



## Guida TCP/IP

Miliardi di bit viaggiano ogni giorno sulla Rete. Vi siete mai chiesti come fanno ad arrivare al corretto destinatario? In questo articolo, primo di una serie vi presentiamo "la suite di protocolli TCP/IP" cioè le regole utilizzate per la trasmissione su Internet. Non è certo fondamentale sapere come funziona uno spinterogeno o un albero di trasmissione per guidare un'automobile. Analogamente, al giorno d'oggi, non serve sapere come funziona un computer per poterlo utilizzare. La tecnologia ci scherma sempre di più dal come una cosa funziona spostando l'attenzione sul cosa fare per utilizzarla. Così, quella che era una volta tecnologia per un'élite abbastanza ristretta di studiosi, ricercatori e studenti, è oggi una realtà a disposizione di tutti. Non solo è diventato semplice navigare nella Ragnatela, ma oggi chiunque può facilmente costruirsi le sue pagine e agganciarle a uno dei tanti siti Web che ospitano pagine private. Non c'è più neanche bisogno di conoscere l'HTML, grazie alla proliferazione di editor HTML commerciali e di pubblico dominio. In quanto ai risultati estetici, beh, lì non c'è programma che tenga.

Ma cosa c'è sotto a tutto ciò? Per chi è ancora e nonostante tutto interessato a capire come funzionano le cose, e se vogliamo anche per coloro ai quali la cosa non interessa per niente, ma hanno qualche minuto per leggere un paio di paginette e poi, chissà, potrebbe sempre tornare utile... insomma, per chi vuole, ecco a voi il "TCP/IP, questo sconosciuto".

Il nome completo è TCP/IP Internet Protocol Suite, ed è un insieme di protocolli di trasmissione di cui i due principali sono appunto il TCP (Transmission Control Protocol) e l'IP (Internet Protocol). Ma che cosa è esattamente un protocollo? Essenzialmente è una serie di regole per comporre dei messaggi e per far sì che essi possano essere scambiati tra due macchine. Non stiamo parlando solo di computer. Anche una centrale telefonica meccanica può ricadere in questa definizione. Un protocollo può contenere regole estremamente dettagliate, come quelle che identificano il significato di ogni singolo bit nella costruzione di un messaggio, oppure fornire uno scenario di alto livello, come per esempio definire come avviene il trasferimento di un file da un computer a un altro.

Fondamentalmente un protocollo sta alla trasmissione dati come un linguaggio di alto livello quale il C++ sta alla programmazione. Infatti, un linguaggio di programmazione comprende sia regole estremamente dettagliate che devono essere seguite alla lettera - guai a dimenticare anche un solo punto e virgola alla fine di un'istruzione C++ - sia strutture di alto livello che vanno costruite nel modo corretto, pena errori nella struttura logica del programma.

Una generica architettura di trasmissione è formata da una torre a più piani, dove ogni piano rappresenta una precisa responsabilità nella trasmissione dei messaggi. Alla base della torre sta la porta di accesso alla rete fisica, che potremmo pensare come una rete di strade. Ogni piano prende il messaggio che arriva dal piano superiore, lo mette in una busta con alcune informazioni aggiuntive, e lo passa come messaggio al piano inferiore.

Le regole di comunicazione tra i vari piani sono dette interfacce. Il messaggio risultante, formato da tante buste una dentro l'altra, viene immesso nella rete dalla porta che si trova alla base della torre. Una volta arrivato al piano terreno infatti, esso viene trasportato alla torre di destinazione e da qui risale un piano dopo l'altro fino all'ultimo piano, detto anche livello applicativo. Ogni piano della torre di destinazione apre solo la busta che gli compete e usa le informazioni aggiuntive per recapitare la busta successiva al piano superiore. Le

informazioni aggiuntive rappresentano il protocollo di comunicazione. Ogni piano comunica quindi solo con il piano corrispondente.

Esempio: Il direttore della Pippo e Figli manda una lettera riservata al direttore della Pluto e Consorte. Il modo con cui i due comunicano, per esempio i riferimenti a lettere precedenti, lo stile della lettera, il modo di salutare alla fine della lettera, e così via, rappresenta il protocollo ad alto livello, cioè quello applicativo. Per spedire la lettera il direttore lo passa alla sua segretaria. Ciò avviene secondo le regole interne della

Pippo e figli, ed è perciò un'interfaccia. La segretaria prende la lettera, la mette in una busta aggiungendo il nome del destinatario e la scritta RISERVATO. Queste informazioni sono per la sua controparte nella Pluto e consorte, ed è quindi anch'esso un protocollo. La busta viene quindi passata all'Ufficio Posta dell'edificio secondo la procedura ordinaria (altra interfaccia), il quale aggiunge l'indirizzo completo, il CAP e altre informazioni necessarie alla spedizione, e la passa quindi al corriere, che rappresenta il meccanismo fisico di trasferimento del messaggio. Terzo protocollo. Quando la lettera arriva, questa viene gestita dall'Ufficio Posta della Pluto e consorte che, dopo aver buttato la busta esterna con l'indirizzo dell'edificio, la passa alla segreteria del direttore. Questa registra l'arrivo della missiva, la toglie dalla busta più interna, e poi consegna la lettera vera e propria al direttore della Pluto e consorte.

È evidente che perché il sistema funzioni bisogna che i vari protocolli siano rispettati da entrambe le aziende. Mentre infatti le interfacce possono essere diverse (ogni azienda ha le sue procedure interne), i protocolli devono essere stati concordati prima, altrimenti alcune informazioni potrebbero andare perse, o addirittura la lettera potrebbe non essere recapitata in tempo. Pensate a una segretaria italiana che riceve una busta da una consociata asiatica con sopra scritto URGENTE in cinese! Ne nasce una considerazione importante. La base di ogni protocollo è il concetto di standardizzazione. Più vasta è l'accettazione dello standard, più forte è diffuso è il protocollo. Gli standard internazionali sono in genere i più importanti, ma non sempre. Un esempio è proprio il TCP/IP, nato per volontà dell'agenzia americana DARPA (Defense Advanced Research Projects Agency) e poi diventato di fatto il maggior sistema di protocolli per l'interconnessione di reti a livello mondiale.

Internet è fatta a strati Internet è basato su tre livelli concettuali: il livello applicativo (Application Services), quello del trasporto (Reliable Stream Transport Service) e quello della spedizione dei pacchetti (Connectionless Packet Delivery Service). Per capire il TCP/IP, è necessario a questo punto capire bene che cosa è Internet. Tanto per cominciare Internet non è una rete di comunicazione. Una rete di comunicazione è in genere legata alle necessità specifiche di chi l'ha disegnata e dell'hardware utilizzato per implementarla. Costruire una rete ideale che vada bene per qualsiasi esigenza, o pensare di poter limitare a un solo tipo di hardware l'implementazione di una qualunque rete non solo non è fattibile, ma neanche auspicabile, date le limitazioni delle tecnologie attuali. A volte è necessario far correre i dati molto velocemente in un ambito molto ristretto, come per esempio all'interno di un edificio. Altre volte si ha l'esigenza di trasmettere dati a migliaia di chilometri di distanza in modo molto affidabile, anche se questo può significare un rallentamento nella velocità di trasmissione. Se si cercasse di utilizzare lo stesso hardware in entrambi i casi, i costi sarebbero assolutamente inaccettabili.

