

Sesta dispensa: la rete

[(c)2002 – Jean François Panico]

Affinché internet possa rappresentare un sistema universale di comunicazione, permetta cioè di far comunicare qualunque macchina connessa a una delle sue reti con una qualsivoglia altra macchina connessa alla stessa o a un'altra rete, è necessario fornire ogni macchina di un nome unico a livello globale. Internet fornisce ogni sistema di un nome, che identifica il sistema stesso, di un indirizzo, che mi dice dove si trova il sistema, e di un cammino, che mi dice come raggiungere il sistema. Ogni macchina connessa a una rete è detta host, nella terminologia internet. Lo stesso termine ha significati differenti in altri contesti informatici, come per esempio in quello client/server, o nel caso di mainframe. Attenzione a non fare confusione quindi. In internet un host può essere anche un vecchio 8088 con 640K di RAM e 10M di disco fisso.

Gli indirizzi IP

L'indirizzo, o IP address, è un campo composto da 32 bit. I primi bit permettono di distinguere 5 forme standard identificate da una lettera del alfabeto, e dette classi. Le prime tre classi dell'IP address contengono sia l'indirizzo di una rete (netid), sia quello di una macchina nella stessa (hostid). In realtà l'indirizzo non identifica necessariamente una macchina, ma una connessione alla rete. Per esempio, un router ha almeno due indirizzi, avendo connessioni ad almeno due reti. Questo in quanto un router appartiene a entrambe le reti, e quindi sono necessari due indirizzi dato che un IP address ha posto per un solo indirizzo di rete. Se l'indirizzo dell'host è 0, allora l'IP address si riferisce alla rete stessa. Se viceversa tutti i bit riservati all'indirizzo dell'host sono 1, allora l'indirizzo viene utilizzato per identificare tutti gli host della rete (broadcasting). Uno speciale indirizzo formato da 32 bit posti a uno è chiamato local network broadcast address e serve solo in casi molto particolari. Il concetto di broadcasting è quello della diffusione a tutto raggio, un po' come fa un'emittente radiofonica. In generale internet interpreta i campi formati da tutti uno come all, cioè "tutti", mentre quelli formati da tutti zero come this, cioè "questo", "qui". Questo per quanto riguarda le classi A, B e C. La classe D è usata per un particolare tipo di distribuzione dei dati detto multicasting. La classe E è riservata a usi futuri. Dato che specificare ogni singolo bit di un indirizzo IP sarebbe alquanto poco pratico e di scarsa leggibilità, la convenzione è quella di leggere ogni ottetto, cioè ogni gruppo di 8 bit, come un intero, e di separare i quattro ottetti con un punto. Oltre a i casi speciali già descritti, l'indirizzo di classe A 127.0.0.0 è riservato per un particolare processo di test che rimanda indietro i dati al mittente senza propagarli nella rete.

Uno dei vantaggi di questo schema è la possibilità da parte dell'organismo centrale che assegna gli indirizzi (Network Information Center) di delegare ai responsabili delle singole reti l'assegnazione di una parte dell'indirizzo all'interno della rete stessa. La cosa avviene un poco come con i numeri di telefono. A livello internazionale ogni stato ha il suo prefisso internazionale. Per esempio, per l'Italia, è 39. All'interno ogni stato divide il paese in aree geografiche a cui assegna un ulteriore codice. Per esempio, Roma è identificata dal 6, Milano dal 2, Firenze da 55, e così via. All'interno poi della provincia o della città possono essere definite ulteriormente sottoaree a cui si assegnano due, tre o quattro cifre. Per esempio 529 oppure 7054. Infine ogni telefono in tali aree avrà il suo numero. Così, se Mr. Smith deve chiamare dagli Stati Uniti il signor Mario Rossi abitante all'EUR, a Roma, comporrà per esempio il numero 011.39.6.529.4467. In questo caso lo 011 serve per uscire dagli USA, un po' come il nostro 00.

Analogamente in internet i numeri di classe C sono assegnati alle piccole reti, quelle cioè con meno di 256 host, quelli di classe B alle reti con al massimo 65536 host, e quelli di classe A alle reti con oltre 16 milioni di host. Ogni rete decide poi come suddividere gli indirizzi che gli sono stati riservati al suo interno come meglio crede. Ovviamente, una internet privata non ha la necessità di seguire queste regole, né a utilizzare indirizzi assegnati dal NIC, ma il non farlo potrebbe impedire in futuro la connessione alla TCP/IP Internet.

Dato che l'indirizzo può essere a volte abbastanza ostico da ricordare, è possibile associare a ogni host anche un nome, che può essere utilizzato come mnemonico per un IP address, e la cui risoluzione è responsabilità di particolari macchine chiamate name server. In realtà il name server è un programma software che può girare in qualunque macchina connessa alla rete, e che mantiene l'associazione tra nomi e indirizzi IP, fornendo tali corrispondenze quando richiesto da un altro programma chiamato name resolver. Di fatto, si preferisce far girare il name server su una macchina dedicata, che prende anch'essa, a questo punto, il nome di name server. Potete pensare al name server come a una agenda telefonica

elettronica, che contiene una lista parziale di nomi e numeri telefonici. In internet infatti, non esiste un singolo elenco telefonico, ma tanti name server che cooperano per fornire quello che è un vero e proprio elenco distribuito. In realtà il sistema funziona in modo gerarchico, un po' come se una certa agenda contenesse solo i prefissi internazionali e il puntatore alle agende di ogni singolo stato, le quali a loro volta contengono i prefissi regionali e i puntatori agli elenchi regionali, e così via, fino ad arrivare all'agenda che contiene solo le estensioni telefoniche di un singolo edificio.

