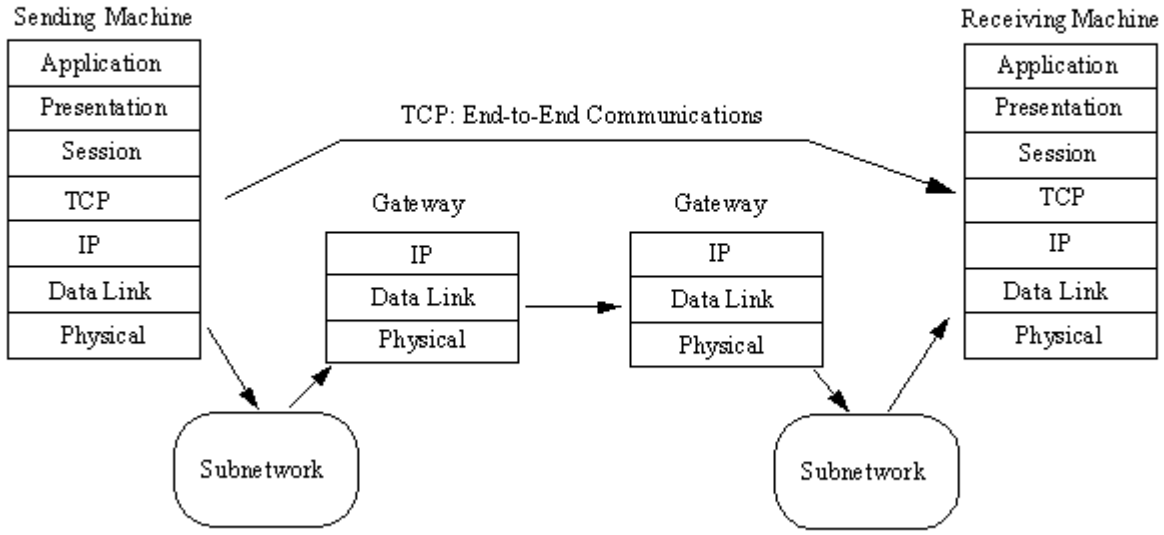
Cos’è TCP

IL Transmission Control Protocol fornisce un considerevole numero di sevizi allo strato IP e agli strati superiori. La cosa più importante è che esso fornisce un protocollo orientato alla connessione agli strati superiori che permette ad un’applicazione ei essere sicura che un datagramma inviato in rete sia stato ricevuto nella sua interezza. In questo ruolo, TCP agisce come un protocollo di validazione dei messaggi che fornisce comunicazioni affidabili. Se un datagramma è corrotto o perso , TCP usualmente gestisce la ritrasmissione.

TCP gestisce il flusso di datagrammi dagli strati superiori allo strato IP, cosi come il flusso di datagrammi in ingresso dallo strato IP agli strati superiori. TCP deve assicurare che le priorità e la sicurezza siano rispettate in maniera appropriata. TCP deve essere capace di gestire la terminazione di un’applicazione sopra di esso che stava aspettando datagrammi in arrivo, cosi come il fallimento degli strati inferiori. TCP, inoltre, deve gestire una tabella di stato di tutti i flussi di dati in ingresso ed in uscita . l’isolamento di tutti questi servizi in uno strato separato consente alle applicazioni di essere sviluppate senza alcuna preoccupazioni per la gestione dei flussi e l’affidabilità della trasmissione. Senza lo strato TCP tutte le applicazioni dovrebbero essere esse stesse ad implementare queste servizi, il che costituirebbe uno spreco di risorse.

TCP risiede nel livello di trasporto , posizionato sopra lo strato IP ma sotto gli strati superiori e le loro applicazioni, come mostrato nella figura seguente.



TCP risiede soltanto sulle macchine che processano realmente i datagrammi, assicurando che essi giungano dalla sorgente alla macchina di destinazione. Esso non risiede su un'apparecchiatura che semplicemente instrada i datagrammi, così non vi è strato TCP in un gateway. Ciò ha senso, poiché in un gateway i pacchetti non hanno bisogno di salire verso gli strati superiori allo strato IP.

Poiché il TCP è un protocollo orientato alla connessione responsabile per il corretto passaggio del pacchetto dalla sorgente alla destinazione (comunicazione end-to-end), esso deve ricevere messaggi dalla macchina di destinazione per essere avvertito della corretta ricezione dello stesso. Il termine circuito virtuale è usualmente utilizzato per far riferimento alla comunicazione che avviene fra le due macchine terminali.

Seguire un messaggio

Per illustrare il ruolo del protocollo TCP, è istruttivo seguire un messaggio campione fra due macchine. A questo stadio semplifichiamo i vari passaggi. Il messaggio ha origine da un'applicazione in uno strato superiore ed è passato allo strato TCP dallo strato superiore più prossimo nell'architettura tramite qualche protocollo (spesso denominato Upper Layer Protocol o ULP). Il messaggio è passato come un flusso o sterai, cioè una sequenza di caratteri individuali inviati in maniera asincrona. Ciò è in contrasto con la maggior part3 dei protocolli, che usano blocchi fissi di dati. Ciò può porre qualche problema di conversione con applicazioni che usano soltanto blocchi di dati costruiti in maniera formale o insistono su messaggi di ampiezza fissa.

