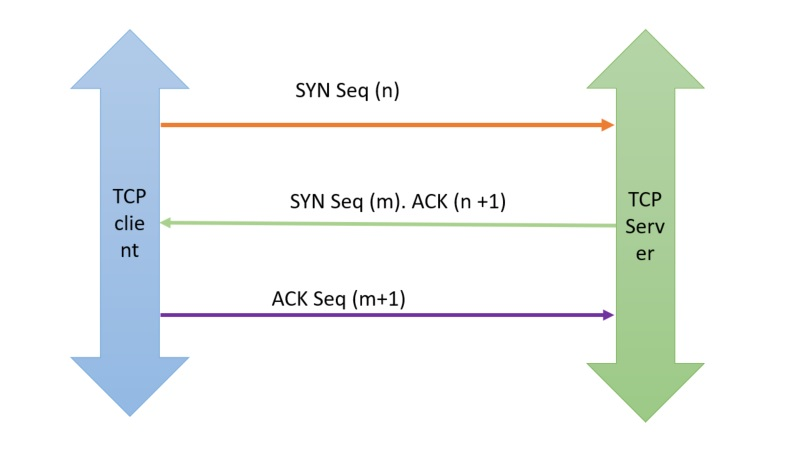
**HANDSHAKE TCP a 3 vie**

È un processo che scambia dei pacchetti tra il cliente e il server prima che inizi la comunicazione con i dati reali ed è divisa in 3 fasi:

**SYN**: SI stabilisce una connessione e manda un pacchetto SYN con una sequenza numerica N.

**SYN**-**ACK**: conferma dall’altra parte che ha ricevuto il SYN e risponde inviando un pacchetto SYN con una sequenza numerica M e invia un pacchetto ACK con la sequenza numerica N+1.

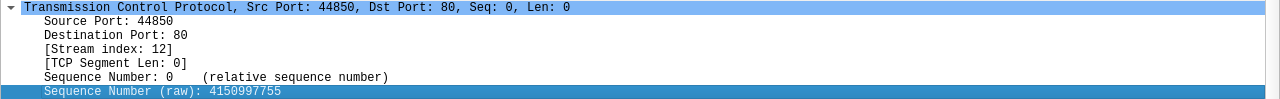
**ACK**: risponde inviando un pacchetto ACK con la sequenza numerica M+1



Il SYN manderà al server una **sequenza numerica**, il SYN-ACK risponderà includendo anche la sequenza numerica **incrementata di 1** per **confermare** che ha ricevuto il SYN, ecco un esempio:

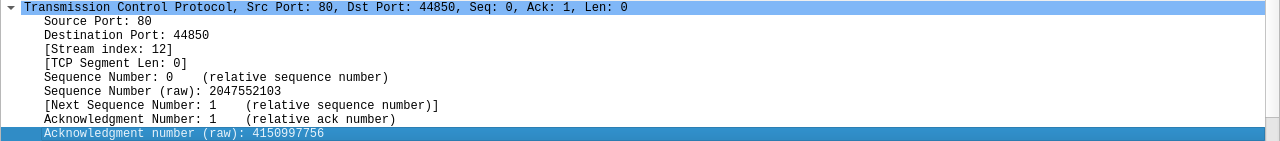
**SYN**

1085 75.658795550 10.0.3.12 188.184.21.108 TCP 74 44850 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK\_PERM=1 TSval=1575878703 TSecr=0 WS=128



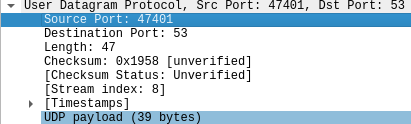
**ACK-SYN**

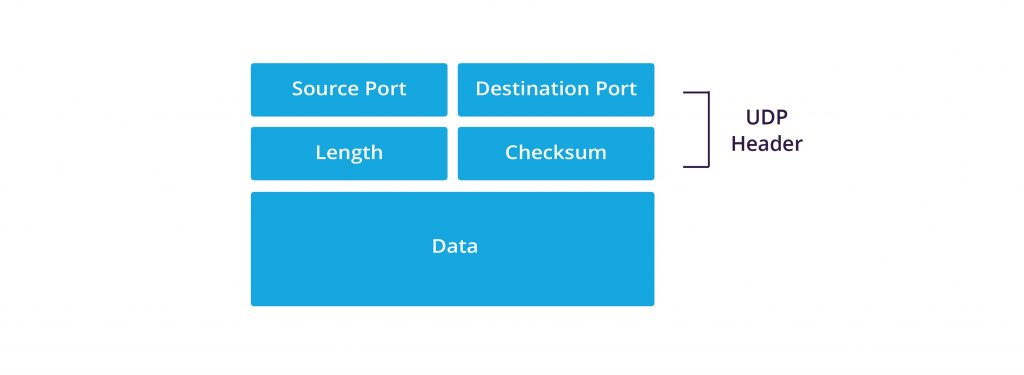
1087 75.683125536 188.184.21.108 10.0.3.12 TCP 74 80 → 44850 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1380 SACK\_PERM=1 TSval=3442939234 TSecr=1575878703 WS=128