Il Domain Name System

I nomi Internet sono basati su una serie di regole dette Domain Name System (DNS), che si basa appunto su uno schema gerarchico in cui il nome è suddiviso in varie parti separate fra loro da punti. Per esempio, vnet.ibm.com. Ogni suffisso è a sua volta un dominio. Quindi, nel nostro esempio, ibm.com è un dominio di secondo livello, mentre com è un dominio di terzo livello. I domini ufficiali riconosciuti dal NIC al livello più elevato sono riportati in tabella 1. Una rete può richiedere di essere registrata come categoria, oppure usando il dominio geografico. Per esempio, l'Italia ha come dominio base it. Supponiamo che il governo decida di costruire un insieme di reti cittadine interconnesse fra loro e connesse a Internet. Si può pensare di assegnare a ogni provincia un dominio xxxxxx.it. Per esempio, Firenze avrebbe come dominio firenze.it. L'università di Firenze potrebbe registrare la sue rete come unifi.edu, e in tal caso sarebbe direttamente il NIC a dover dare l'autorizzazione per tale nome, essendo il dominio edu sotto responsabilità dell'organismo centrale di controllo, oppure potrebbe decidere di far parte del dominio cittadino, come unifi.firenze.it, e quindi potrebbe richiedere il permesso di registrare tale nome direttamente all'amministratore del dominio di Firenze. A questo punto, se il dipartimento di Fisica di Arcetri vuole registrare un proprio dominio, deve chiederlo solo all'Università stessa, ricevendo così, per esempio, arcetri.usf.fi.it oppure fisica.usf.fi.it.

Esiste una piccola complicazione. Ogni oggetto connesso alla rete ha un tipo. Oggetti di tipo diverso possono avere lo stesso nome. Per cui, per poter risolvere un nome e ottenere indietro l'indirizzo IP, è necessario anche specificare il tipo di oggetto: macchina, utente, casella postale, e via dicendo. Dal solo nome non è possibile evincere il tipo di oggetto.

Il DNS definisce anche come associare i nomi agli indirizzi IP, e come ottenere quest'ultimi dal nome. In realtà lo schema è ancora più generale di quanto può sembrare, in quanto permette di estendere la sintassi del nome per usi specifici, sfruttando anche il concetto di tipo. Per esempio, nel caso di una casella postale (tipo MX), il nome sarà del tipo utente@dominio.

Per esempio ddejudicibus@tecnet.it

Innanzitutto una internet è un sistema di interconnessione fra reti differenti che utilizza sia sistemi dedicati per la connessione, detti gateway, sia uno strato (layer) di protocolli che mostrano alle applicazioni una visione omogenea di una rete virtuale e che sono basati sulla trasmissione di piccoli pacchetti di dati. Ogni pacchetto porta con sé l'indirizzo del destinatario il quale identifica univocamente sia la rete di destinazione che la connessione alla quale deve essere recapitato il pacchetto. Un sistema connesso a più reti della stessa internet avrà quindi più indirizzi IP. Un opportuno software, spesso installato su macchine dedicate, permette di associare a ogni indirizzo un nome di più facile utilizzo da parte degli utenti del sistema. Il formato di questo nome si basa su un insieme di regole dette DNS. Quella che è universalmente conosciuta come Internet è di fatto la principale rete interconnessa esistente, che si estende praticamente su tutto il pianeta.

Data questa premessa, vediamo di approfondire la trattazione dei protocolli TCP/IP. Innanzitutto qualunque trasferimento di dati implica la trasmissione di bit da un sistema a un altro. Tali dati devono essere correttamente interpretati dai vari sistemi connessi alla rete. Data l'enorme varietà di hardware e di sistemi operativi questo è tutt'altro che banale. Nei protocolli di trasmissione i bit vengono convenzionalmente raggruppati per multipli di otto, detti ottetti. Una volta questo corrispondeva al bus da 8 bit, cioè un byte, tipico dei computer. Oggi la maggior parte dei computer usa parole di almeno 32 bit. Tuttavia non tutte le macchine memorizzano tali parole nello stesso modo. Esistono vari modi per memorizzare un intero rappresentato da 32 bit. In quello detto Little Endian, la posizione più bassa in memoria contiene il byte di ordine più basso dell'intero. Nei sistemi Big Endian avviene esattamente il contrario, cioè la posizione più bassa in memoria contiene il byte di ordine più elevato. In altri sistemi ancora il raggruppamento viene fatto con parole da 16 bit, in cui la parola meno significativa viene appunto prima. Il risultato è lo stesso del Little Endian ma con i byte invertiti all'interno di ogni singola parola. È evidente che non è pensabile che sia la rete a gestire tutti questi modi diversi di interpretare i dati, anche perché di solito i protocolli di trasmissione non entrano nel merito di come ragionano i singoli

sistemi, ma si occupano solamente di trasferire in modo più o meno affidabile i dati a loro affidati. Ne consegue la necessità di definire un formato standard valido per tutti i dati che corrono lungo i collegamenti, lasciando a i vari sistemi il compito di effettuare le opportune conversioni locali. Lo standard internet prevede che gli interi vengano trasmessi a partire dal byte più significativo, secondo lo stile del Big Endian.

Così in un pacchetto, un intero ha il byte più significativo verso la testa del pacchetto e quello meno significativo verso la coda dello stesso.

A questo punto i sistemi sono in grado di scambiarsi i dati in modo non equivoco. Ma come fa a sapere la rete internet che un sistema è collegato, e soprattutto, come avviene l'associazione tra l'IP address e l'indirizzo fisico di rete? Ogni rete fisica infatti ha un suo formato per gli indirizzi fisici assegnati alle connessioni di rete. In generale esistono due modi di assegnare indirizzi fisici alle macchine connesse in rete. In una rete piccola, come può essere una Token Ring, cioè un anello di un paio di centinaia di macchine al massimo, a ogni connessione può essere assegnato un intero basso, per esempio compreso tra 1 e 254. Questo sistema ha il vantaggio di associare l'indirizzo fisico alla connessione piuttosto che alla scheda che permette la stessa. Per cui, se la scheda si rompe, l'utente può cambiarla senza dover tuttavia modificare l'indirizzo fisico di rete, purché imposti sulla nuova scheda lo stesso indirizzo di quella vecchia. Lo svantaggio è che non esiste alcun controllo che impedisca a due utenti sulla stessa rete di impostare lo stesso indirizzo fisico, creando così una collisione. In altri tipi di reti, come per esempio Ethernet, ogni scheda ha già preimpostato da parte del costruttore un indirizzo fisico fisso, per cui non c'è alcun rischio di collisione, ma cambiare la scheda vuol dire dover necessariamente cambiare indirizzo fisico. Inoltre, dato che questo indirizzo è unico non solo fra le schede installate su una certa rete, ma in assoluto fra tutte le schede costruite, esso è generalmente molto lungo. Nel caso di Ethernet è di ben 48 bit.