TCP riceve il flusso di dati e li assembla in segmenti TCP, o pacchetti. Nel processo di assemblaggio sono inserite informazioni di header in testa al pacchetto. Per ogni pacchetto è calcolato un checksum che viene inserito nell'header, così come un numero di sequenza se vi sono più segmenti nell'intero messaggio. La lunghezza del segmento è normalmente determinata dal protocollo TCP o da un valore di sistema settato dall'amministratore di sistema. (la lunghezza dei segmenti TCP non ha nulla a che fare con la lunghezza dei datagrammi IP anche se spesso vi è una relazione fra i due)

Se sono richieste comunicazioni in entrambi i sensi , una connessione (circuito virtuale) è creata fra le due macchine prima di passare i segmenti allo strato IP per il loro instradamento. Questo processo parte con il software TCP mittente che invia una richiesta di connessione TCP alla macchina destinataria. Nel messaggio vi è un

numero univoco (chiamato numero socket) che identifica la connessione della macchina mittente. Il software TCP ricevente assegna il suo numero univoco socket e lo invia alla macchina mittente. I due numeri univoci individuano la connessione fra le due macchine finché il circuito terminale non cessi di esistere.

Dopo che è stato stabilito il circuito virtuale, TCP manda il messaggio allo strato IP, che poi lo invia sotto forma di datagramma lungo la rete. Tutti i passaggi effettuati da IP come la frammentazione, il riassemblaggio sono completamente trasparenti allo strato TCP. In ricezione lo strato IP passa il segmento ricevuto allo strato TCP, dove esso è processato e inviato agli strati superiori.

Se il messaggio è costituito da più segmenti TCP , 9il sofware TCP ricevente riassembla il messaggio utilizzando i numeri di sequenza . se un segmento manca o è corrotto , TCP invia un messaggio contenente il numero di sequenza incriminato al mittente.

Se viene utilizzato un solo segmento per l'intero messaggio , dopo aver confrontato il checksum con il valore calcolato nella macchina ricevente, il sofware TCP ricevente può generare un messaggio di acknowledgment o una richiesta di rinviare il segmento.

L'implementazione TCP della macchina ricevente può effettuare un semplice controllo del flusso per prevenire sovraccarichi del buffer. Fa questo inviando un ampiezza del buffer chiamata valore di finestra alla macchina mittente , seguendo la quale il mittente può inviare soltanto abbastanza byte per riempire la finestra. Dopo di ciò il mittente deve attender un altro valore di finestra . ciò fornisce un protocollo

handshaking fra le due macchine , sebbene rallenti la comunicazione e incrementi leggermente il traffico lungo la rete.

Come nella maggior parte dei protocolli orientati alla connessione , i timer sono un aspetto importante del protocollo TCP. L'uso di un timer assicura che non si abbiano inutili attese mentre si attende un ACK o un messaggio di errore. Se i timer si esauriscono , si presume che vi sia stata una trasmissione incompleta. Usualmente l'esaurimento di un timer prima dell'invio di un messaggio ACK causa una ritrasmissione del messaggio dalla macchina mittente.

I timer possono causare alcuni problemi con il protocollo TCP. Le specifiche del protocollo TCP prevedono l'acknowledgment soltanto del numero di datagramma più alto che è stato ricevuto senza errore, ma questo non può gestire in maniera appropriata una ricezione frammentaria. Se un messaggio si compone di diversi datagrammi che non arrivano in ordine, le specifiche prevedono che lo strato TCP non può inviare il messaggio di acknowledgment della ricezione del messaggio finché non sono stati ricevuti tutti i datagrammi. Così se venissero ricevuti tutti i datagrammi tranne uno, un timer potrebbe esaurirsi e causare il rinvio di tutto il messaggio, di tutti i datagrammi. Con messaggi grandi ciò provocherebbe un incremento del traffico in rete.

Se il software ricevente TCP riceve datagrammi duplicati (come accade per una ritrasmissione dopo un timeout o una trasmissione duplicata dallo strato IP), lo strato ricevente TCP scarta i datagrammi duplicati, senza preoccuparsi di inviare messaggi

di errore. Dopo tutto, il sistema mittente si preoccupa solo che il messaggio venga ricevuto – non di quante copie siano state inviate.

TCP non ha una funzione di negative acknowledgment (NAK), si affida ad un timer per indicare la mancanza di riscontro. Se il timer si è esaurito dopo l'invio del messaggio senza che vi sia stato riscontro del corretto ricevimento, si assume che il messaggio sia stato perso e viene ritrasmesso. Il sofware TCP mittente mantiene copie di tutti i datagrammi che non hanno ricevuto riscontro in un buffer finché non vi sia stato un riscontro positivo. Quando ciò avviene, il timer di ritrasmissione viene fermato e il datagramma è rimosso dal buffer.

Il protocollo TCP prevede una funzione di push dai protocolli dei livelli superiori. Un push è utilizzato quando un'applicazione vuole inviare dati immediatamente e conferma che un messaggio passato al livello TCP è stato trasmesso con successo. Per fare ciò, un flag di push è settato nella connessione ULP, istruendo il livello TCP ad inviare ogni informazione nel buffer alla destinazione non appena possibile.

Porte e Socket

Tutte le applicazioni di livello superiore che usano TCP o UDP hanno un numero di porta che identifica l'applicazione. In teoria, i numeri di porta possono essere assegnati sulle singole macchine , o comunque l'amministratore di sistema desideri, ma sono state adottate alcune convenzioni per abilitare una comunicazione migliore fra le implementazioni TCP. Ciò abilita il numero di porta ad identificare il tipo di servizio che un sistema TCP sta richiedendo da un latro. I numeri di porta possono

essere cambiati, sebbene questo possa provocare difficoltà. La maggior parte dei sistemi mantiene un file dei numeri di porta e dei loro servizi corrispondenti.

Tipicamente i numeri di porta sopra al 255 sono riservati per usi privati sulla macchina locale , ma i numeri al di sotto del 255 sono usati per processi frequentemente utilizzati. Una lista dei numeri di porta frequentemente utilizzati è pubblicata dall'Internet Assigned Numbers Authority. I numeri di porta comunemente utilizzati su questa lista sono i seguenti. I numeri 0 e 255 sono riservati.

