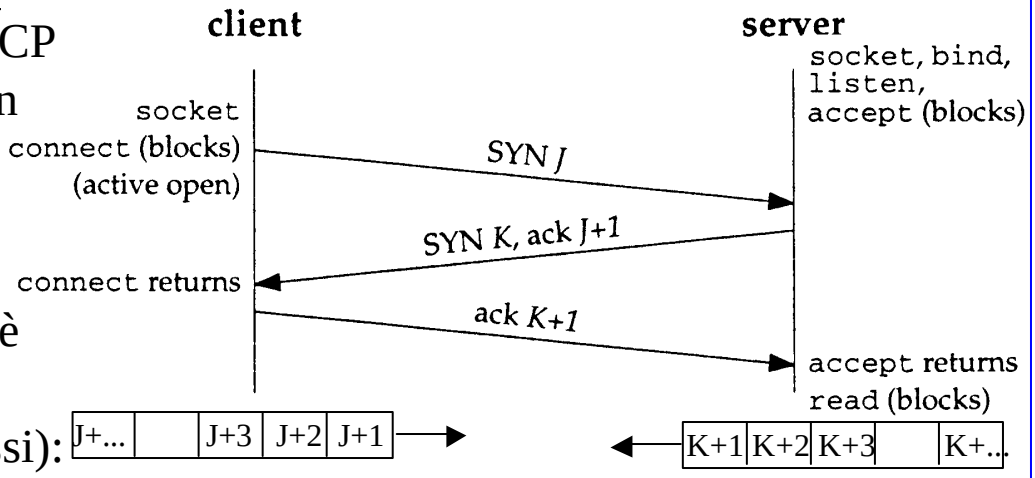
Setup delle ConnessioniTCP

Una connessione TCP viene instaurata con le seguenti fasi, che formano il **Three-Way Handshake** (perchè formato da almeno 3 pacchetti trasmessi):



1) il **server** si predispone ad **accettare una richiesta di connessione**, mediante le chiamate a socket, bind, listen e infine **accept** che realizza una apertura passiva (passive open) cioè senza trasmissione di dati.

2) il client effettua le chiamate a socket, bind ed infine alla **connect** che realizza una apertura attiva (active open) mediante la spedizione di un **segmento** TCP detto **SYN** segment (synchronize) in cui è **settato ad 1 il flag syn**, a zero il flag ack, e che trasporta un **numero di sequenza iniziale** (J) che è il numero di sequenza iniziale dei **dati che il client vuole mandare al server**. Il segmento contiene un header TCP con i numeri di porta ed eventuali opzioni su cui accordarsi, e di solito non contiene dati. Il segmento viene incapsulato in un datagram IP.

3) Il server deve rispondere al segmento SYN del client spedendogli **un segmento SYN** (**flag syn settato ad 1**) con il numero di sequenza iniziale (K) dei **dati che il server vuole mandare al client** in quella connessione. Il segmento presenta inoltre nel campo **Ack number il valore J+1** che indica che si aspetta di ricevere J+1, e presenta il **flag ack settato ad 1**, per validare il campo Ack number.

4) il client, ricevendo il SYN del server con l'Ack numer J+1 sa che la sua richiesta di connessione è stata accettata, e dal sequence number ricevuto K capisce che i dati del server inizieranno da K+1, quindi risponde con un segmento ACK (**flag syn settato a zero** e **flag ack settato a 1**) con **Ack number K+1**, e termina la connect.

5) al ricevimento dell'ACK K+1 il server termina la accept.