La soluzione è l'interconnessione delle reti, o internetworking. Grazie a ponti di collegamento (detti gateway) e la definizione di opportuni protocolli, si possono collegare fra di loro reti anche molto diverse, fornendone agli utenti una visione comune. Questa è la forza di Internet rispetto alle varie reti proprietarie, e di conseguenza del TCP/IP sui vari protocolli proprietari. Il TCP/IP è un insieme di regole pubbliche, aperte a tutti, o come si dice nell'ambiente, un sistema aperto (open system), che permette l'interconnessione di reti anche molto differenti, indipendentemente dalla tecnologia usata da ogni rete. I suoi principali vantaggi sono appunto l'indipendenza dalle tecnologie delle singole reti interconnesse, la

possibilità di far comunicare fra di loro ogni computer connesso al sistema, la possibilità di trasmettere conferme di ricezione (acknowledgement) direttamente dal destinatario al mittente, e soprattutto una notevole quantità di protocolli applicativi per qualunque possibile bisogno, come vedremo più avanti. Il TCP/IP definisce quindi una unità di trasmissione dati chiamata datagram, e le regole da seguire per trasmettere un datagram in una particolare rete.

Il principio che sta alla base dell'interconnessione è quello di schermare le applicazioni dalle caratteristiche fisiche delle reti in modo semplice e flessibile. Questo avviene attraverso un livello intermedio che si occupa di spedire e ricevere piccoli pacchetti di dati fra due punti qualsiasi del sistema di reti. Questo meccanismo si chiama packet-switching. Esso consiste nella divisione di ogni messaggio in un certo numero di pacchetti di dati. Ogni pacchetto è formato da poche centinaia di byte, e contiene una intestazione che fornisce informazioni sul destinatario e su come raggiungerlo. Questo meccanismo ha il vantaggio di ottimizzare l'utilizzo della rete, parallelizzando la trasmissione di più messaggi contemporaneamente. Lo svantaggio è che ogni nuovo sistema che si aggancia alla rete per trasferire dati riduce la disponibilità della rete per tutti gli altri sistemi già connessi. Una rete infatti ha una certa capacità ben definita, che dipende sostanzialmente dalla tecnologia hardware e software che utilizza. Tale capacità viene misurata in bit per second (bps). Questa grandezza non rappresenta la velocità dei dati in rete, come si potrebbe pensare in prima istanza, bensì dà una misura del numero massimo di bit che possono essere trasmessi nella rete in un secondo. La velocità reale di un singolo messaggio dipende da tanti fattori, come il numero di sistemi che stanno utilizzando la rete, la qualità delle connessioni e di conseguenza il numero di tentativi necessari per trasferire correttamente i dati, le modalità di trasmissione e i dati aggiuntivi necessari al trasferimento degli stessi.

Ci sono altri modi per trasferire dati in una rete: per esempio, quando fate una telefonata, la rete stabilisce un collegamento diretto fra il vostro telefono e quello della persona chiamata. A questo punto il telefono incomincia a campionare il microfono della vostra cornetta in modo continuo, trasferendo il segnale al ricevitore all'altro capo. Il tutto a 64.000 bit per secondo, che è la velocità di campionamento necessaria a digitalizzare la voce. Questo avviene comunque, indipendentemente dal fatto che stiate parlando o meno. Anche se state in silenzio la linea è saturata al massimo della sua capacità. Questo meccanismo è detto circuit-switching. Al contrario del meccanismo usato dal TCP/IP, quello cioè a pacchetti, la linea è completamente assegnata alla comunicazione in atto, per cui il fatto che altri stiano telefonando non riduce la capacità della connessione. D'altra parte la linea è utilizzata completamente indipendentemente dal fatto che ci siano o meno dati da trasferire. Di qui gli elevati costi di tale meccanismo. La telefonata, infatti, la pagate lo stesso sia che parliate molto velocemente, sia che stiate completamente in silenzio. Questo meccanismo è troppo costoso per una rete informatica, specialmente se si tiene conto che la disponibilità di tecnologie hardware sempre più raffinate e veloci per il trasferimento dei dati bilanciano in buona parte quello che è uno dei punti deboli del sistema a pacchetti, e cioè l'impossibilità di garantire a ogni utente e in ogni momento una certa capacità di trasferimento ben definita.

Torniamo al sistema a pacchetti. Per trasferire dati da un sistema a un altro ogni sistema ha un nome unico ben definito. Non esistono cioè due sistemi con lo stesso nome, anche se in reti diverse, indipendentemente da quale è il nome locale di un sistema nella sua rete di appartenenza. All'interno di ciascuna rete, i vari computer usano la tecnologia hardware e software specifica di quella rete.

Tuttavia, grazie a questo strato intermedio di software, le varie applicazioni hanno una visione unica e globale del sistema interconnesso di reti, detto appunto internet. Notate la "i" minuscola. Il concetto di internet è infatti quello appena descritto. Viceversa Internet con la "I" maiuscola, identifica quel sistema di reti, basato sull'architettura internet, che viene detto anche Connected Internet.

La connessione tra due reti avviene attraverso macchine opportune che sono collegate fisicamente a entrambe le reti, e hanno la responsabilità di far passare i vari pacchetti da una rete all'altra e viceversa. Tali macchine sono dette internet gateway, o anche IP router. Sono loro il vero elemento portante di una internet. Ogni router non solo deve sapere che determinati pacchetti vanno passati da una rete a un'altra, ma deve passare dall'altra parte anche pacchetti destinati a ulteriori reti connesse attraverso altri router. Essi però ragionano solo in termini di reti, non di destinazione finale. A un router non interessa chi è effettivamente il destinatario di un pacchetto, ma solo a quale rete appartiene. Questo semplifica molto l'implementazione di un router. Alla base del meccanismo dei router c'è l'indirizzo IP, o IP address.

Ogni cosa che conosciamo ha un nome. Cane, casa, auto, e via dicendo. Se ci interessa specificare meglio ciò di cui stiamo parlando, possiamo assegnare un nome anche a un sottogruppo di cose. Così abbiamo che i cani bassotti sono alquanto diversi dai San Bernardo, una catapecchia non è certo una villa, e una Ferrari costa un po' più di una Cinquecento. Se poi dobbiamo identificare una cosa in modo chiaro e univoco, è necessario assegnarle un nome che solo quella cosa ha. Già un nome come Mario Rossi non va bene, perché non è unico, e comunque, anche se scegliessimo oggi un nome veramente strano e originale, non avremmo la garanzia in futuro di non ritrovarci con un caso di omonimia. Ecco allora le targhe per le automobili, i codici fiscali per le persone, i numeri di telefono, e via dicendo. Ognuno di questi nomi ha tre caratteristiche. La prima è che esiste un organo competente centrale che li assegna, proprio per garantirne l'univocità. La seconda, è che hanno una struttura a sottogruppi. Esistono cioè degli elementi che garantiscono l'univocità a un certo livello, all'interno del quale esiste una certa libertà di scelta, e così via, livello dopo livello. Per esempio, il codice fiscale viene costruito in modo che un uomo e una donna non possano mai avere lo stesso codice, anche se fossero nati lo stesso giorno, nella stessa città e si chiamassero nello stesso modo. Similmente, i numeri di telefono di due città diverse si distinguono per il prefisso e se queste si trovano anche in stati diversi, per il prefisso internazionale.

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.

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 dell'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.