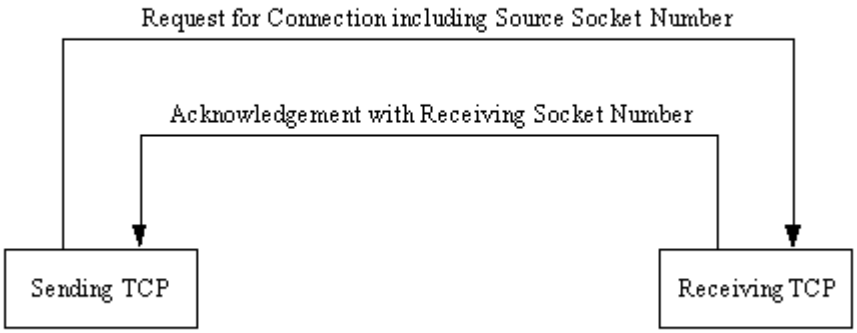
Port Number	Process Name	Description
1	TCPMUX	TCP Port Service Multiplexer
5	RJE	Remote Job Entry
7	ECHO	Echo
9	DISCARD	Discard
11	USERS	Active Users
13	DAYTIME	Daytime
17	Quote	Quotation of the Day
19	CHARGEN	Character generator
20	FTP-DATA	File Transfer Protocol•Data
21	FTP	File Transfer Protocol•Control
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer Protocol
27	NSW-FE	NSW User System Front End
29	MSG-ICP	MSG-ICP
31	MSG-AUTH	MSG Authentication

33	DSP	Display Support Protocol
35		Private Print Servers
37	TIME	Time
39	RLP	Resource Location Protocol
41	GRAPHICS	Graphics
42	NAMESERV	Host Name Server
43	NICNAME	Who Is
49	LOGIN	Login Host Protocol
53	DOMAIN	Domain Name Server
67	BOOTPS	Bootstrap Protocol Server
68	BOOTPC	Bootstrap Protocol Client
69	TFTP	Trivial File Transfer Protocol
79	FINGER	Finger
101	HOSTNAME	NIC Host Name Server
102	ISO-TSAP	ISO TSAP
103	X400	X.400
104	X400SND	X.400 SND
105	CSNET-NS	CSNET Mailbox Name Server
109	POP2	Post Office Protocol v2
110	POP3	Post Office Protocol v3
111	RPC	Sun RPC Portmap
137	NETBIOS-NS	NETBIOS Name Service
138	NETBIOS-DG	NETBIOS Datagram Service
139	NETBIOS-SS	NETBIOS Session Service
146	ISO-TP0	ISO TP0
147	ISO-IP	ISO IP

150	SQL-NET	SQL NET
153	SGMP	SGMP
156	SQLSRV	SQL Service
160	SGMP-TRAPS	SGMP TRAPS
161	SNMP	SNMP
162	SNMPTRAP	SNMPTRAP
163	CMIP-MANAGE	CMIP/TCP Manager
164	CMIP-AGENT	CMIP/TCP Agent
165	XNS-Courier	Xerox
179	BGP	Border Gateway Protocol

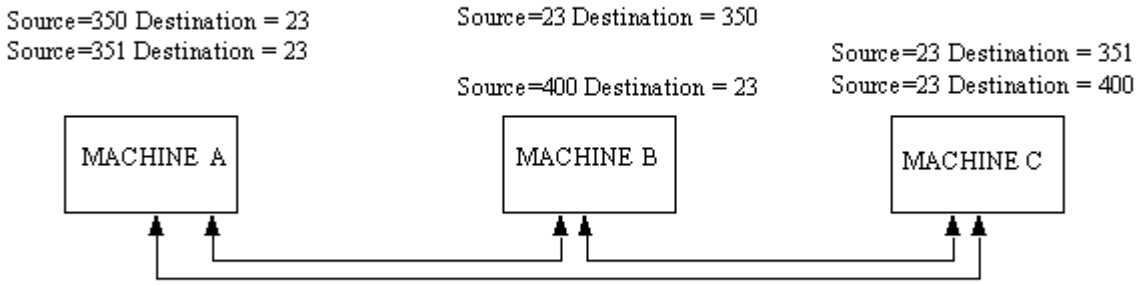
Ogni circuito di comunicazione in ingresso ed in uscita dallo strato TCP è identificato in maniera univoca da una combinazione di due numeri , che insieme prendono il nome di socket. Il socket è composto dell'indirizzo IP della macchina e del numero di porta usato dal sofware sia la macchina mittente che quella ricevente hanno un socket. Poiché l'indirizzo IP è unico lungo la rete, ei numeri di porta sono univoci su ogni singola macchina, i numeri di socket sono anch'essi univoci lungo l'intera rete. Ciò permette ad un processo di parlare con un altro processo lungo la rete, basato sui socket.

La figura seguente mostra il processo secondo il quale lo strato TCP della macchina mittente invia una richiesta di connessione con lo strato TCP della macchina ricevente usando i numeri socket.