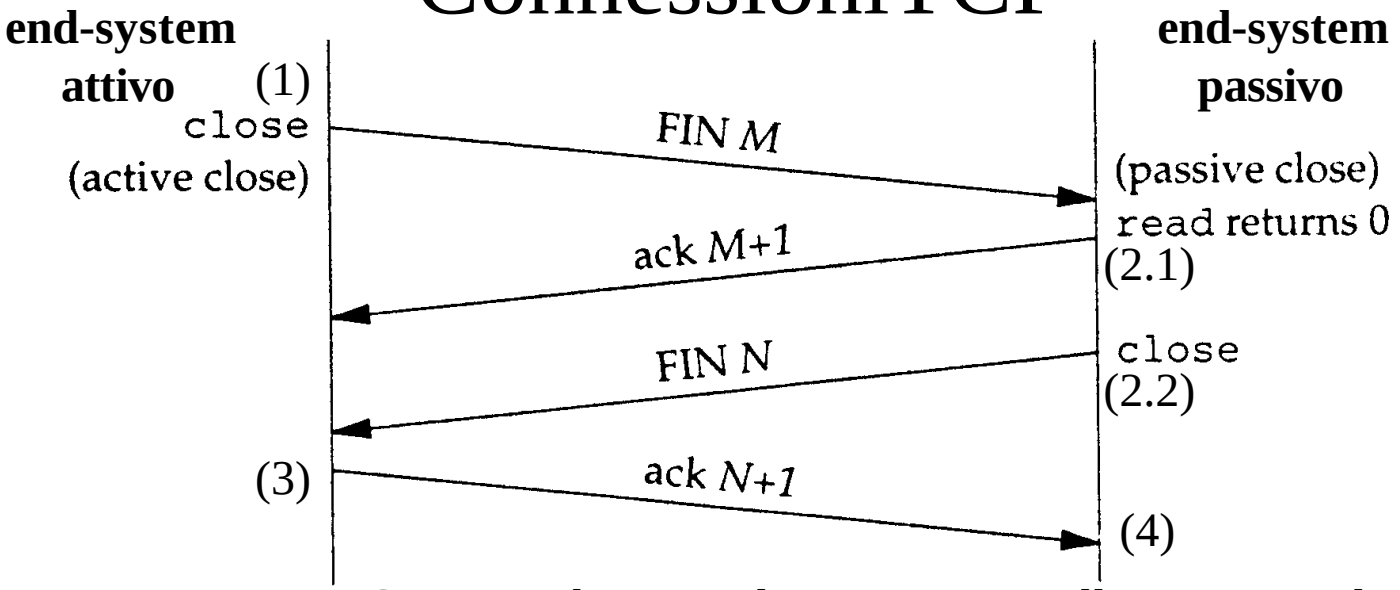
Opzioni TCP nel Setup

Ogni segmento di tipo SYN può contenere delle opzioni, che servono a stabilire alcune caratteristiche della connessione che si sta instaurando. Tra le più importanti ricordiamo:

1) **MSS options**: con questa opzione il TCP che manda il proprio SYN annuncia il maximum segment size, la più grande dimensione di segmento che è in grado di ricevere. L'opzione TCP_MAXSEG resa disponibile dall'interfaccia socket, consente di settare questa opzione.

2) **Windows scale options**: la finestra scorrevole più grande che due TCP possono concordare è 65535, perchè il campo Window Size occupa 16 bit. Per situazioni in cui il collegamento è a larghissima banda ma con grande ritardo di trasmissione (es. via satellite) una dimensione di finestra più grande rende più veloce la trasmissione di grandi quantità di dati. Per identificare finestre più grandi nell'header TCP, si setta questa opzione che indica di considerare il campo Window Size dopo averlo shiftato a sinistra di un numero di posizione compreso tra 0 e 14, in modo da moltiplicare la Window Size di un fattore fino a 2 elevato alla 14, ovvero in modo da raggiungere valori dell'ordine del GigaByte. L'opzione SO_RECVBUF resa disponibile dall'interfaccia socket, consente di settare questa opzione.

Terminazione delle ConnessioniTCP (1)



Una connessione TCP viene chiusa mediante un protocollo composto da **quattro** messaggi trasmessi:

1) una delle applicazioni su un end-system (chiamiamola **attiva**) effettua la chiusura attiva (active close) chiamando la funzione **close()** che **spedisce un segmento FIN** (flag FIN settato a 1), con un numero di sequenza M pari all'ultimo dei byte trasmessi in precedenza più uno. Con ciò si indica che viene trasmesso un ulteriore dato, che è il FIN stesso. Per convenzione il FIN è pensato avere dimensione pari ad 1 byte, quindi l'end-system attivo si aspetta di ricevere per il FIN un ACK con Ack Number pari a M+1.

