Livello di trasporto.

- TCP (posta, HTTP, FTP): connesso e confermato, reti punto a punto, in realtà fa 2 connessioni, 1 per il controllo ed è sempre aperta l'altra per i dati aperta e chiusa per ogni trasporto.
- UDP (DNS): non connesso e non confermato, multicast (al gruppo) o broadcast (a tutti)

• TCP:

Vede le info come un flusso continuo di byte numerati con un ordine di 32 bit utile al controllo di flusso alla gestione degli ack ed al piggy-backing.
La massima grandezza di un segmento è di 65535 byte (che viene spezzato quando passa all'IP) ed è composto di: header (20byte + parte opzionale), dati da trasportare.
Esso usa il protocollo a finestre scorrevoli di tipo go-back-n con timeout, se questo scade il segmento viene ritrasmesso (la dimensione della finestra ed i valori ack sono espressi in numero di byte)
Formato del frame: n° porta sorgente, n° porta dest., numero d'ordine del 1° byte dei dati, numero d'ordine del prossimo byte da aspettare, dimensione dell'header (per via del campo opzionale a dimensione variabile), bits: URG(1 se il urgent pointer è usato) ACK(1 se l'ack # è valido) PSH(invia subito i dati senza aspettare il riempimento del buffer→come fflush) RST(resetta la connessione→problemi) SYN(syn=1 e ack=0 →richiesta connessione, syn=1 e ack=1 →accettata connessione) FIN(fine connessione), dimensione della finestra (in ricezione indica lo spazio a disposizione nel buffer), checksum, urgent pointer (punta ai dati che devono essere letti subito), campo opzionale (si possono decidere varie cose: dimensione dei segmenti, uso del protocollo selective-repeat, uso di nack, uso del timer).
Nel calcolo del campo checksum entra anche uno pseudo-header gli indirizzi IP ed altre info→ violazione della gerarchia.

▪ Attivazione connessione.

- una delle 2 parti (server) si mette in ascolto ('listen') e quando arriva una richiesta di connessione la accetta ('accept')
- l'altra esegue la primitiva 'connect'(host,porta,ecc..) →invio del segmento TCP
- il server controlla se c'è qualche processo in ascolto su quella porta altrimenti rifiuta la connessione

▪ Rilascio connessione.

La connessione full-duplex viene considerata come 2 simplex

- quando una delle 2 parti ha finito invia un segmento TCP col bit FIN=1
- quando questo viene confermato la connessione scelta viene rilasciata
- anche quando l'altra parte fa lo stesso la connessione full-duplex viene terminata

NB: per evitare il problema dei 3 eserciti vengono usati i timer impostati col doppio del tempo di vita massimo di un pacchetto

▪ Controllo congestione.

Il grosso del controllo della congestione viene effettuato a livello di trasporto.
Il TCP affronta 2 problemi: 1)scarsità buffer destinazione 2) congestione; che vengono gestiti dal mittente tramite la *finestra di congestione* che varia in base al traffico ed alla dimensione della finestra del ricevente (rappresenta quanto si può spedire senza causare la congestione)
La finestra viene gestita così :

- il suo valore iniziale è pari alla dimensione del massimo segmento usato nella connessione
- ogni volta che un ack torna indietro in tempo la finestra si raddoppia fino ad un valore di soglia (inizialmente di 64KB) dopodichè aumenta linearmente di 1 segmento alla volta
- se avviene un timeout: 1) il valore di soglia viene impostato come la metà della dimensione della finestra 2) la dimensione della finestra viene impostata alla dimensione del massimo segmento usato nella connessione

NB: i timeout vengono calcolati in base alla media degli intervalli di tempo degli ack arrivati

Livello presentazione

Gestisce i seguenti problemi: transcodifica, compressione, sicurezza (→livello sessione)

▪ Transcodifica

Le stazioni usano varie rappresentazioni per i numeri e codici differenti (ascii, unicode,…)→ deve conoscere tutte le possibili traduzioni oppure traduce il tutto in una codifica comune. (es: source[A]→cod[B]→dest[A])
NB: l'ASN.1 è uno dei metodi più noti che descrive la sintassi per la traduzione

▪ Compressione: - Con perdita (immagini jpeg)
- Senza perdita (eseguibili)

Esistono vari metodi:

- A lunghezza variabile: dà ad ogni elemento un # diverso di bit utilizzando l'algoritmo di Huffmann (ogni elemento ha una sua probabilità→ si costruisce un albero con questi elementi e a seconda dei rami che si visitano viene assegnato un codice all'elemento)
- RLE (run lenght encoding) (es: bitmap): sfrutta le lunghe sequenze di bit uguali, scrivendo prima un bit e poi quante volte è ripetuto
- A dizionario (es: zip,rar,arj,cab,ace,…): si scandisce il testo e in un buffer vengono inserite le stringhe che sono comparse con probabilità maggiore, poi viene creato l'esito contenente tutte le parole escluse

quelle del buffer + il buffer + i puntatori all'interno dell'esito per individuare le zone dove copiare le parti del buffer.