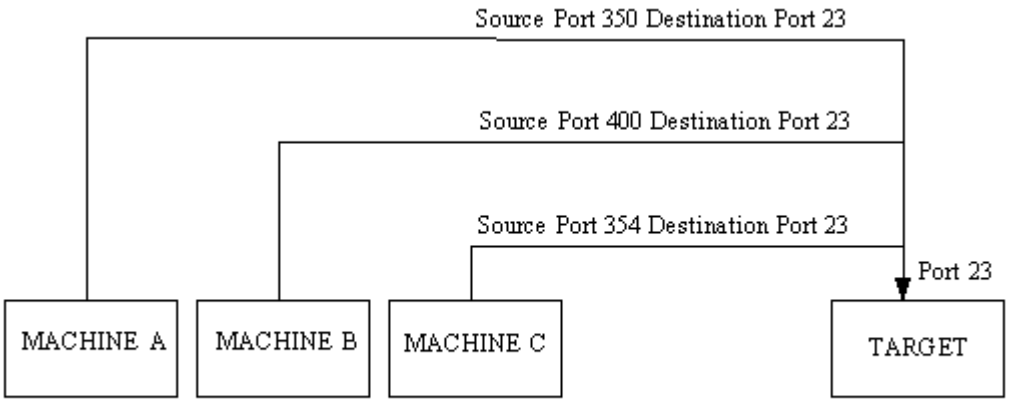
se il TCP mittente vuole stabilire una sessione Telnet dalla sua porta numero 350, il numero di socket sarà composto dall'indirizzo IP della macchina e dal numero di porta 350, ed il messaggio avrebbe un numero di porta destinazione 23 (porta del Telnet). Lo strato TCP ricevente ha una porta sorgente 23 e una porta di destinazione 350.

Le macchine mittente e ricevente mantengono una tabella delle porte , che elenca i numeri di tutte le porte attive. Le due macchine coinvolte hanno registrazioni riservate per ogni sessione fra le due. Questo è chiamato binding come mostrato in figura seguente. I numeri di sorgente e destinazione sono semplicemente invertiti per ogni connessione nella tabella delle porte. Naturalmente gli indirizzi IP e quindi i numeri socket sono differenti.



se la macchina trasmittente sta richiedendo più di una connessione, i numeri di porta sorgente sono differenti, sebbene i numeri di porta destinazione potrebbero essere gli stessi. Per esempio, se la macchina trasmittente stava tentando di stabilire tre sessioni Telnet contemporaneamente, i numeri di porta della macchina sorgente potrebbero essere 350, 351 e 352 e i numeri delle porte destinazione sarebbero tutti 23.

E' possibile per più di una macchina condividere lo stesso socket di destinazione - un processo chiamato multiplexing. Nella figura seguente tre macchine stanno stabilendo una sessione Telnet con una destinazione. Tutte usano la porta di destinazione 23. poiché i datagrammi che emergono dalla porta hanno la informazione socket completa (con l'indirizzo IP univoco), non vi è confusione circa la macchina cui è destinato il datagramma.



quando vengono stabiliti socket multipli , è teoricamente possibile che più di una macchina potrebbe inviare una richiesta di connessione con le stesse porte di sorgente e destinazione. Comunque, l'indirizzo IP per le due macchine è differente, cosicché i socket sono ancora identificati in maniera univoca nonostante abbiano numeri di porta identici.

Comunicazione dello strato TCP con gli strati superiori

TCP deve comunicare con le applicazioni dello strato superiore e il sistema di rete nello strato inferiore. Diversi messaggi sono stati ideati per la comunicazione dal protocollo dello strato superiore verso il TCP, ma non vi sono metodi definiti per la comunicazione fra TCP e gli strati inferiori (usualmente ma non necessariamente IP). TCP si aspetta che siano gli strati inferiori a definire i metodi di comunicazione. Si assume usualmente che TCP e gli strati inferiori comunichino in maniera sincrona.

Il metodo di comunicazione fra TCP e il protocollo dello strato superiore (ULP) è ben definito, e consiste di un set di primitive di richiesta di servizi. Esse sono illustrate nella tabella seguente.

Command	Parameters Expected
	ULP to TCP Service Request Primitives
ABORT	Local connection name
ACTIVE-OPEN	Local port, remote socket
	Optional: ULP timeout, timeout action, precedence, security, options
ACTIVE-OPEN-WITH-DATA	Source port, destination socket, data, data length, push flag, urgent flag
	Optional: ULP timeout, timeout action, precedence, security
ALLOCATE	Local connection name, data length
CLOSE	Local connection name
FULL-PASSIVE-OPEN	Local port, destination socket
	Optional: ULP timeout, timeout action, precedence, security, options
RECEIVE	Local connection name, buffer address, byte count, push flag, urgent flag
SEND	Local connection name, buffer address, data length, push flag, urgent flag
	Optional: ULP timeout, timeout action
STATUS	Local connection name
UNSPECIFIED-PASSIVE-OPEN	Local port
	Optional: ULP timeout, timeout action, precedence, security, options
	TCP to ULP Service Request Primitives

CLOSING	Local connection name
DELIVER	Local connection name, buffer address, data length, urgent flag
ERROR	Local connection name, error description
OPEN-FAILURE	Local connection name
OPEN-ID	Local connection name, remote socket, destination address
OPEN-SUCCESS	Local connection name
STATUS RESPONSE	Local connection name, source port, source address, remote socket, connection state, receive window, send window, amount waiting ACK, amount waiting receipt, urgent mode, precedence, security, timeout, timeout action
TERMINATE	Local connection name, description

Porte passive e attive

TCP consente due metodi per stabilire una connessione: passivo e attivo. Si instaura una connessione attiva quando TCP inoltra una richiesta di connessione , basata su una istruzione di un protocollo di strato superiore che fornisce il numero socket. Un approccio passivo si ha quando il protocollo di livello superiore dice a TCP di attendere l'arrivo di richieste di connessione da un sistema remoto. Quando TCP riceve la richiesta , assegna un numero di porta. questo consente ad una connessione di procedere rapidamente senza attendere il processo attivo.

Vi sono due primitive passive open . una specified passive open crea una connessione quando i livelli di precedenza e sicurezza sono accettabili. Una primitive unspecified passive open apre la porta a tutte le richieste. Quest'ultima è utilizzata da server che attendono connessioni da parte di clienti tipo non noto a priori.