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@tecnnet.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'informazione 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 alcuna 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, ovvero sia 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. Il campo tra la lunghezza dell'intestazione e quella totale del datagram identifica il tipo di servizio che va offerto al pacchetto, ed è formato da un campo di 3 bit che specifica l'importanza che va data al datagram, e da tre campi da 1 bit ciascuno che identificano il tipo di trasporto desiderato per questo pacchetto. Purtroppo questo campo non può essere sempre preso in considerazione da tutte le reti, in quanto non sempre la rete fisica è in grado di soddisfare le richieste di priorità e trasporto memorizzate in questo campo. Per cui esso viene considerato una sorta di raccomandazione alla rete, piuttosto che un vero obbligo. In ogni caso il campo di priorità può contenere valori da 0 a 7. Lo zero è il valore di base di un normale pacchetto, mentre il 7 rappresenta la richiesta di precedenza più elevata, e va usato per i datagram che contengono dati per il controllo della rete stessa. I tre bit relativi al tipo di trasporto servono a definire il livello di qualità relativo al trasferimento del pacchetto. Se impostati a uno, essi richiedono rispettivamente: di evitare al massimo ritardi nel recapitare il pacchetto al destinatario, di fornire la massima capacità di trasferimento, e di garantire un'elevata affidabilità durante il trasporto. Ovviamente è estremamente difficile poter fornire tutti e tre questi servizi contemporaneamente. Spesso la rete non riesce a garantirne neanche uno solo. I tre campi successivi vengono utilizzati nel meccanismo di frammentazione spiegato poco fa, e in particolare sono quelli che permettono all'host che riceve i vari frammenti di riassembleare il tutto per ottenere il datagram originario. Essi sono assolutamente necessari in quanto non è prevista alcuna comunicazione tra il mittente e il destinatario su come ricomporre il datagram, tanto più che la frammentazione finale può essere il risultato di più frammentazioni successive. Inoltre i vari frammenti possono arrivare in qualunque ordine, dato che possono avere seguito cammini differenti. Dulcis in fundo, anche se l'intestazione di ogni frammento è ottenuta da quella del datagram originale, il quarto campo dell'intestazione di un frammento contiene la sua lunghezza totale, e non quella di tutto il datagram. Quest'ultima informazione deve essere calcolata dal destinatario in qualche modo. Ed ecco il perché di questi tre campi. Il primo campo serve a identificare univocamente il datagram ed è lungo 16 bit. Tutti i frammenti che appartengono a uno stesso datagram hanno lo stesso identificativo. Il secondo campo è una maschera di 2 bit che controlla il meccanismo di frammentazione. Il primo bit specifica se il datagram può essere frammentato: se impostato a uno, la frammentazione non è permessa. Il secondo bit serve a marcare l'ultimo frammento. Vedremo tra un attimo a cosa serve. Il terzo campo contiene la posizione dei dati del frammento nel blocco originale di dati misurato in parole da 64 bit. Questo campo si chiama fragment offset. Per esempio, se un frammento ha un offset 7, vuol dire che il primo bit dei suoi dati corrisponde al quattrocentoquarantanovesimo bit dei dati del frammento originale, dato che  $7 * 64 + 1$  fa appunto 449. A questo punto è chiaro come si può ottenere la

lunghezza totale del datagram originario. Basta sommare l'offset e la lunghezza totale dell'ultimo frammento, riconoscibile grazie al secondo bit del campo di controllo.

Il campo successivo, posto a partire dal 64° bit dell'intestazione, è lungo un byte e serve a stabilire quanto a lungo un datagram può rimanere nella rete. È cioè il campo che specifica la scadenza di un datagram di cui avevamo accennato in precedenza. L'idea originaria era che tale campo contenesse il numero massimo di secondi che il pacchetto potesse restare nella rete. Tuttavia, data l'evidente difficoltà di sincronizzare gli orologi di tutti gli hosts e i gateway della rete, si è deciso di semplificare il meccanismo come segue: ogni gateway che processa il pacchetto decrementa il campo di uno quando questo arriva e memorizza il tempo di arrivo. Se il pacchetto non riparte subito ma rimane in attesa nel gateway, il valore di questo campo viene ulteriormente decrementato di una unità per ogni secondo di attesa. Come il campo arriva a zero, il datagram viene cancellato dalla rete e un messaggio di errore viene rispedito al mittente.

Il campo seguente, lungo 8 bit, identifica il protocollo di alto livello utilizzato che ha generato i dati contenuti nel datagram, e definisce di fatto il loro formato. Ne riparleremo in seguito, quando vedremo i protocolli applicativi. Abbiamo quindi un campo di controllo di 16 bit che serve a verificare l'integrità dell'intestazione, e che utilizza il meccanismo di checksum ben conosciuto nel mondo del software. In suo valore è la somma complementata a uno delle parole da 16 bit che compongono l'intestazione, addizionate con il metodo del complemento a uno. Quindi vengono gli indirizzi IP del mittente e del destinatario, ognuno lungo 32 bit. Di tali indirizzi e del loro scopo abbiamo parlato esaurientemente nella seconda e terza parte di questo corso.

Per finire abbiamo un campo di lunghezza variabile che può contenere varie opzioni, e quindi il campo di riempimento di cui abbiamo già parlato. Queste opzioni non sono presenti in tutti i datagram e vengono usate prevalentemente nelle verifiche e nella identificazione dei problemi della rete.

Parliamo ora di due aspetti fin qui solo accennati: i meccanismi di instradamento dei pacchetti (routing) e la gestione degli errori. Incominceremo a salire nella torre dei protocolli internet, introducendo il primo protocollo che si poggia sull'IP, e precisamente lo User Datagram Protocol (UDP). Come vedremo si tratta ancora di un protocollo molto legato all'IP, ma comunque considerato al di sopra di questo.

Come abbiamo già detto in precedenza, IP è un protocollo connectionless. Questo vuol dire che non esiste un collegamento diretto tra i due host che si scambiano dati, bensì una rete di connessioni attraverso la quale si possono identificare vari potenziali cammini da un host all'altro. Il cammino attraverso il quale i dati giungono all'host destinatario è scelto dinamicamente e può variare per ogni singolo pacchetto di dati.

Tale scelta non avviene quando il pacchetto parte, ma è il risultato di numerose decisioni prese a ogni singolo gateway. Per questo motivo i gateway sono detti anche router. Tali scelte possono dover tenere conto di molti elementi, quali la priorità del messaggio, il carico di rete, la capacità delle reti intermedie, e via dicendo. La base tuttavia del meccanismo sono le tabelle di instradamento (routing table). Vediamo di che si tratta.

Consideriamo prima una singola rete fisica. Se un host vuole spedire un messaggio a un altro host nella stessa rete, non deve far altro che incapsulare il messaggio in un datagramma IP fornendo quindi l'indirizzo IP del destinatario, e passare il tutto al livello inferiore. Questi provvederà a ricavare dall'indirizzo IP l'identificativo del destinatario nella rete fisica, a incapsulare il datagramma in un frame, e a spedire direttamente il tutto all'host finale. Questa tecnica si chiama instradamento diretto (direct routing).

Vediamo adesso quello che succede quando abbiamo due reti interconnesse tramite un gateway. L'host mittente si accorge che il destinatario non è nella sua rete fisica, dato che il network id del suo indirizzo IP è diverso da quello a cui deve

spedire il datagramma. Spedisce allora il messaggio al gateway passando sia il datagramma che l'indirizzo IP del gateway al livello inferiore. Il messaggio arriva quindi direttamente al gateway che estrae l'indirizzo IP del destinatario, si accorge che fa parte della seconda rete a cui è connesso, e spedisce quindi il tutto all'host finale attraverso la rete fisica. Questa tecnica si chiama di instradamento indiretto (indirect routing).