Associare un IP address a un sistema con indirizzi formati da piccoli numeri e per giunta tali che a parità di connessione l'indirizzo non cambia mai, come nel caso di una rete proNET-10, è molto semplice. Per esempio, per un IP address di classe C, si può usare l'indirizzo fisico come host identifier. Così, se la rete ha IP address del tipo 10.214.32.x, l'host con indirizzo fisico 16 avrà IP address 10.214.32.16. Un altro paio di maniche è gestire indirizzi molto più lunghi dei 32 bit utilizzati per gli indirizzi internet, e per giunta che possono cambiare nel tempo a parità di connessione. Ovviamente si potrebbe tenere da qualche parte una tabella per gli accoppiamenti, e di fatto si fa così, ma non è certo molto pratico pensare che qualcuno la tenga aggiornata a mano. Il problema è stato risolto efficacemente utilizzando un meccanismo di risoluzione dinamica implementato dal protocollo ARP, o Address Resolution Protocol.

ARP funziona più o meno così. Quando un host deve spedire un pacchetto a un certo destinatario, spedisce a tutti gli host nella stessa rete fisica un messaggio in cui chiede chi è l'host con quel ben preciso IP address. Nello stesso messaggio mette anche i propri indirizzi, sia quello fisico che quello IP. Si adopera cioè una tecnica di broadcasting. L'host il cui IP è quello cercato, rimanda indietro al richiedente il proprio indirizzo fisico, permettendo così l'associazione tra i due. Ciò è possibile in quanto esso ha comunque ricevuto anche l'indirizzo fisico del mittente. Ma allora per ogni pacchetto che va spedito a un certo IP address è necessario prima mandare un pacchetto a tutti gli host nella rete? E perché allora non mandare direttamente il pacchetto da trasmettere a tutti, invece di chiedere prima chi è che ha un certo indirizzo IP? Ovviamente la cosa non funziona così, anche perché si rischierebbe di appesantire inutilmente la rete con pacchetti che vengono recapitati ai sistemi sbagliati. Quello che si fa è di mantenere presso ogni host una tabella con tutti gli accoppiamenti già trovati, e di aggiornarla periodicamente per evitare che diventi obsoleta. A questo punto i meccanismi di broadcasting servono ad aggiornare tali tabelle. Per esempio, se un host deve spedire un pacchetto a un certo indirizzo IP, prima controlla nella sua tabella se non ha già l'indirizzo fisico del destinatario. Solo nel caso l'informazioni manchi, l'host spedisce a tutti gli altri host il messaggio di richiesta. Quando questo arriva a un qualunque host, sia esso il vero destinatario o no, ogni host aggiorna la sua tabella con l'indirizzo fisico e quello IP del mittente, tanto per guadagnare tempo. Il destinatario, in più, spedisce indietro anche il suo indirizzo fisico al mittente, così da potergli permettere di aggiornare la sua tabella di indirizzi. Un'ulteriore tecnica che si usa per assicurarsi che tali tabelle siano sempre aggiornate, è quella di far distribuire la propria

coppia di indirizzi, fisico ed IP, ogni qual volta un sistema si connette alla rete, per esempio al reboot.

ARP non viene considerato propriamente un protocollo internet, quanto un meccanismo della rete fisica. Su ARP si basa il protocollo IP per far comunicare fra loro le varie macchine quando non è possibile risolvere in altro modo gli indirizzi IP in indirizzi fisici. Un protocollo analogo è il RARP, o Reverse Address

Resolution Protocol, con il quale una macchina senza disco fisso (diskless) è in grado di conoscere il proprio indirizzo IP a partire da quello fisico. Per far ciò la rete deve avere uno o più RARP Server, i quali contengono una tabella di associazione fra gli indirizzi IP e quelli fisici di tutte le macchine diskless. Anche questo protocollo si basa su un messaggio mandato in broadcasting. L'esistenza di questo protocollo è legata al fatto che una macchina diskless non può memorizzare il proprio indirizzo IP in alcun posto, non avendo memoria secondaria.

E veniamo ora al TCP/IP vero e proprio. Come detto prima l'architettura internet è basata su tre livelli. L'Application Services è il livello più alto, cioè quello delle applicazioni. I programmi che utilizzate quando usate internet ricadono in questo livello. Il Reliable Stream Transport Service è il livello intermedio. Esso si occupa dell'affidabilità della comunicazione, gestendo gli errori di trasmissione e la perdita di eventuali dati. Esso inoltre fornisce una visione della comunicazione ad alto livello, in cui esiste una connessione tra i due host che si trasmettono grandi volumi di dati. Il livello più basso, chiamato Connectionless Packet Delivery Service è quello che effettua la spedizione vera e propria dei singoli pacchetti, senza garantire l'affidabilità sulla singola trasmissione, nella modalità detta connectionless.