TCP ha regole stringenti circa l'uso di processi di connessione attivi e passivi. Usualmente un processo passive open è utilizzato sulla prima macchina e quello active open sull'altra, con informazioni specifiche sul numero socket, precedenza, e livello di sicurezza.

Sebbene la maggior parte delle connessioni TCP viene stabilita da una richiesta attiva ad una porta passiva, è possibile aprire una connessione senza che vi sia una porta passiva in attesa. In questo caso il protocollo TCP che invia la richiesta di connessione sia il numero socket locale sia il numero socket remoto. Se il TCP ricevente è configurato per abilitare la richiesta la connessione può essere aperta.

Timer TCP

TCP usa diversi timer per assicurare che non si abbiano ritardi eccessivi durante la trasmissione. Diversi di questi timer sono eleganti, gestendo problemi che non sono ovvi ad una prima analisi.

IL timer di ritrasmissione

Il timer di ritrasmissione gestisce i timeout di ritrasmissione, che si hanno quando si è esaurito un intervallo tra l'invio di un datagramma e la ricezione di un messaggio di acknowledgment. Il valore del timeout tende a variare, in dipendenza del tipo di rete, per compensare differenze di velocità. Se il timer espira , il datagramma viene rinviato con un RTO modificato che usualmente si incrementa esponenzialmente fino ad un limite massimo. Se si supera il limite massimo, si presume un fallimento della connessione e viene passato un messaggio di errore allo strato superiore.

I valori per il timeout sono determinati misurando il tempo medio necessario perché i dati impiegano vengano trasmessi ad un'altra macchina e l'ACK ritorni indietro, tempo chiamato round-trip time. Da prove viene effettuata una media degli RTT e si determina un valore atteso chiamato smoothed round-trip time o SRTT. Questo valore è poi incrementato per tener conto di ritardi non previsti.

Il timer di quiete

Dopo che è stata chiusa una connessione TCP , è possibile che datagrammi in ritardo tentino di accedere alla porta chiusa. Il timer di quiete ha lo scopo di impedire che la porta appena chiusa venga riaperta di nuovo per ricevere questi datagrammi.

Il timer di quiete è usualmente impostato al doppio del tempo di vita massimo di un segmento (lo stesso valore del campo Time to live in un header IP), assicurando che tutti i segmenti che stiano ancora dirigendosi verso la porta vengano scartati. Tipicamente ciò può portare al risultato che una porta non sia utilizzabile fino a 30 secondi , emettendo messaggi di errore quando altre applicazioni tentano di accedere alla porta durante questo intervallo.

Il timer di persistenza

Il timer di persistenza gestisce un evento molto raro. E' pensabile che una finestra di ricezione possa avere un valore zero, costringendo la macchina trasmittente a mettere in pausa la trasmissione. Il messaggio per riavviare l'invio dei dati potrebbe venir perso, causando un ritardo infinito. Il timer di persistenza attende un tempo

prestabilito e poi invia un segmento di un byte ad intervalli predeterminati per assicurarsi che la macchina ricevente sia ancora intasata.

La macchina ricevente rinvia il messaggio di finestra di ampiezza zero ogni volta che riceve uno di questi segmenti di stato, se è ancora intasata. Se la finestra è aperta , invia un messaggio con il nuovo valore di finestra e la comunicazione è ripristinata.

Il timer keep-alive e il timer di idle

Il timer keep-alive manda un pacchetto vuoto ad intervalli regolari per assicurare che la connessione con l'altra macchina sia ancora attiva. Se non si è ricevuta risposta dopo che è stato inviato il messaggio prima che espiri il timer di idle si presume che la connessione sia interrotta. Il timer keep-alive è usualmente impostato da un'applicazione con valori che variano fra 5 e 45 secondi. Il timer di idle è usualmente impostato a 360 secondi. [1]

Blocchi di controllo della trasmissione e controllo del flusso

TCP deve tenere traccia di molte informazioni su ogni connessione. Esso lo fa attraverso un blocco di controllo della trasmissione (TCB – Transmission Control Block), che contiene informazioni circa i numeri di socket locale e remoto, i buffer di invio e ricezione , i valori di sicurezza e priorità, ed il corrente segmento nella coda. Il TCB gestisce inoltre i numeri di sequenza di invio e ricezione.

---

[1] TCP usa algoritmi di impostazione del timer che sono adattativi per reimpostare i ritardi. I timer si adattano ai ritardi sperimentati in una connessione , alterando il proprio valore per tener conto di problemi insiti in quella realtà di network.

TCB usa diverse variabili per tener traccia dello stato di invio e ricezione e controllare il flusso di informazioni. Queste variabili sono mostrate nella tabella seguente

Variable Name	Description
	Send Variables
SND.UNA	Send Unacknowledged
SND.NXT	Send Next
SND.WND	Send Window
SND.UP	Sequence number of last urgent set
SND.WL1	Sequence number for last window update
SND.WL2	Acknowledgment number for last window update
SND.PUSH	Sequence number of last pushed set
ISS	Initial send sequence number
	Receive Variables
RCV.NXT	Sequence number of next received set
RCV.WND	Number of sets that can be received
RCV.UP	Sequence number of last urgent data
RCV.IRS	Initial receive sequence number

Usando queste variabili il protocollo TCP controlla il flusso di informazioni tra due socket. Un esempio di connessione aiuta ad illustrare l'uso delle variabili. Iniziamo con la macchina A che vuole inviare cinque blocchi di dati alla macchina B. Se il limite della finestra è di sette blocchi , può essere inviato un massimo di sette blocchi senza riscontro. La variabile SND.UNA sulla macchina A indica quanti blocchi sono