In questo secondo caso la tabella di instradamento è semplice, dato che il gateway ritrasmette sempre i messaggi che devono andare da una rete all'altra attraverso la rete fisica. Se invece abbiamo più gateway tra i due host, ogni gateway, tranne l'ultimo, dovrà spedire il messaggio a un altro gateway.

Per far questo userà appunto la tabella di instradamento che fornisce per ogni possibile rete destinataria finale l'indirizzo IP del gateway successivo. È da notare che queste tabelle non contengono di solito gli indirizzi IP di tutti i possibili host destinatari, cosa che sarebbe fisicamente impossibile, bensì gli indirizzi delle reti raggiungibili a partire da quel gateway. Esiste poi la possibilità di specificare un gateway di default e cammini specifici per host speciali. La prima cosa è molto comune, mentre la seconda è usata solo in casi eccezionali. La logica di instradamento è quindi quella riportata nel diagramma. La gestione dei messaggi di errore: Come si può facilmente capire dal meccanismo di instradamento appena spiegato, non è possibile sapere se il destinatario effettivamente esiste fintanto che il datagramma non arriva all'ultimo gateway. In generale l'IP non contiene grossi meccanismi di verifica, ed è per questo che è detto "inaffidabile". Esso richiede quindi un protocollo ausiliario per l'emissione di messaggi di errore in rete, che permettano ai protocolli di livello superiore di implementare una logica più affidabile e robusta. Tale protocollo è chiamato Internet Control Message Protocol (ICMP).

L'ICMP è considerato parte integrante dell'IP ed è sostanzialmente un protocollo per la segnalazione di errori il cui utente principale è l'IP stesso. Solo in casi particolari l'ICMP arriva a informare dell'errore eventuali livelli superiori all'IP. In ogni caso cosa fare quando avviene un errore non spetta all'ICMP, ma ai processi che se ne avvalgono. L'ICMP informa sempre l'IP mittente, non i vari gateway intermedi. Questo in quanto del cammino percorso da un datagramma non rimane traccia, per cui l'unico possibile destinatario di un messaggio di errore è solo chi ha generato il datagramma. Inoltre, dato che i datagrammi ICMP viaggiano incapsulati in datagrammi IP, come un qualunque messaggio di livello superiore, essi sono soggetti alle stesse limitazioni in termini di affidabilità di qualunque altro messaggio spedito via TCP/IP. Non analizzeremo in dettaglio tutti i datagrammi ICMP, anche perché ognuno può avere una struttura differente. Diremo solamente che l'intestazione di un datagramma ICMP contiene sempre almeno tre campi: il tipo di messaggio, un codice di errore, e una somma di controllo.

Il livello di Trasporto: Come sicuramente ricorderete, la torre Internet si può schematizzare più o meno su quattro livelli. Alla base della torre sta l'hardware che rappresenta la rete vera e propria. Sopra a questo sta il primo livello, quello cioè di interfaccia alla rete fisica, detto appunto Network Interface o anche Data Link. I protocolli a questo livello si scambiano blocchi di dati chiamati frame, la cui struttura è strettamente legata alle caratteristiche hardware della rete stessa. Al di sopra di questo livello c'è il livello di interconnessione fra reti, ovvero il livello dell'IP la cui unità di scambio dati è appunto il datagramma IP, mentre il terzo livello è quello detto di Trasporto. Concettualmente è a questo livello che si pongono sia il TCP che appunto l'UDP. L'unità di scambio dati al terzo livello si chiama pacchetto (transport packet). Il quarto livello è infine quello Applicativo, al quale vengono scambiati i messaggi applicativi (message e data stream).

Abbiamo visto come l'IP permette di scambiare datagrammi fra host, cioè a livello di macchine. Tuttavia non è certo la macchina il destinatario finale dei dati che fluiscono nella rete, bensì le applicazioni e i programmi che girano su di essa. I moderni sistemi operativi permettono di far girare più processi contemporaneamente, e comunque questi non hanno le caratteristiche di permanenza che ha invece un host.

Un programma infatti è un componente dinamico e temporaneo in un sistema. Non è quindi pensabile di poter associare a un processo un identificativo fisso come si fa con gli host e gli indirizzi IP. Un processo infatti non ha un identificativo univoco in un sistema, dato che ogni volta che viene lanciato esso può assumere un identificativo diverso. Inoltre non è detto che gli stessi dati siano sempre processati dalla stessa applicazione. Per esempio, oggi potrei voler usare un certo programma per gestire la mia posta elettronica, domani potrei decidere di usarne un altro, e non è sicuramente pensabile che si debba informare tutta la rete ogni volta che si decide di cambiare l'applicazione che gestisce un certo tipo di dati. Quindi, più che il processo, quello che è importante identificare è la funzione, come per esempio, trasferire file oppure spedire posta elettronica. Lo scopo dell'UDP è appunto quello di permettere di distinguere in un singolo host più destinatari per i dati che arrivano dalla rete. Ma come?

Facciamo un attimo una digressione. Se io devo stampare un file cosa faccio? Collego al mio computer una stampante, attivo il driver corrispondente, e uso una applicazione o un comando del sistema operativo per lanciare l'ordine di stampa. Se ora stacco la stampante e ne attacco un'altra alla stessa porta non devo far altro che cambiare il driver per continuare a lavorare senza che il sistema si sia accorto di niente. Se poi le due stampanti usano lo stesso driver generico devo solo staccare e riattaccare fisicamente le stampanti senza modificare il sistema. In generale, tutto lo scambio di dati da e verso un computer avviene attraverso porte di I/O. Ogni applicazione accede la porta che gli serve in modo dinamico. La periferica di I/O non ha bisogno di sapere con quale applicazione sta parlando: la porta fa da schermo fra i due. L'UDP fa una cosa analoga. Esso permette di associare a un indirizzo IP più punti di ingresso e di uscita virtuali, detti protocol port. Queste porte si comportano un po' come quelle di I/O di un computer. Ogni porta è identificata da un numero intero positivo e i processi possono chiedere al sistema l'accesso a tali porte. Quando un processo accede una porta, esso si mette in attesa dei dati sulla stessa. Il meccanismo è cioè sincrono. Inoltre, se dei dati arrivano a una porta alla quale non si è agganciato ancora un processo, questi vengono mantenuti in memoria nell'ordine di arrivo. Si dice cioè che le porte hanno un buffer. A questo punto, sia il processo mittente che quello destinatario sono univocamente identificati dall'indirizzo IP dell'host su cui girano e dal numero di porta che utilizzano per la trasmissione in rete. Tale associazione è tuttavia

dinamica, e così come più applicazioni possono stampare sulla stessa stampante purché non contemporaneamente, così più processi possono attaccarsi uno alla volta alla stessa porta ed essere visti come lo stesso destinatario dalla controparte mittente. Questo permette di spedire un file di testo con un word processor, e subito dopo spedire un file binario, per esempio un file ZIP, con un comando di sistema. Il tutto sempre attraverso la stessa porta e con lo stesso destinatario in termini di host e di processo.

Come abbiamo visto nel caso dell'IP e dei vari protocolli di rete, anche il datagramma UDP è formato da una intestazione (header) e da una parte dati. Esso è inoltre incapsulato in un datagramma IP il quale a sua volta è contenuto in un frame della rete fisica.