Il protocollo su cui si basa il livello più basso della torre internet è appunto l'Internet Protocol, o IP. Tale protocollo si basa su alcuni concetti fondamentali. Innanzi tutto il servizio che fornisce è detto unreliable, cioè inaffidabile, in quanto non dà alcun garanzia che il singolo pacchetto arrivi effettivamente a destinazione. In secondo luogo è detto connectionless, cioè senza connessione diretta, in quanto la trasmissione non avviene direttamente verso il destinatario, ma il messaggio è lanciato nella rete lasciando poi a questa il compito di portarlo a destinazione utilizzando l'indirizzo IP dell'host destinatario. Infine si parla di best-effort delivery, cioè spedizione al meglio delle possibilità, in quanto la rete fa tutto il possibile per portare comunque a destinazione il pacchetto. In pratica l'IP si comporta come un naufrago su un'isola deserta che lancia nella corrente un messaggio in una bottiglia per un tizio che si trova su di un'altra isola dello stesso arcipelago, contando sul fatto che se la bottiglia arriva sull'isola sbagliata qualcuno ributterà a mare il messaggio fintanto che non arriverà a destinazione. Detta così c'è quasi da stupirsi che internet funzioni così bene. Anzi, che funzioni del tutto! In realtà non dimentichiamoci che sopra al livello più basso ce n'è un altro che garantisce appunto l'affidabilità della comunicazione. Torniamo comunque all'IP. Esso è formato da tre regole base: come è fatto il pacchetto da trasmettere, detto IP datagram, come avviene la scelta del cammino che il pacchetto segue per raggiungere il destinatario, come gli host e i gateway devono trattare i pacchetti e in particolare le modalità per l'emissione dei messaggi di errore e quelle per la soppressione dei pacchetti.

Prima però di entrare nel dettaglio dei singoli campi, vediamo come si comporta l'IP nella gestione dei pacchetti di dati. Questo ci permetterà più avanti di comprendere meglio il significato di alcuni campi dell'IP datagram.

Innanzi tutto va ricordato che l'IP è un protocollo unreliable, non dà cioè alcuna garanzia che il singolo pacchetto arrivi effettivamente a destinazione, ed è connectionless, ovverosia il messaggio non viene spedito direttamente al destinatario ma viene immesso nella rete lasciando poi a questa il compito di portarlo a destinazione. Esso inoltre è di tipo best-effort delivery, in quanto la rete fa tutto il possibile per portare comunque a destinazione il pacchetto.

Detto questo, vediamo come avviene la trasmissione vera e propria dei dati. L'unità fisica di trasferimento dei dati in una rete è la frame. Questa è composta di due parti: l'intestazione (header) e l'area dati (data area). L'unità di misura è invece l'ottetto, formato da otto bit, cioè un byte. Ogni rete fisica ha un limite massimo di capacità di trasferimento per un singolo frame, detto Maximum Transfer Unit (MTU). L'MTU è cioè il massimo numero di ottetti di dati che può essere trasferito in un singolo frame. Per esempio, Ethernet ha generalmente una MTU di 1.500 ottetti (1492 secondo lo standard IEEE 802.3). Questo vuol dire che se si devono spedire 2.000 byte di dati via Ethernet, è necessario spezzarli in due blocchi tali che ogni blocco sia minore o uguale a 1.500. A ogni blocco si aggiunge poi l'intestazione del frame. Dal punto di vista della rete fisica l'IP datagram è un blocco di dati. La rete fisica ignora cioè come tali dati vengano utilizzati dall'IP. Quindi, il primo compito di IP è quello di decidere come costruire il datagram affinché possa essere trasmesso in un frame fisico. L'ideale sarebbe di poter mettere un singolo datagram in ogni frame, ottimizzando così la trasmissione e semplificando la logica. Ma quale frame? Quello della rete di partenza? Quello della rete di arrivo? E se durante la trasmissione il datagram deve passare attraverso più reti con MTU differenti? Il punto è che non c'è modo di fare una scelta che assicuri di avere un datagram per frame. D'altra parte internet ha come obiettivo quello di svincolarsi dalle caratteristiche fisiche delle varie reti interconnesse fra loro. E allora? La soluzione adottata è molto semplice. Le dimensioni del datagram sono scelte convenzionalmente secondo una logica del tutto indipendente dalle MTU delle singole reti fisiche, dopodiché, a seconda della rete in cui il

datagram deve passare, questo è spezzato in più pezzi di dimensioni inferiori alla MTU della rete fisica, detti frammenti (fragment).

Il datagram è anch'esso un frame, che potremmo chiamare logico per distinguerla da quello usata da una specifica rete fisica per trasmettere i dati. Come tale anch'esso è formato da una intestazione e da un'area dati. All'atto della frammentazione, ogni frammento viene costruito replicando l'header del datagram, modificandone alcuni campi che vedremo in seguito, e aggiungendo a questo un pezzo dell'area dati originaria. L'aspetto più importante di questo meccanismo è che il riassetto dei frammenti non viene effettuato quando i vari frammenti rientrano in una rete fisica ad alto MTU, ma sempre e comunque presso l'host di destinazione. Così, se due reti con MTU uguale a 1.500 ottetti sono separate da una rete con MTU più bassa, per esempio 500 ottetti, i frammenti che arriveranno a destinazione saranno di soli 500 ottetti. In questo caso la frammentazione avviene nel primo gateway mentre il riassetto avviene solo nell'host di destinazione. Il protocollo IP richiede che sia gli host che i gateway siano capaci di gestire datagram di almeno 576 ottetti. In aggiunta, questi ultimi devono essere capaci anche di gestire datagram grandi quanto l'MTU più grande tra quelle delle reti a cui sono connessi. Ricordiamo che un gateway, per definizione, ha almeno due connessioni e quindi almeno due indirizzi IP.

Il punto debole di questo meccanismo è che la perdita di anche un solo frammento comporta la perdita dell'intero datagram. Dato che ogni frammento è trasmesso indipendentemente, passare attraverso reti a bassa MTU comporta un'elevata frammentazione anche nelle reti a maggiore MTU e comunque aumenta i rischi di perdita dei dati. Quando un frammento arriva a destinazione, e non è detto che il primo arrivi per primo, l'host fa partire un timer. Se questo scade prima che tutti i frammenti siano arrivati, il sistema cancella tutti i frammenti e considera perduto il datagram. Il concetto di timer e di tempi è estremamente importante per l'IP ed è spesso usato per ottimizzare la rete. Per esempio, ogni datagram ha una scadenza. Se il datagram è ancora all'interno della rete quando il suo tempo è scaduto, esso viene cancellato definitivamente. Lo scopo è quello di evitare che un pacchetto possa restare all'infinito in internet a causa di un errore in una routing table. Queste tabelle infatti servono a gestire il processo di instradamento del pacchetto nella rete. Se una o più tabelle sono sbagliate, si potrebbero creare cammini chiusi in cui i datagram potrebbero rimanere intrappolati. Veniamo finalmente al formato del datagram. Come si è già detto esso è composto di un'intestazione e di un'area dati. L'area dati contiene semplicemente una parte dei dati da trasmettere. Questo in quanto il datagram è piccolo mentre l'oggetto da trasmettere può essere anche molte centinaia di Kilobyte, se non addirittura migliaia, come per esempio un'immagine o un file compresso. L'intestazione è invece alquanto più complessa. Vediamola in dettaglio.