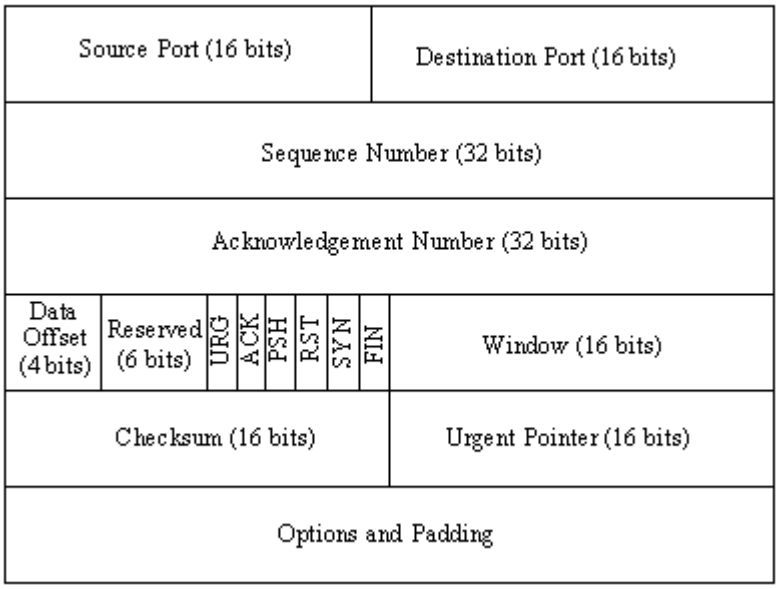
stati inviati e sono senza riscontro (5), e la variabile SND.NXT contiene il valore del prossimo blocco in sequenza (6). Il valore della variabile dell'ampiezza di finestra SND.WND è due (sette possibili blocchi meno cinque già inviati), così solo altri due blocchi potrebbero essere inviati ancora senza sovraccaricare la finestra. La macchina B restituisce un messaggio con il numero di blocchi ricevuti e il limite della finestra viene reimpostato in accordo con questo valore.

Il passaggio di messaggi avanti e indietro può diventare molto complesso se la macchina mittente manda blocchi non riscontrati fino al limite della finestra, in attesa del riscontro dei blocchi più vecchi che sono stati rimossi dalla coda di ingresso , e poi inviando nuovi blocchi per riempire di nuovo la finestra.

TCP Protocol Data Units

TCP deve comunicare con IP nello strato inferiore (usando un metodo definito da IP) e le applicazioni nello strato superiore (utilizzando primitive TCP-ULP). TCP deve inoltre comunicare con altre implementazioni TCP lungo la rete. Per fare ciò esso usa le Protocol Data Units TCP dette anche segmenti.

L'organizzazione di una PDU TCP è la seguente.



i campi sono i seguenti:

- Porta sorgente: un campo a 16 bit che identifica l'utente locale TCP (usualmente un'applicazione di strato superiore).
- Porta di destinazione:un campo a 16 bit che identifica l'utente TCOP della macchina remota.
- Numero di sequenza: un numero che indica la posizione del blocco corrente all'interno del messaggio. Questo numero è usato anche tra due implementazioni TCP per fornire il numero iniziale della sequenza di invio (ISS - Initial Send Sequenze)
- Numero di riscontro: un numero che indica il prossimo numero di sequenza atteso.
- Offset dei dati: il numero di word a 32 bit che si trovano nell'header TCP.questo campo è utilizzato per identificare l'inizio del campo dati.

- Riservato: un campo a 6 bit riservato per usi futuri.
- Flag di urgenza: se settato ad 1 indica che il campo puntatore di urgenza è significativo
- Flag di riscontro (Ack flag) : se settato ad 1 indica che il campo Acknowledgment è significativo.
- Flag di push (Psh flag): se ad 1 indica che deve essere eseguita la funzione push.
- Flag di reset (Rst flag): se ad 1 indica che la connessione deve essere resettata.
- Flag di sincronizzazione (Syn flag): se ad 1 indica che i numeri di sequenza devono essere sincronizzati. Questo flag viene usato quando si sta stabilendo una connessione.
- Flag di termine (Fin flag): se ad 1 indica che il mittente non ha più dati da inviare. Questo è equivalente ad un marcatore di fine trasmissione.
- Window: un numero che indica quanti blocchi di dati può accettare la macchina ricevente.
- Checksum: calcolato prendendo il complemento ad 1 a 16 bit della somma complemento ad 1 nell'header (incluso il pseudo header) e testo.
- Puntatore di urgenza: indica la porzione del messaggio che è urgente specificando l'offset dal numero di sequenza nell'header. TCP non intraprende azioni specifiche che sono invece compito dell'applicazione.
- Opzioni: simile al campo opzioni nell'header IP ed usato per specificare le opzioni TCP. Ciascuna opzione consiste di un numero di opzione (un byte), il

numero di byte nell'opzione e il valore dell'opzione. Soltanto tre opzioni sono attualmente previste nel protocollo TCP: 0 (fine della lista di opzioni), 1 (nessuna operazione), 2 (massima ampiezza del segmento).

- Padding: riempimento aggiunto per assicurare che l'header abbia un'ampiezza che sia un multiplo di 32 bit.

I dati seguono l'header. Il campo opzioni ha una utile funzione. Specificare la massima ampiezza del buffer che può gestire un'implementazione TCP in ricezione. Poiché TCP usa aree dati di lunghezza variabile , è possibile per una macchina mittente creare un segmento più lungo di quello che può gestire il sofware ricevente.

Il campo checksum calcola il checksum basato sull'ampiezza dell'intero segmento, incluso uno pseudoheader a 96 bit che è prefisso all'header TCP durante il calcolo. Lo pseudoheader contiene l'indirizzo sorgente, l'indirizzo destinazione , l'identificatore di protocollo e la lunghezza del segmento. Questi sono i parametri passati allo strato IP quando si chiede di effettuare un invio, e quelli letti quando si tenta di effettuare una consegna.