Al contrario tuttavia di quello IP, l'header UDP è molto più semplice. Esso è formato dal numero di porta del mittente, da quello del destinatario, dalla lunghezza del messaggio UDP, sia dei dati che dell'intestazione, e da una somma di controllo (checksum) per la verifica dell'integrità dei dati. Infatti, anche l'UDP, come l'IP, è un protocollo cosiddetto "inaffidabile". Questo vuol dire che un messaggio UDP può andare perso, essere duplicato, o arrivare nell'ordine sbagliato. L'UDP non fa alcun controllo al riguardo. L'affidabilità della comunicazione è infatti affidata a i protocolli di livello più elevato. Tutti i campi dell'intestazione sono lunghi 16 bit. Benché il formato del datagramma UDP sia alquanto semplice, la sua gestione può essere un po' più complessa. Il punto riguarda la somma di controllo. Questo valore è opzionale e, al contrario di quello che succedeva con la somma di controllo nel datagramma IP, esso non è relativo solo

all'intestazione ma a tutto il datagramma, compresa la parte dati. Questo vuol dire che tale campo rappresenta l'unico elemento di controllo a livello IP e UDP dell'integrità dei dati arrivati. Se esso non viene utilizzato, il campo va posto a zero. Questo in quanto la somma di controllo segue la logica a complemento uno. Il che vuol dire che se la somma calcolata è zero, essa può essere memorizzata come 16 bit impostati a uno, dato che in tale logica un valore con tutti i bit a uno e uno con tutti i bit a zero rappresentano lo stesso numero. Quindi: se il campo è a zero, vuol dire che non è stato utilizzato; se viceversa ha tutti i bit ad 1, vuol dire che la somma era zero.

La somma di controllo, tuttavia, per ragioni pratiche, non riguarda solo il datagramma UDP, ma viene calcolata anche utilizzando alcune informazioni aggiuntive. Queste formano il cosiddetto pseudo-header.

In pratica, quando si deve calcolare il checksum UDP si mette davanti al datagramma UDP un altro blocco di dati che contiene l'indirizzo IP del mittente e del destinatario, il codice del protocollo UDP (17), e la lunghezza del datagramma UDP. Il motivo di questa tecnica risiede nel fatto che, per verificare se un messaggio UDP è arrivato al giusto destinatario, non basta verificare il numero di porta, ma anche l'indirizzo IP dell'host in cui gira il processo che è collegato a quella porta. Il datagramma UDP contiene tuttavia solo il numero di porta, per cui il controllo fornito da una somma sul solo datagramma non potrebbe realmente verificare che il destinatario dei dati è quello giusto.

Questa tecnica ha tuttavia un prezzo: gli indirizzi IP fanno parte appunto del livello internet, non di quello di trasporto. Perché l'UDP conosca tali informazioni è necessario violare una legge fondamentale dei protocolli di comunicazione, e cioè che ogni livello della torre deve gestire i suoi dati senza esportarli agli altri livelli a cui offre, o da cui riceve, servizi. Questo vuol dire che la separazione tra UDP ed IP non è così pulita come dovrebbe essere, ma che i due protocolli sono in qualche modo legati tra loro. D'altra parte i vantaggi in termini di controllo e semplicità di implementazione sono tali che si è deciso di fare un'eccezione ai principi dell'architettura. Da notare che la pseudo-intestazione serve solo a calcolare la somma di controllo. Essa non viene fisicamente trasmessa con il datagramma UDP, dato che le informazioni che contiene fanno già parte dell'intestazione del datagramma IP in cui quello UDP sarà incapsulato.

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 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 rispezzettati 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 trenino 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.

Il controllo di flusso: Il controllo del flusso dei dati è un aspetto estremamente importante in un sistema in cui sono collegate macchine anche molto differenti fra loro per dimensioni e capacità di trasmissione. Per controllo del flusso si intende la possibilità di regolare dinamicamente la quantità di dati che vengono immessi nella rete. Non solo è importante che il destinatario possa regolare la velocità di spedizione in funzione della sua capacità di ricezione, ma è fondamentale che ogni gateway intermedio possa frenare il flusso dei dati che riceve per evitare di entrare in saturazione. Il meccanismo appena descritto della soglia di capacità permette di risolvere il primo problema, non il secondo. Quest'ultimo è detto congestione, ed è estremamente importante perché non tenerne conto vuol dire mandare in tilt la rete.

Lo standard TCP non prevede alcun meccanismo di controllo della congestione, lasciando agli implementatori di tale protocollo il non banale compito di sviluppare una logica capace di evitare questo tipo di problemi. Per quello che riguarda i segmenti, il fatto che TCP sia libero di dividere il flusso in segmenti può a volte causare problemi dal punto di vista applicativo. Per esempio, supponiamo di implementare via TCP/IP un terminale remoto. Questo vuol dire che tutte le operazioni effettuate con la tastiera e il mouse su di una macchina (chiamiamola locale) saranno visibili su di un'altra macchina (remota) come se esse fossero state effettuate dalla tastiera e dal mouse della macchina remota. Non solo: sarà possibile vedere lo schermo della macchina remota all'interno di una finestra della macchina locale. Questo tipo di applicazioni è molto utile per esempio se per un qualche motivo la macchina da controllare non ha

una sua tastiera oppure si trova in un locale non generalmente accessibile all'operatore. È evidente che affinché l'applicazione funzioni essa debba lavorare in tempo reale. Se cioè si preme il tasto T sulla tastiera locale, la lettera T deve apparire immediatamente sullo schermo della macchina remota, e quindi apparire anche all'interno della finestra locale che riproduce tale schermo. Lo stesso se si fa click sul pulsante di chiusura di una finestra. Ovviamente se TCP fosse libero di accumulare questi comandi per poi spedirli tutti in una volta l'applicazione sarebbe di difficile utilizzo. Infatti, se l'operatore decidesse di chiudere una finestra dello schermo remoto per accedere un'icona sottostante e TCP non spedisse il comando fintanto che il buffer di partenza non fosse pieno, non sarebbe possibile eseguire l'operazione successiva, cioè aprire l'icona sulla scrivania del sistema. Per questo motivo TCP prevede la possibilità di forzare la spedizione del buffer (push). Questo tuttavia non è sufficiente. Se infatti TCP che riceve i dati accumulasse gli stessi nel buffer di ricezione prima di passarli all'applicazione destinataria saremmo punto e da capo. Per questo motivo, quando un segmento è forzato in uscita, TCP imposta a uno un certo bit nell'intestazione del segmento in modo che questi possa venire riconosciuto e immediatamente passato all'applicazione remota.

Abbiamo detto che il TCP utilizza il metodo della finestra di scorrimento per tenere la rete sempre impegnata al massimo della sua capacità e che esiste un'importante differenza tra il meccanismo generale e quello più sofisticato utilizzato effettivamente dal TCP. Tale differenza consiste in un'asimmetria rispetto al meccanismo base dove l'unità dati utilizzata nella finestra di scorrimento era la stessa utilizzata nella trasmissione. Il TCP utilizza il segmento come unità dati da trasmettere, mentre ragiona in termini di ottetti per quello che riguarda il meccanismo di ritrasmissione. Questo comporta una complicazione nella gestione delle conferme di ricezione (acknowledgement).