2) l'end system che riceve il FIN (chiamiamolo **passivo**) effettua la chiusura passiva (passive close) all'insaputa dell'applicazione.

2.1) Per prima cosa il modulo TCP del **passivo spedisce all'end-system attivo un segmento ACK** con Ack number pari a M+1, come riscontro per il FIN ricevuto.

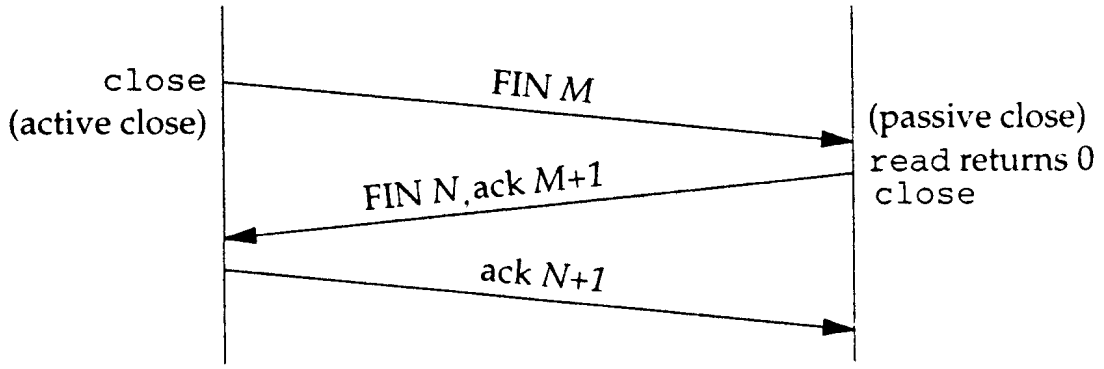
2.2) Poi il TCP passivo trasmette **all'applicazione** padrona di quella connessione il segnale FIN, sotto forma di **end-of-file** che viene accodato ai dati non ancora letti dall'applicazione. Poiche la ricezione del FIN significa che non si riceverà nessun altro dato, con l'end-of-file il TCP comunica all'applicazione che lo stream di input è chiuso.

Terminazione delle ConnessioniTCP (2)

3) Quando **l'applicazione del passivo finalmente legge dal buffer l'end-of-file** (con una read() che restituisce 0), deve effettuare per quel socket la chiamata alla funzione **close().** La close() ordina al modulo TCP di **inviare** a sua volta **all'end-system attivo un segmento FIN**, col numero di sequenza (N) del FIN, cioè l'ultimo byte trasmesso più 1.
4) il modulo TCP dell'attivo, quando riceve il FIN spedisce un ACK con Ack number N+1, cioè il numero di sequenza del successivo al FIN, cioè il FIN più uno, poichè 1 è per convenzione la dimensione del FIN. Terminato questo passo viene conclusa anche la funzione close() dell'attivo.