TCP e le connessioni

TCP ha molte regole per la comunicazione.

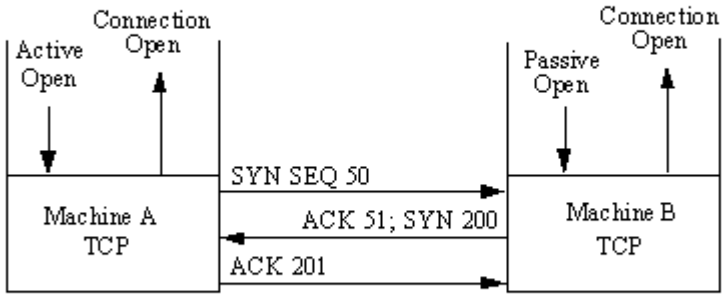
Stabilire una connessione

Una connessione può essere stabilita tra due macchine soltanto se non esiste una connessione fra i due socket, entrambe le macchine sono d'accordo, ed entrambe le

macchine hanno adeguate risorse TCP per servire la connessione. Se qualcuna di queste condizioni non è verificata , la connessione non può essere effettuata.

Quando è stabilita una connessione , ad essa vengono attribuite certe proprietà che restano valide finché la connessione viene chiusa. Tipicamente esse sono un valore di precedenza ed un valore di sicurezza. Questi valori sono concordati fra le due applicazioni durante il processo di instaurazione della connessione.

La figura seguente mostra il diagramma di flusso di un'apertura di connessione.



Il processo inizia con lo strato TCP della macchina A che riceve una richiesta di connessione dal suo ULP, per cui esso manda una primitiva active open alla macchina B. il segmento che viene costruito ha il flag SYN settato ad 1 ed ha un numero di sequenza assegnato. Il diagramma mostra questo con la notazione "SYN SEQ 50", indicando che il flag SYN è settato e il numero di sequenza (numero Initial Send Sequenze o ISS) è 50. (50 è un esempio, avremmo potuto scegliere un qualsiasi numero).

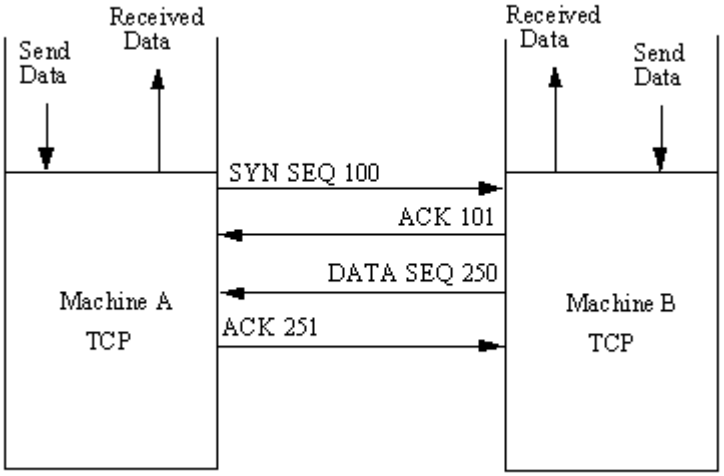
L’applicazione sulla macchina B ha passato un’istruzione passive open al suo strato TCP. Quando viene ricevuto il segmento SYN SEQ 50, lo strato TCP della macchina B manda un acknowledgment alla macchina A con il numero di sequenza 51. Anche la macchina B imposta un numero ISS per suo conto. Il diagramma mostra questo messaggio come “ACK 51; SYN 200”, indicando che il messaggio è un riscontro con il numero di sequenza 51, ha il flag SYN settato, e ha un ISS 200.

Al ricevimento la macchina A manda un messaggio di riscontro con il numero di sequenza settato a 2001. esso è il messaggio “ACK 201” nel diagramma. Poi, avendo aperto e riscontrata la connessione, sia la macchina A che la macchina B mandano messaggi connection open attraverso l’ ULP alle applicazioni richiedenti.

Non è necessario per la macchina remota avere un’istruzione passive open. In questo caso , la macchina mittente fornisce sia il numero socket mittente che quello ricevente, così come i valori di precedenza, sicurezza e timeout. E’ comune per due applicazioni richiedere una active open nello stesso momento. Ciò viene risolto in maniera abbastanza semplice anche se richiede un po’ di traffico.

Trasferimento di dati

Il trasferimento delle informazioni viene diagrammato nella figura seguente.



per ogni blocco di dati ricevuto dal livello TCP della macchina A da parte del ULP, TCP lo incapsula e lo manda alla macchina B con un numero di sequenza progressivo. Dopo che la macchina B ha ricevuto il messaggio, lo riscontra con un acknowledgment che incrementa il successivo numero di sequenza (e quindi indica che ha ricevuto tutto fino a quel numero di sequenza).