A causa dell'asimmetria suddetta, la ritrasmissione in caso di mancata ricezione non avviene per segmenti, ma a livello di ottetti. Questo vuol dire che un segmento può contenere contemporaneamente sia nuovi dati sia una parte dei dati persi in precedenza. Ovviamente a queste condizioni ha poco senso numerare semplicemente i segmenti e usare questo identificativo nelle conferme di ricezione. Né è pensabile di usare i datagrammi IP a tale scopo, dato che questi sono generalmente di lunghezza fissa mentre i vari segmenti TCP sono di lunghezza variabile. Ne consegue che l'unico modo per gestire le conferme è quello di ragionare in termini di cursore all'interno del flusso di dati. Come dire "ho ricevuto i primi 300 caratteri della lettera che mi hai spedito".

Ecco che cosa accade: ogni segmento contiene la posizione dell'area dati del segmento TCP all'interno del flusso di dati. Tale posizione si chiama numero di sequenza (sequence number) ed è calcolata in byte. Il destinatario estrae i vari ottetti dai segmenti ricevuti e li ricomponde per ricostruire il flusso dei dati, utilizzando i numeri di sequenza per riordinare i vari segmenti.

Questi possono infatti arrivare in qualunque ordine, o essere andati persi. A questo punto, chi sta ricevendo i dati, avrà ricostruito in modo completo una parte del messaggio originario e si ritroverà alcuni dati in eccesso che non sono contigui alla parte di flusso ricostruito. Ogni volta che il destinatario riceve un segmento, manda indietro nella conferma di ricezione il numero di sequenza dell'ottetto che si aspetta di ricevere per continuare la ricostruzione, cioè il valore dell'ultimo ottetto della parte contigua ricostruita più uno.

Immaginate di dover spedire una lettera a un vostro amico. Il TCP negozia con la controparte la lunghezza massima del segmento, come vedremo più avanti. Quindi inizia a riempire il primo segmento un carattere alla volta. Quando il segmento è pieno viene spedito e viene fatto partire il contatore a tempo per quel segmento. Quindi il TCP inizia a riempire il secondo segmento, che parte regolarmente, e così dicendo. Man mano che i segmenti partono arrivano dalla controparte le conferme di ricezione. Supponiamo che a un certo punto, dopo aver spedito 450 ottetti, arrivi per due volte al mittente la conferma che il destinatario è riuscito a ricostruire il flusso fino al 300°

carattere e che si aspetta il 301°. È evidente che qualcosa è andato storto e che si sono persi dei dati. Il TCP allora spedisce un segmento che contiene di nuovo dal 301° carattere in poi, diciamo fino al 400°. Dato che i caratteri dal 370° al 450° erano comunque arrivati regolarmente, il successivo messaggio di conferma richiederà direttamente il 451° carattere, e non il 401°.

Vantaggi e svantaggi: Un vantaggio è che il valore di conferma è estremamente semplice da calcolare e di immediata comprensione. Inoltre, se una conferma di ricezione va persa, non è detto che questo causi automaticamente la ritrasmissione dei dati. Ci sarà comunque la conferma successiva che fornirà l'indicazione esatta del punto a cui è arrivato il destinatario nel ricostruire il flusso.

Lo svantaggio più grosso è che il mittente non ha modo di sapere quanti dati siano effettivamente arrivati con successo al destinatario, dato che basta un buco nel flusso per far segnalare come validi un numero di byte molto inferiore a quelli effettivamente ricevuti. Questo crea seri problemi al mittente, che non sa se ritrasmettere tutti i dati successivi, e quindi sprecare tempo a ritrasmettere dati già arrivati, o trasmettere solo una piccola parte e aspettare la conferma che il potenziale buco si è chiuso.

Entrambi gli schemi sono alquanto inefficienti. Sta allo sviluppatore dello stack TCP/IP decidere quali algoritmi utilizzare, tenendo presente che un algoritmo troppo complesso ha comunque lo svantaggio di avere potenzialmente basse prestazioni.

Un altro punto importante è il calcolo della lunghezza ottimale del segmento. Abbiamo detto più sopra che ogni conferma di ricezione contiene una soglia di capacità (window advertisement) la quale specifica il numero di ulteriori ottetti che il destinatario è in grado di ricevere. Questo meccanismo permette di adattare la finestra di spedizione alle dimensioni del buffer di ricezione. Tuttavia è anche necessario definire la lunghezza del segmento oltre che in funzione delle capacità di trasmissione del mittente e di ricezione del destinatario, anche e soprattutto in funzione delle caratteristiche della rete, come per esempio la grandezza massima del frame fisico, o Maximum Transfer Unit (MTU). La lunghezza massima di un segmento, o Maximum Segment Size (MSS), viene calcolata appunto sulla base dell'MTU se entrambi gli estremi della connessione si trovano nella stessa rete fisica, altrimenti lo standard raccomanda di utilizzare un valore di 536 byte, equivalente alla dimensione normale di un datagramma IP meno le dimensioni standard delle intestazioni IP e TCP sommate insieme, 40 byte appunto.

Tale calcolo è ovviamente solo un primo tentativo di ottimizzare l'utilizzo della rete da parte del TCP. Durante la trasmissione il TCP può modificare tale valore in funzione della situazione contingente. Non esiste tuttora un algoritmo standard per definire il giusto valore per l'MSS, data la complessità del problema. Una cattiva definizione dell'MSS può seriamente penalizzare la comunicazione. Se il segmento è troppo piccolo, il rapporto tra i dati trasmessi e quelli utilizzati nella trasmissione stessa è sfavorevole. Per esempio, un segmento di cinque byte utilizza solo un ottavo della larghezza di banda (bandwidth) disponibile, dato che per ogni cinque byte di dati ce ne sono ben quaranta di intestazione. Viceversa, se il segmento è molto grande, altrettanto è il datagramma IP. Se il datagramma è più grande dell'MTU, verrà spezzato in più frammenti non indipendenti fra loro, per cui basta che si perda un solo frammento per perdere tutto il datagramma e di conseguenza il segmento TCP.

Il controllo di flusso: Il controllo del flusso dei dati è un aspetto estremamente importante in un sistema in cui sono collegate macchine anche molto differenti fra loro per dimensioni e capacità di trasmissione. Per controllo del flusso si intende la possibilità di regolare dinamicamente la quantità di dati che vengono immessi nella rete. Non solo è importante che il destinatario possa regolare la velocità di spedizione in funzione della sua capacità di ricezione, ma è fondamentale che ogni gateway intermedio possa frenare il flusso dei dati che riceve per evitare di entrare in saturazione. Il meccanismo descritto della soglia di capacità permette di risolvere il primo problema ma non il secondo. Quest'ultimo

è detto congestione, ed è estremamente importante perché non tenerne conto vuol dire mandare in tilt la rete.

Lo standard TCP non prevede alcun meccanismo di controllo della congestione, lasciando agli implementatori di tale protocollo il non banale compito di sviluppare una logica capace di evitare questo tipo di problemi.