I primi 4 bit contengono la versione del protocollo IP che è stato utilizzato per creare il datagram. Infatti, come spiegato nella prima parte di questo corso, il tutto funziona se e solo se tutti seguono le stesse regole alla lettera. D'altra parte le convenzioni, e di conseguenza i protocolli, seguono un processo di evoluzione, per cui un datagram creato con una versione più recente potrebbe creare problemi a un protocollo più vecchio se questi non avesse modo di accorgersene in tempo. I 4 bit successivi danno la lunghezza dell'intestazione misurata in parole da 32 bit. Questa è necessaria agli algoritmi usati per leggere il datagram (parsing algorithms). Dato che i campi dell'intestazione potrebbero non risultare un multiplo intero di 32, è necessario porre alla fine dell'intestazione un campo di riempimento. Inoltre il programma di ricezione ha bisogno di conoscere anche la lunghezza totale del datagram, cioè la lunghezza dell'intestazione più quella dell'area dati. Questa è memorizzata nei bit dal 16 al 31 inclusi, e il suo valore è espresso in ottetti, al contrario del precedente. Poiché il campo è lungo 16 bit, il datagram non può essere più grande di 216 ottetti, cioè 65.535 byte.

Se l'IP rappresenta il braccio del TCP/IP, il TCP ne rappresenta la mente. Il primo si limita a spedire rapidamente i dati che gli arrivano senza preoccuparsi troppo se qualcosa va male. Il secondo si occupa invece di controllare che l'informazione passatagli dai livelli superiori arrivi correttamente a destinazione. Insieme sono sicuramente una coppia molto affiatata.

In questo articolo useremo il termine applicazioni per indicare tanto i protocolli applicativi come FTP o SMTP, quanto i programmi applicativi veri e propri, salvo indicazione contraria. Indicheremo inoltre con il termine utente di un servizio colui che utilizza tale servizio, sia esso direttamente una persona, un'applicazione, o un protocollo. Per esempio, il TCP è un utente dell'IP.

C'è subito da dire due cose importanti sul TCP. La prima è che lo standard del TCP non definisce né l'implementazione dello stesso, né le modalità con cui un'applicazione accede a i servizi di questo protocollo. Esso definisce solamente le caratteristiche di tali servizi, per cui si possono trovare molte

differenti implementazioni del TCP, ognuna con la propria interfaccia applicativa. Per chi non programma ricordo che un'interfaccia applicativa o API (Application Programming Interface) non è altro che l'insieme delle funzioni, delle istruzioni, dei parametri e dei blocchi di controllo che vengono utilizzati dai programmatori per accedere ai servizi di un sistema. Per esempio, se ho un sistema di posta elettronica potrei definire un'API basata su due funzioni, una chiamata `spedisci`, e una chiamata `ricevi`. Per ogni funzione sarebbero poi da definire quali informazioni sono da passare al momento dell'utilizzo (parametri in ingresso), quali si ottengono una volta espletato il servizio (parametri di ritorno), eventuali codici di errore, e le regole di utilizzo delle singole funzioni. Il motivo che sta alla base della scelta di non standardizzare l'interfaccia con il TCP è che in molti casi questo protocollo è direttamente definito nel sistema operativo, o comunque fa parte del cosiddetto corredo di base di un sistema, per cui si è voluto evitare di forzare una sintassi che potesse essere in contrasto con quella nativa del sistema ospite. Il secondo punto fondamentale è che il TCP è stato definito per funzionare su un qualsiasi sistema di trasmissione dati a pacchetto, e non necessariamente solo sull'IP. Di fatto esso può essere poggiato, per esempio, direttamente sopra una rete Ethernet senza bisogno di un livello Internet intermedio.

Ma qual è lo scopo del TCP nell'architettura internet? Il protocollo non fornisce le garanzie di affidabilità e robustezza necessarie per implementare un sistema di trasmissione dati sicuro e di facile gestione. L'IP è inaffidabile e benché schermi lo sviluppatore dalla conoscenza della rete fisica, fornisce ancora una visione di livello troppo basso del sistema di reti interconnesse. Questo vuol dire che l'IP è troppo complesso per essere utilizzato direttamente dalle applicazioni. Per avere un protocollo di trasmissione affidabile abbiamo bisogno di gestire tutte le possibili situazioni di errore, la duplicazione o la perdita dei pacchetti, la caduta delle connessioni o di un router, e via dicendo. Se le

applicazioni utilizzassero direttamente i servizi dell'IP, ognuna di esse dovrebbe implementare una serie alquanto complessa di algoritmi e servizi per tenere conto di tutto ciò. A parte il fatto che esistono relativamente pochi programmatori in grado di far questo fra gli svariati milioni di sviluppatori di applicazioni, nella maggior parte dei casi si tratterebbe di reinventare ogni volta la ruota. In generale questi problemi, seppure complessi, sono abbastanza standard, per cui si è pensato di poggiare sui sistemi di trasmissione a pacchetti un protocollo affidabile che potesse essere implementato da sviluppatori altamente specializzati, lasciando così agli altri la possibilità di concentrarsi sulla logica applicativa piuttosto che sugli aspetti specifici della trasmissione dei dati a basso livello.

Vediamo allora quali sono le caratteristiche principali del TCP, eventualmente comparate a quelle dell'IP.