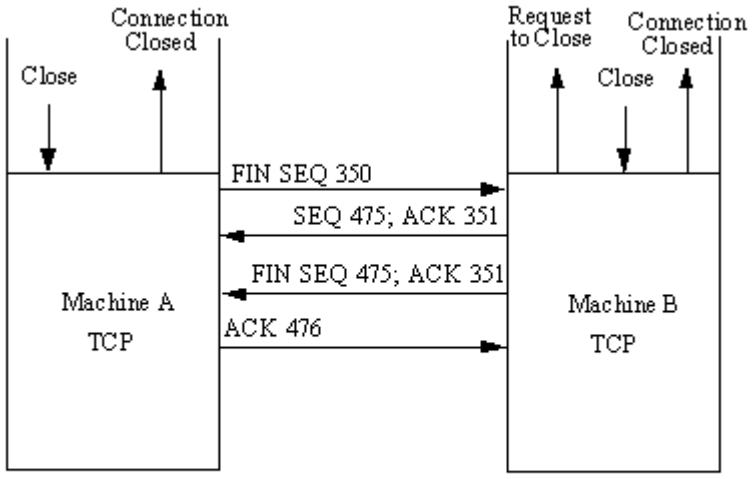
Il servizio di trasporto dei dati TCP contiene sei sottoservizi:

- Full duplex: abilita entrambe le estremità della connessione a trasmettere ad ogni istante, anche simultaneamente
- Temporizzato:l'uso di timer assicura che i dati vengano trasmessi in un ragionevole lasso di tempo.
- Con ordinamento: i dati mandati da un'applicazione sono ricevuti nello stesso ordine all'altra estremità. Questo accade anche se i datagrammi vengono ricevuti dallo strato IP senza ordine, poiché TCP assembla il messaggio in ordine corretto prima di passarlo agli strati superiori.

- Etichettatura: tutte le connessioni hanno un valore concordato di precedenza e sicurezza
- Flusso controllato: TCP può regolare il flusso di informazioni attraverso l'uso di buffer e limiti di finestra.
- Correzione di errori: checksum assicurano che i dati siano privi di errori (nei imiti dell'algoritmo di checksum adottato)

Chiusura di una connessione

Per chiudere una connessione uno dei livelli TCP riceve una primitiva close dal suo ULP ed invia un messaggio con il flag FIN settato. Ciò è mostrato nella figura seguente



nella figura la macchina A invia una richiesta di chiusura della connessione alla macchina B con il successivo numero di sequenza. La macchina B manda indietro un acknowledgment della richiesta e il suo successivo numero di sequenza. Poi la macchina B manda il messaggio di close attraverso il suo ULP all'applicazione e

attende che l'applicazione riscontri la chiusura. Questo passo non è strettamente necessario ; TCP può chiudere la connessione senza l'approvazione dell'applicazione, ma un buon sistema avvertirà la applicazione del cambiamento di stato.

Dopo aver ricevuto l'approvazione per la chiusura della connessione dall'applicazione (o dopo un time out), lo strato TCP della macchina B manda un segmento indietro alla macchina A con il flag FIN settato. Infine la macchina A riscontra la chiusura e la connessione viene terminata.

Una chiusura improvvisa della connessione può aversi quando una della due parti chiude il socket. Questo può essere fatto senza darne notifica all'altra parte e senza preoccupazione per eventuali informazioni che stiano viaggiando fra le due parti. A parte le chiusure improvvise dovute a malfunzionamenti o perdita di energia, terminazioni improvvise si possono avere da un utente, un'applicazione, o una routine di monitoraggio del sistema che giudica la connessione degna di essere terminata. L'altro capo della connessione potrebbe non capire che si è avuta una terminazione improvvisa della connessione finché non tenta di mandare un messaggio e il timer corrispondente espira.

Per tenere traccia di tutte le connessioni, TCP usa una tabella di connessioni. Ogni connessione esistente ha una registrazione sulla tabella che mostra le informazioni sulla stessa . l'organizzazione di questa tabella è mostrata dalla figura seguente

	STATE	LOCAL ADDRESS	LOCAL PORT	REMOTE ADDRESS	REMOTE PORT
Connection 1					
Connection 2					
Connection 3					
Connection n					

Il significato di ogni colonna è il seguente

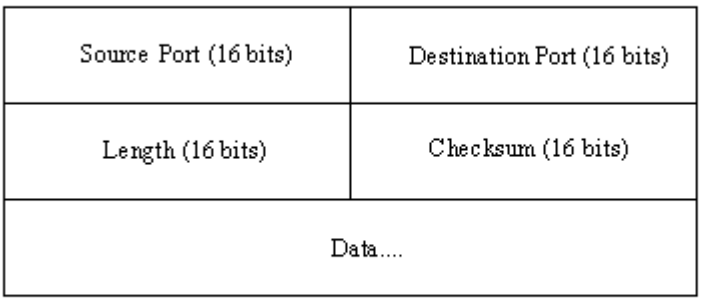
- Stato: lo stato della connessione (chiuso, in fase di chiusura, in ascolto, in attesa, e così via)
- Local address: l'indirizzo IP per la connessione.
- Local port: il numero della porta locale
- Remote Address: l'indirizzo IP della macchina remota
- Remote port: il numero di porta della connessione remota

User Datagram Protocol (UDP)

TCP è un protocollo orientato alla connessione. A volte è richiesto un protocollo non orientato alla connessione, così viene utilizzato UDP: Le comunicazioni senza connessione non forniscono affidabilità, volendo significare che non vi è indicazione per la macchina mittente che un messaggio sia stato ricevuto correttamente. I

protocolli non orientati alla connessione non offrono poi capacità di recupero degli errori. UDP è più semplice di TCP. Esso si interfaccia con lo strato IP (o latri protocolli) senza preoccuparsi di meccanismi di controllo del flusso o di r4ecupreo degli errori, agendo semplicemente come un mittente o ricevitore di datagrammi.

L'header di un messaggio UDP è molto più semplice di quello TCP. Esso è mostrato nella figura seguente



I campi sono i seguenti:

- Source port: un campo opzionale con il numero di porta. se non è specificato un numero di porta esso è posto a zero.
- Destination port: la porta della macchina di destinazione
- Lenght: la lunghezza del datagramma, inclusi header e dati
- Checksum: un complemento ad uno a 16 bit della somma dei complementi ad 1 del datagramma . se il checksum non è utilizzato il campo è posto a zero.