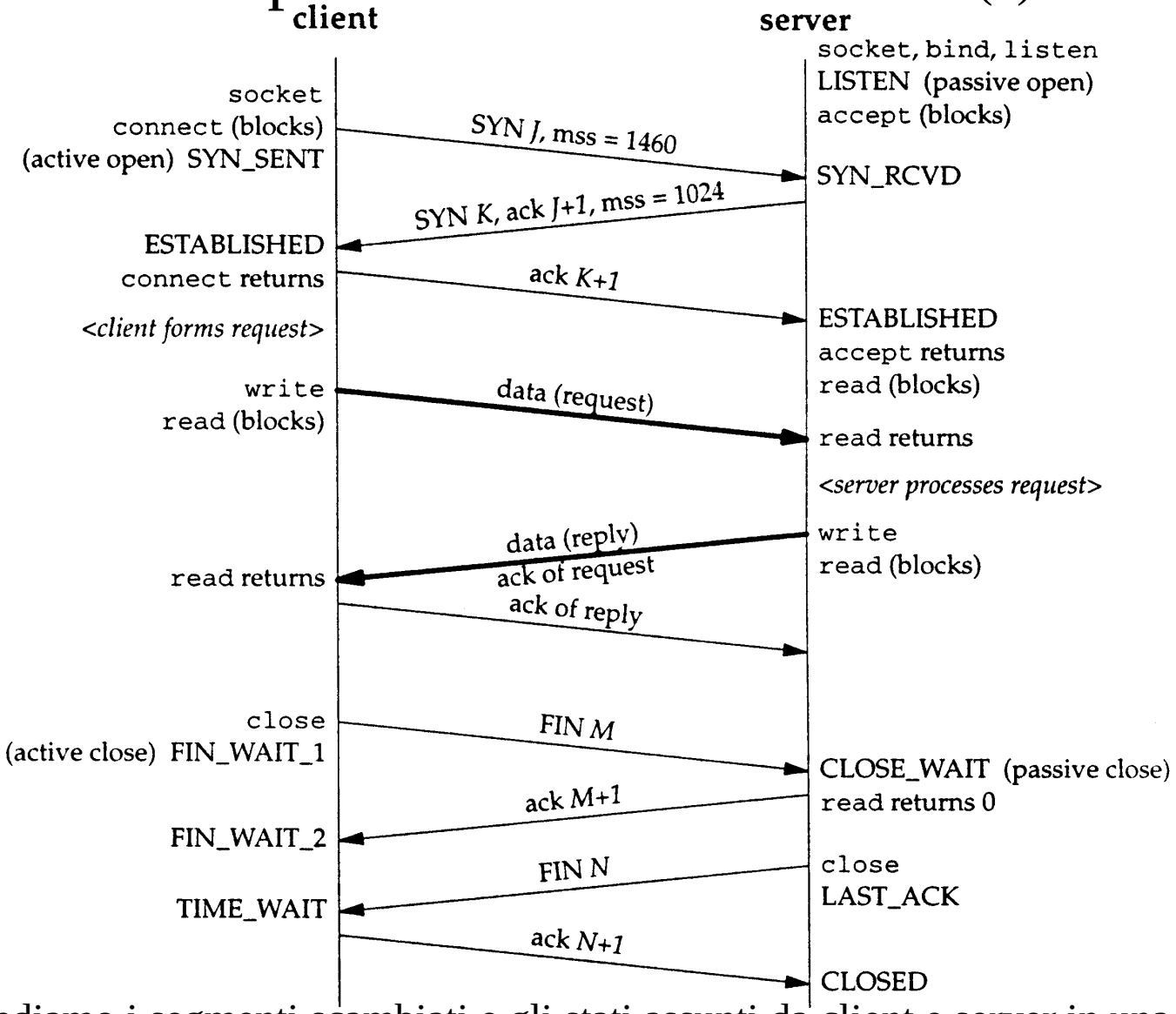
• Chiusura attiva o passiva non dipendono dall'essere client o server, ma solo da chi per primo effettua la chiamata alla funzione close().

• Notare che i due segmenti dal passivo all'attivo degli step 2.1 e 2.2 (ACK M+1 e FIN N rispettivamente) potrebbero essere combinati in un solo messaggio a seconda del comportamento del passivo.



• Un'ulteriore variazione, di carattere opposta alla precedente, è che tra gli step 2.1 e 2.2 **il passivo** ha ancora la possibilità di inviare dei dati verso l'attivo, perchè al momento dello step 2.1 è stata effettuata, mediante il FIN, una chiusura del flusso solo nella direzione dall'attivo al passivo, chiusura che viene detta half-close.

Esempio di connessione TCP (1)



Vediamo i segmenti scambiati e gli stati assunti da client e server in una connessione TCP in cui il client chiede un servizio ed il server risponde. Il client inizia e specifica l'opzione Maximum Segment Size di 1460 byte, il server risponde e specifica una diversa richiesta di MSS di 1024. La MSS può essere diversa nelle due direzioni. Stabilita la connessione il client spedisce una richiesta al server nei dati di un solo segmento. Il server risponde spedendo la risposta nei dati di un solo segmento. Notare che, per diminuire il numero di segmenti scambiati, assieme alla risposta il server spedisce nel segmento anche l'ACK per il segmento ricevuto. Tale tecnica, detta **piggybacking**, viene utilizzata quando il ritardo nella risposta è inferiore ai 200 msec. Infine vengono utilizzati quattro segmenti per effettuare la terminazione della connessione.