Innanzitutto il TCP fornisce una visione dei dati di tipo a flusso (data stream), cioè i dati sono ricevuti in sequenza e nello stesso ordine con il quale sono stati trasmessi. A questo livello cioè, l'utente del TCP spedisce i dati come un singolo flusso di byte e nello stesso modo li riceve. Nell'IP avevamo invece la divisione dei dati in pacchetti che potevano subire un'ulteriore frammentazione se si trovavano a passare attraverso reti caratterizzate da una soglia molto bassa sulle dimensioni dei frame fisici. I pacchetti potevano inoltre arrivare in ordine sparso rispetto a quello di trasmissione.

Secondo punto: nell'IP non si sa mai a priori il cammino che effettua un pacchetto. Il TCP fornisce al suo utente una visione del collegamento come se esso fosse una linea dedicata. Ovviamente sotto sotto il meccanismo è ancora quello a pacchetti, ma la cosa è schermata agli utilizzatori del TCP. Tale caratteristica è detta *virtual circuit connection*, cioè circuito di connessione virtuale. Il TCP si basa sul concetto di connessione, piuttosto che su quello di indirizzo come fa invece l'IP. Una connessione, per definizione, richiede la definizione di due punti piuttosto che di uno solo, detti punti terminali o estremi della connessione (endpoint). Parleremo anche di interlocutori per indicare gli utenti posti agli estremi della connessione.

Terzo punto: abbiamo visto che l'IP divide i dati in pacchetti che vengono costruiti sulla base di esigenze di trasmissione legate alle varie reti fisiche su cui si poggia il sistema. D'altra parte le applicazioni dividono i dati in funzione delle esigenze applicative. Per esempio, un'applicazione di posta elettronica può considerare una lettera da 8.000 caratteri una singola unità dati, mentre un protocollo per la gestione della rete può avere l'esigenza di spedire tanti piccoli messaggi di non più di 16 byte l'uno. Il TCP permette di disaccoppiare il modo di dividere i dati delle applicazioni da quello dell'IP. Così la lettera di cui sopra viene prima spezzata in tante parti, spedita via IP e poi ricomposta dal livello TCP del destinatario, mentre per i messaggi di controllo avviene il contrario: prima vengono accumulati in un singolo pacchetto, e poi rispediti presso il destinatario. Questo meccanismo è detto *buffered transfer*. Naturalmente può sorgere l'esigenza di forzare la trasmissione dei dati anche se il buffer non è pieno. Per esempio, se serve sapere se un certo sistema è attivo o meno manderò prima un messaggio di interrogazione, e solo una volta ricevuta la conferma incomincerò a spedire gli altri dati. Dato che il

messaggio di interrogazione è più piccolo del buffer, esso non verrebbe realmente spedito dal TCP fintanto che questi non è stato riempito. È quindi necessario forzare la trasmissione del primo messaggio (push) se si vuole evitare di attendere inutilmente la risposta a un messaggio che in realtà non è mai partito.

Quarto punto: per quanto intelligente, il TCP si preoccupa di trasferire i dati che gli vengono passati senza entrare in merito a il loro significato dal punto di vista applicativo. In che modo il flusso di dati vada interpretato semanticamente è responsabilità delle due applicazioni che utilizzano la connessione TCP per cooperare. Questo vuol dire che se un'applicazione manda alla sua controparte una serie di indirizzi, questi arriveranno uno di seguito all'altro nel giusto ordine, ma senza alcuna garanzia che ogni buffer contenga un numero intero di indirizzi. Sta all'applicazione ricomporre un indirizzo capitato a cavallo di due buffer consecutivi. Si parla quindi di flusso senza struttura (Unstructured Stream).

Quinto e ultimo punto: le connessioni TCP permettono il trasferimento contemporaneo dei dati in entrambe le direzioni, quello che nel gergo delle comunicazioni si chiama una connessione full-duplex. Si hanno cioè due flussi che scorrono indipendentemente in direzioni opposte, senza interagire fra loro. Le applicazioni hanno comunque la possibilità di passare alla modalità half duplex semplicemente bloccando uno dei due flussi di dati.

Ma in che modo il TCP garantisce quella affidabilità che manca all'IP? Il meccanismo di base utilizzato sia dal TCP che da molti altri protocolli cosiddetti "affidabili" è quello della ritrasmissione in caso di mancata conferma (positive acknowledgement with retransmission). Si tratta di un meccanismo concettualmente semplice: ogni qual volta uno dei due interlocutori di una connessione spedisce dei dati, questi attende una conferma dell'avvenuta ricezione. Se questa arriva entro un tempo stabilito viene spedito il pacchetto successivo, altrimenti l'applicazione rispedisce quello precedente. Tale tempo viene misurato con un timer che viene fatto partire ogni volta che un pacchetto è spedito. Questo meccanismo risolve il problema dei pacchetti persi o danneggiati, ma può crearne un altro. Supponiamo che a causa di problemi di saturazione della rete un pacchetto ci metta molto più tempo del previsto ad arrivare. A questo punto il mittente, non vedendosi arrivare indietro la conferma ne rispedisce una copia. Succede così che il destinatario riceve a una certa distanza l'uno dall'altro due copie dello stesso pacchetto. Il problema della duplicazione dei pacchetti viene risolto facendo numerare sequenzialmente al mittente tutti i pacchetti da spedire e facendo verificare al destinatario la sequenza ricevuta. Naturalmente questo non vale solo per i messaggi ma anche per le conferme agli stessi. Infatti anche una conferma potrebbe venire erroneamente duplicata. Per evitare questo ogni conferma riporta il numero di sequenza del messaggio a cui si riferisce, permettendo così al mittente di verificare che a ogni messaggio spedito corrisponda una e solo una conferma di ricezione. È un po' lo stesso meccanismo di una raccomandata con ricevuta di ritorno.

In realtà gli algoritmi utilizzati dal TCP sono un po' più complicati, e tengono conto di tutta una serie di situazioni che si possono verificare. Senza contare che il tempo di attesa prima della ritrasmissione è un punto chiave di tutto il discorso. Se si attende troppo poco si rischia di generare un sacco di duplicati inutili, saturando per giunta la rete, mentre se si attende troppo si rischia di abbassare notevolmente e inutilmente le prestazioni della trasmissione dei dati, rallentando le applicazioni alle estremità della connessione.