Per quello che riguarda i segmenti, il fatto che il TCP sia libero di dividere il flusso in segmenti può a volte causare problemi dal punto di vista applicativo. Per esempio, supponiamo di implementare via TCP/IP un terminale remoto. Questo vuol dire che tutte le operazioni effettuate con la tastiera e il mouse su di una macchina (chiamiamola locale) saranno visibili su di un'altra macchina (remota) come se esse fossero state effettuate dalla tastiera e dal mouse della macchina remota. Non solo: sarà possibile vedere lo schermo della macchina remota all'interno di una finestra della macchina locale. Questo tipo di applicazioni è molto utile per esempio se per un qualche motivo la macchina da controllare non ha una sua tastiera oppure si trova in un locale non generalmente accessibile all'operatore. È evidente che affinché l'applicazione funzioni essa debba lavorare in tempo reale. Se cioè si preme il tasto T sulla tastiera locale, la lettera T deve apparire immediatamente sullo schermo della macchina remota, e quindi apparire anche

all'interno della finestra locale che riproduce tale schermo. Lo stesso se si fa click sul pulsante di chiusura di una finestra. Ovviamente se il TCP fosse libero di accumulare questi comandi per poi spedirli tutti in una volta l'applicazione sarebbe di difficile utilizzo. Infatti, se l'operatore decidesse di chiudere una finestra dello schermo remoto per accedere un'icona sottostante e il TCP non spedisse il comando fintanto che il buffer di partenza non fosse pieno, non sarebbe possibile eseguire l'operazione successiva, cioè aprire l'icona sulla scrivania del sistema. Per questo motivo il TCP prevede la possibilità di forzare la spedizione del buffer (push). Questo tuttavia non è sufficiente. Se infatti il TCP che riceve i dati accumulasse gli stessi nel buffer di ricezione prima di passarli all'applicazione destinataria saremmo punto e da capo. Per questo motivo, quando un segmento è forzato in uscita, il TCP imposta a uno un certo bit nell'intestazione del segmento in modo che questi possa venire riconosciuto e immediatamente passato all'applicazione remota. Esiste poi la possibilità che il TCP debba spedire dei dati che non fanno parte del flusso normale e che vanno immediatamente gestiti dalla controparte indipendentemente dallo stato in cui si trova la ricostruzione del messaggio originario. Tali dati sono detti urgenti, e anche in questo caso esiste un bit nell'intestazione del segmento che informa il destinatario del fatto che il segmento va gestito immediatamente. Il concetto è analogo a quello del BREAK da tastiera. Se avete lanciato un

programma che va in loop è necessario poterlo interrompere senza dover far ripartire il sistema. Su molti sistemi operativi basta premere una sequenza di tasti, come per esempio Control-C (^C) per bloccare l'esecuzione del programma. Similarmente, se un estremo della connessione deve bloccare (o sbloccare) l'elaborazione del flusso di dati dall'altra parte, dovrà poter mandare un messaggio urgente che abbia la precedenza rispetto ai normali segmenti di dati. Si dice che tale messaggio è fuori banda (out of band).

Benché il TCP presenti all'utente una visione continua dei dati, detta flusso, l'unità di trasferimento dei dati del TCP è il segmento. Un segmento è formato come al solito da una intestazione e da un'area dati. Al contrario del datagramma IP, il segmento ha dimensioni variabili nel tempo, cioè i vari segmenti spediti a fronte di uno stesso flusso possono avere lunghezze differenti. I segmenti sono utilizzati dal TCP per aprire e chiudere una connessione, trasferire dati, spedire conferme di ricezione e modificare la finestra di spedizione, quel meccanismo che garantisce un utilizzo ottimale della rete, come spiegato in precedenza. Due caratteristiche peculiari del TCP sono che lo stesso segmento può portare contemporaneamente sia dati veri e propri sia dati di controllo, e che le informazioni di controllo possono riferirsi sia allo stesso flusso dell'area dati, sia al flusso opposto (piggybacking).

L'intestazione: Innanzitutto abbiamo i numeri di porta del mittente e del destinatario, esattamente come nell'UDP. Come già nell'UDP, infatti, gli indirizzi IP delle due controparti sono contenuti nell'intestazione del datagramma IP. Al contrario di quanto avveniva nell'UDP, tuttavia, la conoscenza da parte del TCP degli indirizzi IP non rompe il paradigma che vuole un certo isolamento fra le responsabilità dei vari livelli dello stack. Il TCP infatti, architetturealmente, ragiona in termini di connessioni, e queste comprendono sia l'informazione relativa alle porte, sia quella relativa agli indirizzi IP. Anzi, ogni qual volta l'IP consegna un segmento al TCP, gli passa anche gli indirizzi IP contenuti nell'intestazione del datagramma.

Anche nel caso del segmento TCP la verifica della correttezza dell'intestazione da parte del destinatario viene effettuata utilizzando un meccanismo di somma di controllo con pseudointestazione. All'interno dell'intestazione TCP, infatti, esiste un campo chiamato somma di controllo (checksum). Il TCP imposta inizialmente tale campo di 16 bit a zero. Costruisce quindi una pseudointestazione che contiene gli indirizzi IP del mittente e del destinatario, il numero di protocollo del sottosistema di trasmissione (nel caso del TCP basato su IP è sei) e la lunghezza del segmento TCP compresa l'intestazione. A questo punto appende alla pseudointestazione il segmento IP e aggiunge alla fine dello stesso tanti zeri quanti ne servono per far sì che il blocco risultante sia un multiplo di parole da 16 bit (padding). Divide quindi il blocco in parole da 16 bit e ne calcola la somma a complemento uno. Il risultato viene quindi salvato nel campo apposito dell'intestazione e sia la pseudointestazione sia i bit aggiunti in fondo vengono rimossi prima di spedire il segmento. Il destinatario ovviamente effettuerà un calcolo analogo per verificare che il valore di controllo così ottenuto corrisponda con quello arrivato nell'intestazione del segmento.

Nell'intestazione ci sono tre campi calcolati in ottetti. Il primo è il numero di sequenza (sequence number), che rappresenta la posizione dell'area dati del segmento TCP all'interno del flusso di dati. Il secondo è il numero di conferma (acknowledgement number), ovvero sia il numero di sequenza dell'ottetto che il mittente si aspetta di ricevere per continuare la ricostruzione. Da notare che tale valore corrisponde al flusso opposto rispetto a quello in cui viaggia il segmento che lo contiene. Il terzo campo è il puntatore ai dati "urgenti". Come detto prima, è possibile che il TCP debba spedire dei dati urgenti che vanno elaborati indipendentemente dal flusso normale di dati, e con priorità rispetto a quest'ultimo. In questo caso il segmento contiene un segnalatore (flag) che informa il destinatario della presenza d'informazioni urgenti nell'area dati. I dati urgenti sono posizionati all'inizio dell'area dati, e il puntatore in questione indica dove tali dati finiscono e dove ricominciano i dati normali, se ce ne sono. I segnalatori: Separati da un'area riservata per usi futuri c'è il campo che contiene la posizione dell'area dati nel segmento e un blocco di sei segnalatori. Il primo, misurato in parole da 32 bit, indica di fatto la lunghezza dell'intestazione del segmento in tale unità di misura. questo campo è necessario in quanto in fondo all'intestazione esiste una zona riservata a eventuali opzioni che rende la lunghezza dell'intestazione non fissata a priori. Il secondo campo contiene invece sei indicatori. Data infatti nel segmento la presenza contemporanea, almeno in potenza, sia di dati di controllo sia di dati applicativi normali e urgenti, è necessario utilizzare dei segnalatori per informare il destinatario su cosa effettivamente contiene il segmento. Tutti i segnalatori sono attivi se impostati a uno, inattivi altrimenti. Il primo segnalatore indica se l'area dati contiene dati urgenti. Il secondo indica la presenza nel segmento di una conferma di ricezione valida. Dato infatti che il campo corrispondente esiste sempre e comunque nell'intestazione, se il segmento non trasporta alcuna conferma di ricezione è necessario informare in qualche modo il destinatario che tale campo va ignorato. Il terzo bit è posto a uno quando si vuole forzare la trasmissione dei dati all'utente finale indipendentemente dal fatto che il buffer di ricezione sia o meno completamente riempito. Il quarto segnalatore serve per interrompere immediatamente la connessione (reset). Tale evento avviene solo in situazioni