Livello sessione

Gestisce:
- Crittografia: cifratura del contenuto di un msg
- Autenticazione: essere sicuri che qualcuno non si spacci per un altro utente o server
- Firma digitale: firmato un documento dopo non lo si può disconoscere, il destinatario è certo del mittente

▪ Crittografia

Algoritmi:
- Sostituzione: A=B, B=C, … → la chiave è il nuovo alfabeto creato
- Trasposizione: Si divide il testo in N colonne e le si ordina secondo una chiave di lunghezza N
- A chiave segreta: (molto veloce)

si possiede una chiave segreta, condivisa tra mittente e destinatario, per cifrare e decifrare il msg.
DES: (della IBM) usa una K a 56bit e blocchi di testo di 64bit elaborati con sostituzioni trasposizioni e xor (cifrando decifrando e cifrando con 2 chiavi differenti si aumenta la sicurezza)
DES concatenato: si lega alle parti precedenti del testo
IDEA: usa una K a 126bit e blocchi di testo di 64bit elaborati con xor, moltiplicazioni, divisioni
- A chiave pubblica: 2 chiavi: Kpub→ codifica ; Kpriv → decodifica

RSA:
2 numeri primi p,q > 10^100
n=p*q e z=(p-1)(p+1)
scelgo un numero 'd' che sia primo rispetto a 'z'
ora devo trovare un numero 'e' tale che (e*d)%z=1
→ e,d sono le chiavi di cifratura
es: B (blocco di caratteri da cifrare), cifratura→ C=(B^e)%n, decifratura→ B=(C^d)%n

▪ Autenticazione

Se A dialoga con B, impedisce ad un intruso di capire i loro msg e di interagire con A e B
- Sfida-risposta (A e B hanno una chiave segreta in comune):

A(identità)→B ; A←(valore sfida)B ; A(calore codificato)→B ; A(valore sfida)→B ; A←(calore codificato)B
- Scambio di chiavi:

A e B scelgono 2 grandi numeri primi n,g pubblici (anche (n-1)/2 deve essere primo)
Poi scelgono ciascuno un numero grande x(A) e y(B) segreto
A[n,g,(g^x)%n]→B ; A←[(g^y)%n]B
A eleva alla 'x' ciò che B gli ha spedito e B fa lo stesso con (g^x)%n
Il risultato per entrambi è (g^xy)%n che diventa così la chiave segreta
Un intruso però può intercettare la 1° tripletta e così dirigere la comunicazione A–C–B
- Centro di ditrib. Chiavi

Come in sfida risposta solo che si dialoga con un server intermedio inviandogli il destinatario e la chiave di sessione da usare A–Server–B
- Kerberos

Come il Centro distrib. Chiavi solo che si hanno 2 server: 1 che restituisce una chiave di sessione ogni volta diversa ed 1 che restituisce una chiave per parlare con B
- Con chiave pubblica

Ka,Kb,Ks funzioni di codifica in base alla chiave a,b,sessione; Ra,Rb numeri casuali generati da a,b
A[Kpubb (A,Ra)]→B ; A←[Kpuba (Ra,Rb, propone una Ks)]B ; A[Ks(Rb)]→B

▪ Firma digitale

Per rendere il msg segreto applico una chiave pubblica. Senza segretezza invece invio il testo in chiaro e la sua sintesi codificata con la chiave pubblica, un metodo famoso è l'MD5: divide il msg in blocchi di 512bit+64bit contenenti la lunghezza del msg+128bit di sintesi

▪ Certificati digitali

Contengono informazioni (identità e chiave pubblica di un utente) firmate con la chiave privata di un'autorià di certificazione CA.

▪ PGP

È un pacchetto completo per la sicurezza(cifratura,compressione,firma dig.) e usa algoritmi come IDEA,RSA,MD5