevole: per ottenere questa banda abbiamo dovuto introdurre il meccanismo dello store & forward nello switch, dunque, la latenza di questa rete sarà doppia rispetto a una equivalente con un Hub al posto dello Switch.  
Reti ad alta velocità hanno una grande banda, ma hanno anche una maggiore latenza: dunque se queste sono ottimali per l’invio di messaggi di grandi dimensioni, per messaggi di piccole dimensioni può convenire di più usare reti a bassa velocità, che hanno quindi una latenza inferiore.