Il meccanismo della conferma di ricezione con ritrasmissione ha inoltre un grosso svantaggio. Anche se i tempi di attesa sono scelti in modo ottimale, esso causa un notevole sottoutilizzo della rete. Infatti, indipendentemente dalla capacità della rete, i due interlocutori passano la maggior parte del tempo attendendo le varie conferme. È un po' come avere un tubo nel quale vengono fatte cadere una a una delle palline numerate in sequenza. All'altra estremità del tubo c'è una cesta poggiata su un prato, un po' distante dal foro di uscita. Se la pallina cade nella cesta fa rumore, altrimenti cade nel prato e non si sente niente. Se ogni volta che metto una pallina nel tubo aspetto di sentire il rumore che mi conferma che la pallina è caduta nel cesto, il tubo resta per la maggior parte del tempo vuoto. Una tecnica di ottimizzazione usata dal TCP per rendere più efficiente il meccanismo appena descritto è quella delle finestre di scorrimento (sliding window). Funziona più o meno in questo modo. Immaginate di immettere nel tubo una sequenza di dieci palline senza attendere che la prima sia arrivata. Come si sente il primo flop si aggiunge un'undicesima pallina, e poi una dodicesima e così via. Se si salta un flop si reinserisce una pallina con lo stesso numero di quella che non è arrivata, tanto il destinatario può comunque riordinare le palline utilizzando i numeri scritti sopra. Il numero di palline che compongono il trenino da spedire indipendentemente dalla ricezione del flop si chiama dimensione della finestra di scorrimento (sliding window size). Se si sceglie una dimensione tale da riempire tutto il tubo nella sua lunghezza si sfrutta al massimo la capacità dello stesso.

In pratica questo sistema divide la sequenza di pacchetti in tre fasce. La prima è rappresentata dai pacchetti spediti e di cui si è avuta la conferma di ricezione. La seconda è formata dai pacchetti spediti ma dei quali non si sa ancora niente, e la terza è formata dai pacchetti ancora da spedire. Con questa tecnica il TCP mantiene un timer per ogni singolo pacchetto che appartiene alla seconda fascia. Il nome "Finestra di scorrimento" deriva dal fatto che è come se ci fosse una finestra ampia quanto il treno di pacchetti che possono essere spediti senza attendere la conferma dell'avvenuta ricezione che scorre in avanti un pacchetto alla volta ogni qual volta arriva una conferma. Anche in questo caso, come in quello del tempo di attesa prima di ritrasmettere un pacchetto, le dimensioni della finestra di scorrimento rappresentano un fattore critico per determinare l'efficienza del sistema. In generale, se le dimensioni della finestra sono maggiori del tempo di attesa per il singolo pacchetto, allora la finestra continua a scorrere regolarmente senza interruzioni, salvo nel caso di ritrasmissioni, e la capacità di carico della rete viene sfruttata al massimo.

Affinché infatti due applicazioni possano comunicare fra di loro esse debbono conoscersi e il sistema di trasmissione che le serve deve sapere a chi effettivamente vanno recapitati i dati. È evidente che non basta l'indirizzo IP, che identifica univocamente un host nella rete. Basti pensare che un PC collegato in rete ha in genere un solo indirizzo IP, a meno che non sia collegato a più reti fisiche, come per esempio un gateway. Se una lettera viene spedita via rete a un certo indirizzo come fa TCP a sapere a quale applicazione deve far arrivare i dati? Un sistema potrebbe essere quello di assegnare un identificativo a ogni singola applicazione, ma come garantire allora l'univocità dell'identificativo? Senza contare che questo costringerebbe la controparte a sapere a priori tale valore per ogni possibile destinatario. Non è inoltre detto che un utente utilizzi sempre lo stesso programma per spedire o ricevere la posta elettronica. In realtà, più che la specifica applicazione, quello che è importante identificare è la funzione, come per esempio trasferire file oppure spedire posta elettronica.

La soluzione è quella di definire dei punti di ingresso e di uscita virtuali chiamati porte. Ogni porta rappresenta di fatto un punto di accesso a un'applicazione nel sistema. Si tratta in pratica di un'estensione del concetto di porta hardware. Un PC moderno, per esempio, può avere porte hardware parallele, seriali, video, audio e di vari altri tipi. Ad ogni porta possono essere attaccati dispositivi molto differenti. Per esempio, a una porta parallela è possibile attaccare una stampante, uno scanner, un'unità Cd-Rom oppure un'unità disco ad alta capacità. Tutti questi dispositivi non hanno bisogno di una porta specifica, ma possono utilizzare la stessa porta perché gestiscono flussi di dati simili, possono cioè usare lo stesso protocollo di base per la trasmissione dei dati. Ovviamente, a livello applicativo, ogni periferica darà ai propri dati una struttura differente. Questo vuol dire che i dati costruiti per una stampante non possono certo essere mandati a uno scanner. D'altra parte anche TCP non entra in merito della struttura applicativa dei dati, ma solo alle modalità di trasmissione degli stessi.

Ogni porta TCP è identificata da un numero. I numeri sotto il 256 sono utilizzati per le cosiddette "porte conosciute", cioè porte alle quali è stata assegnata una responsabilità ben precisa, mentre quelli al di sopra sono utilizzati per le assegnazioni dinamiche. Avremo per esempio una porta per i servizi di posta elettronica X.400 chiamata appunto X400 (103) alla quale faranno riferimento tutte le applicazioni che utilizzano tali servizi, oppure le due porte per il trasferimento dei file via FTP, una per il controllo (FTP, 21) e una per i dati (FTP-DATA, 20). Una lista delle porte conosciute attualmente assegnate è riportata nella RFC 1060, reperibile a <ftp://ds.internic.net/rfc/rfc1060.txt> Mentre in UDP la porta rappresenta un elemento sufficiente alla comunicazione, per cui il protocollo non fa altro che smistare i vari datagrammi nelle code dati (queue) associate alle varie porte. Una connessione è l'insieme di due punti, detti estremi della connessione (endpoint), ognuno identificato univocamente da due coordinate: l'indirizzo IP e il numero di porta. Una connessione è quindi rappresentata da ben quattro identificativi: gli indirizzi IP delle due macchine su cui girano le due applicazioni che si scambiano i dati, e i rispettivi numeri di porta. È importante capire che l'identificazione della connessione richiede tutti e quattro i valori, per cui la stessa porta con lo stesso indirizzo IP può essere condivisa simultaneamente da più connessioni senza creare alcun problema o ambiguità.