eccezionali e causa l'interruzione immediata delle trasmissioni da ambo le parti e il rilascio del contenuto dei buffer di ricezione. Il quinto bit è detto di sincronizzazione, ed è utilizzato durante la fase iniziale di negoziazione della connessione, detta in gergo handshake. In pratica, i segmenti scambiati quando questo bit è impostato a uno servono a sincronizzare i numeri di sequenza delle due controparti prima d'iniziare la trasmissione vera e propria dei dati. L'ultimo bit serve a informare il destinatario che il mittente intende terminare in modo pulito la connessione e che non ha più dati da spedire. All'apertura e alla chiusura della connessione il TCP utilizza un algoritmo chiamato saluto a tre vie (three-way handshake) che garantisce la corretta sincronizzazione delle due operazioni. L'ultimo campo fisso è quello relativo alla soglia di capacità del mittente (window advertisement) che contiene il numero di ulteriori ottetti che esso è in grado di ricevere. A questo punto è di nuovo importante ricordare il concetto di piggybacking, a cui già si è accennato. Ovverosia, ogni segmento può portare contemporaneamente informazioni in cui una controparte è vista sia come chi spedisce i dati contenuti nel segmento, cioè come mittente, sia come chi ha ricevuto o deve ricevere dati dall'altro capo della connessione, cioè come destinatario. Quando noi parliamo di mittente, per evitare confusione, ci riferiamo sempre al mittente del segmento di cui stiamo parlando. È importante comunque tenere sempre presente che alcuni dati del segmento hanno senso solo se si considera il mittente quale destinatario di dati precedenti o ancora da venire. In fondo all'intestazione c'è un'area opzionale che può essere utilizzata a vari scopi. In genere essa contiene opzioni che permettono alle due controparti di negoziare alcuni aspetti della comunicazione. Un esempio è il calcolo della lunghezza ottimale del segmento.

L'implementazione del protocollo TCP: Abbiamo visto che tutto il meccanismo funziona ed è affidabile grazie alle conferme di ricezione e alla ritrasmissione dei pacchetti probabilmente andati perduti. Ma come fa a sapere il mittente che un pacchetto è andato effettivamente perduto? Ovviamente perché non è arrivata la conferma di ricezione, direte voi. Va bene, ma quanto devo aspettare tale conferma prima di assumere che sia necessaria una ritrasmissione? E qui son dolori. Se aspetto troppo rischio di rallentare la comunicazione in modo inaccettabile. Se aspetto troppo poco rischio di ritrasmettere inutilmente troppi segmenti, magari semplicemente un po' in ritardo. Tutto il sistema si basa sul calcolo del tempo di attesa massimo, o timeout. Il TCP calcola il timeout sulla base del tempo intercorso fra la spedizione di un segmento e l'arrivo della conferma corrispondente. Sembra facile, ma non è così. Vediamo rapidamente i punti chiave del discorso.

Il TCP calcola continuamente il timeout, ogni volta che arriva una conferma di ricezione. In questo modo il sistema è sempre aggiornato in funzione dello stato effettivo della connessione e della rete.

Il timeout è calcolato come media pesata dei tempi intercorsi fra la spedizione del segmento e la ricezione della conferma. Chiamiamo quest'ultimo tempo rilevato di andata e ritorno (Round Trip Sample) o RTS. Il tempo stimato di andata e ritorno (Round Trip Time) è calcolato utilizzando una specifica formula. In pratica ogni nuovo RTS pesa più o meno sul calcolo dell'RTT in base al valore di alfa. Se alfa è molto vicina a zero, l'RTT varia rapidamente a ogni cambiamento dell'RTS, per cui il sistema risponde rapidamente alle variazioni. Se viceversa alfa è vicina a uno, è necessario che la nuova RTS rimanga stabile più a lungo per avere effetto sull'RTT.

A questo punto il timeout viene calcolato moltiplicando l'RTT per un valore maggiore di uno. Se il valore di beta è troppo vicino a uno la perdita di un pacchetto viene immediatamente rilevata, ma si rischia di ritrasmettere più pacchetti del necessario. Se viceversa beta è troppo alto si rischia di aspettare troppo a lungo prima di ritrasmettere un pacchetto perso, abbassando così le prestazioni della connessione. In genere si raccomanda per beta un valore di due.

Una scelta difficile: La scelta di alfa e di beta sembra dunque essere critica, ma i problemi non sono ancora finiti. Infatti, se un segmento è trasmesso due volte, quando arriva la conferma di ricezione, a chi si riferisce? Al pacchetto originale o a quello ritrasmesso? Se usiamo il primo pacchetto per il calcolo dell'RTS rischiamo di far crescere esponenzialmente il valore di timeout. Infatti un pacchetto è ritrasmesso quando scade il timeout precedente. Di conseguenza il nuovo RTS è ovviamente più grande del vecchio timeout. Se viene perso un nuovo pacchetto l'RTS cresce ancora, e così via. Se usiamo il pacchetto ritrasmesso abbiamo il problema opposto, cioè il timeout rischia di ridursi sempre di più, o almeno si è dimostrato sperimentalmente che si stabilizza su valori alquanto bassi. Supponiamo infatti di avere un ritardo in rete: la conferma di ricezione arriva conseguentemente in ritardo. Nel frattempo il mittente ha rispedito il pacchetto che credeva perso. Appena arriva la conferma questa è associata al segmento ritrasmesso generando così un RTS molto piccolo. Il timeout si riduce, aumentando il rischio di considerare persi pacchetti la cui conferma di ricezione arriva in ritardo, e così via.

P. Karn propose nel 1987 d'ignorare i pacchetti ritrasmessi nel calcolo del timeout. Questo evitava il problema suddetto, ma ne creava un altro. Se un pacchetto è ritrasmesso perché si è avuto un repentino calo di prestazioni della rete, il timeout rimarrà sempre troppo basso, in quanto il mittente continuerà a ritrasmettere pacchetti le cui conferme arrivano in ritardo rispetto al timeout calcolato prima del calo di prestazioni. Dato che tali conferme vengono regolarmente ignorate per il calcolo dell'RTT, il timeout non viene più aggiornato almeno fintanto che la rete non torna normale, cosa per giunta complicata dal sovraccarico dovuto all'inutile ritrasmissione dei pacchetti. La soluzione consiste nell'aumentare il timeout precedente a una ritrasmissione di un fattore arbitrario, diciamo gamma, fino a un limite massimo ragionevole calcolato sulla base dei possibili cammini all'interno della rete. In genere gamma non è minore di due. Questa tecnica è detta di backoff.

Conclusione: Implementare il protocollo TCP non è certo banale. Il che tra l'altro fa capire come non tutti i pacchetti TCP siano uguali: anzi, è proprio il contrario. Una scelta oculata degli algoritmi implementativi può fare seriamente la differenza fra un prodotto e un altro. Il fatto che essi implementino lo stesso standard non dà alcuna indicazione sulla qualità delle prestazioni dello stack che state utilizzando. Se poi alcuni parametri possono essere personalizzati dall'utente, una opportuna calibrazione del programma studiata sulle caratteristiche specifiche della vostra rete, può modificare significativamente i tempi di risposta del sistema. Naturalmente non è fra gli scopi di questi articoli entrare nel dettaglio di tutte le problematiche TCP/IP.

Autore: Maiuscolo.net

Guida scaricata dal sito: <http://www.maiuscolo.net>