Ecco perché in TCP si pensa in termini di linea dedicata. È come se ci fosse un filo che lega univocamente i due interlocutori. Ogni interlocutore può avere più connessioni aperte nello stesso momento a partire dallo stesso capo purché non ce ne siano due con la stessa controparte. Il vantaggio è che una singola applicazione, per esempio di posta elettronica, necessita di una sola porta TCP per fornire servizi a molte macchine contemporaneamente attraverso differenti connessioni che condividono uno stesso estremo. Va tenuto presente che, anche se UDP e TCP usano gli stessi numeri per le porte, non esiste possibilità di confusione, dato che i pacchetti IP portano con sé l'identificativo del protocollo utilizzato che è ovviamente diverso per i due protocolli.

Affinché la connessione venga stabilita, entrambi gli estremi devono dare la loro autorizzazione. L'aggancio avviene nel seguente modo. Una delle due applicazioni che si vogliono connettere effettua un'apertura passiva (passive open), cioè informa il suo sistema che è disposta ad accettare una richiesta di connessione. TCP assegna all'applicazione un numero di porta. L'altra applicazione deve invece effettuare un'apertura attiva (active open), specificando l'indirizzo IP e la porta con la quale si vuole connettere. A questo punto i due livelli TCP stabiliscono la connessione e verificano che tutto sia a posto.

La gestione dei dati

Vediamo adesso come TCP gestisce i dati. Innanzi tutto, come già detto, TCP vede i dati come una sequenza non strutturata di ottetti, cioè byte, detto flusso di dati (data stream). Questo flusso viene diviso in segmenti ognuno dei quali viaggia di solito in un singolo pacchetto IP. Per aumentare l'efficienza della trasmissione, TCP utilizza una versione particolare del meccanismo a finestre di scorrimento spiegato sopra. Ricordo che questo meccanismo consiste nel mandare un gruppetto di dati prima di aver ricevuto la conferma di ricezione di ogni singolo pacchetto, in modo da tenere costantemente sotto carico la linea. Se infatti si dovesse attendere la conferma di ricezione per ogni singolo pacchetto prima di spedire il successivo la linea resterebbe per la maggior parte del tempo inutilizzata. Si dà insomma fiducia alla rete, partendo dal presupposto che la perdita di dati sia l'eccezione piuttosto che la regola.

Esistono tuttavia due importanti differenze tra il meccanismo base presentato prima e quello più sofisticato utilizzato effettivamente da TCP.

La prima è che l'unità base per i dati non è né il segmento né il pacchetto IP ma il singolo ottetto. Ogni ottetto viene numerato e TCP mantiene tre puntatori per ogni flusso di dati in uscita: uno che separa gli ottetti già spediti e arrivati felicemente a destinazione da quelli di cui non si hanno ancora notizie, uno che separa quelli già spediti da quelli che devono ancora essere spediti senza attendere la conferma di ricezione per i precedenti ottetti, e uno che separa questi ultimi da quelli che non possono essere spediti fintanto che la finestra non scorre in avanti. Una serie di informazioni speculari è mantenuta dal destinatario che deve ovviamente ricostruire il flusso di dati nel modo corretto indipendentemente dall'ordine di arrivo dei dati. Dato che una connessione è full-duplex, TCP manterrà quindi per ogni connessione due finestre di scorrimento, una per i dati in uscita e una per quelli in ingresso: un totale di quattro finestre per connessione considerando entrambi gli estremi. Esiste quindi un'asimmetria rispetto al meccanismo base dove l'unità dati utilizzata nella finestra di scorrimento era la stessa utilizzata nella trasmissione. Qui TCP utilizza il segmento come unità dati da trasmettere, mentre ragiona in termini di ottetti per quello che riguarda il meccanismo di ritrasmissione.

La seconda differenza è che le dimensioni della finestra di scorrimento non sono fisse ma variano nel tempo in funzione della capacità di ricezione del destinatario. Ogni conferma di ricezione che ritorna al mittente contiene una soglia di capacità (window advertisement) che contiene il numero di ulteriori ottetti che il destinatario è in grado di ricevere. In pratica questo meccanismo permette di adattare la finestra di spedizione alle dimensioni del buffer di ricezione. Si tratta cioè di un meccanismo di controllo del flusso dei dati che limita il numero dei dati in ingresso man mano che il buffer di ricezione si riempie, fino a poter interrompere momentaneamente la trasmissione nel caso che si sia raggiunta la massima capacità di ricezione del destinatario. Basta infatti che il destinatario mandi una soglia uguale a zero perché il mittente interrompa la spedizione degli ottetti fino all'arrivo di una conferma di ricezione contenente di nuovo una soglia maggiore di zero.

In realtà il mittente non smette del tutto di mandare dati. Innanzi tutto, se ci sono dati urgenti da spedire, il mittente informa comunque il destinatario di tale necessità trasmettendo un segmento con un indicatore di urgenza al suo interno. Questo permette al destinatario di prendere delle contromisure per ricevere comunque i dati urgenti, per esempio aumentando le dimensioni del buffer. In secondo luogo, è sempre possibile che la conferma con soglia positiva che dovrebbe far ripartire la trasmissione dei dati vada perduta. Per questo motivo il mittente prova ogni tanto a far partire un segmento per vedere se per caso il destinatario è di nuovo pronto a ricevere i dati.