**Protocollo UDP:**

* Non è orientato alla connessione
* È più veloce ma non è affidabile
* Non gestisce il riordinamento dei pacchetti né quelli persi
* È un protocollostateless, quindi ha meno informazioni da memorizzare





**Source** **port**: numero della porta sorgente

**Destination** **port**: numero della porta del destinatario

**UDP** **length**: lunghezza dell’UDP incluso l’header e la data

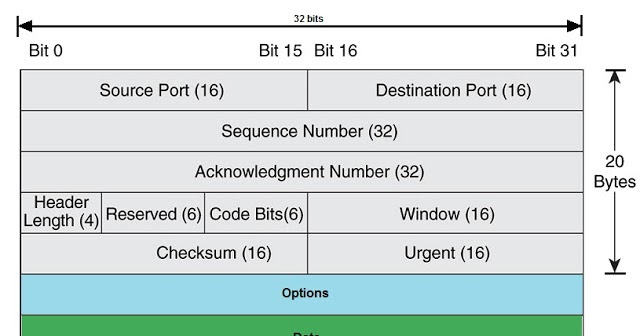
**Checksum:** controllo degli errori

**Data**: proviene dal livello applicativo

Non c’è l’indirizzo ip perché vengono gestiti a livello 3 e qui siamo a livello 4

**Protocollo TCP:**

* È un protocollo utilizzato per il trasferimento dati in modo affidabile
* Orientata alla connessione
* È un protocollo non connesso, cioè i pacchetti non seguono la stessa strada
* È full-duplex, cioè permette la comunicazione contemporanea nelle due direzioni
* I flussi di byte vengono frazionati in segmenti
* Garantisce che i dati arrivino in ordine e una volta sola

****

**Source port:** porta della sorgente

**Destination port:** porta del destinatario

**Sequence Number:** indica la posizione del primo byte di dati del segmento

**Acknowledgment Number:** indica la posizione del prossimo byte che si aspetta di ricevere

Questi due servono per numerare ciò che viene mandato e ciò che viene ricevuto, per capire se qualcosa è andato male (duplicazioni o fuori ordine)

**Header length:** lunghezza dell’intestazione

**Reserved:** bit non utilizzati

**Code Bits:** bit utilizzati per il controllo del protocollo:

* **SYN**
* URG
* **ACK**
* CWR
* ECE
* PSH
* RST
* **FIN**

**Window:** indica la dimensione della finestra della ricezione dell’host destinatario, cioè il numero massimo di byte che il destinatario è in grado di accettare a partire dell’**acknowledgment number.** Ogni volta che il destinatario riceve dei pacchetti lui risponde con l’acknowledge facendo sapere quanto spazio ha ancora, poi quando finisce lo spazio e quando lo riottiene invierà di nuovo l’acknowledge facendo sapere lo spazio disponibile.

**Checksum:** controlla che i dati siano arrivati senza errori

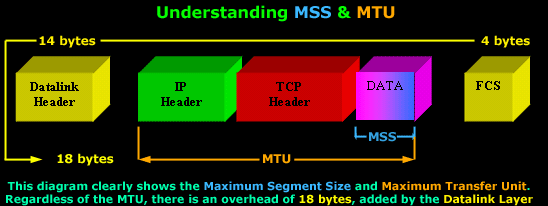
**Urgent:** funziona se solo il flag URG è impostato a 1 e permette dei segmenti di superare altri segmenti.

**Options:**

* **MSS**
* **Window Scaling**
* **Selective Acknowledgment (SACK)**
* **Timestamps**
* **Nop**

Aiutano a superare i problemi creati dai nuovi collegamenti e velocità disponibili. Questo campo può valere fino a 40 bytes.

**MSS:**



Indica la **grandezza massima del segmento utilizzato nella connessione tra i dispositivi**.

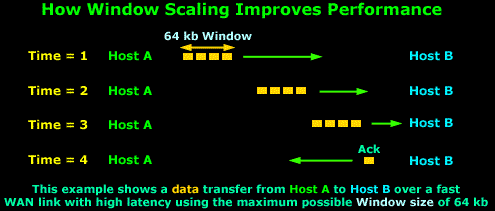
Viene usato solo durante la **fase di SYN e SYN-ACK.** La grandezza del MSS di solito è inferiore alla MTU, nelle reti Ethernet di solito sono 1460 byte, dei quali 1500 sono della MTU meno 20 byte per l’intestazione IP e 20 byte per l’intestazione TCP

Se l'MSS è troppo grande, i segmenti più grandi possono essere frammentati e riassemblati, il che aumenta la probabilità di perdita di pacchetti o ritardi. Se l'MSS è troppo piccolo, può comportare un uso inefficiente delle risorse di rete, poiché sarà necessario trasmettere più segmenti per inviare la stessa quantità di dati.

**Window Scaling:**

È un'estensione del campo Window che aumenta la sua dimensione. In totale sono 30 bit, 16 bit del campo Window più i 14 bit dell’opzione Window Scaling. È stato creato per la trasmissione dati in aree molto estese (WAN)

**Usando solo window:**

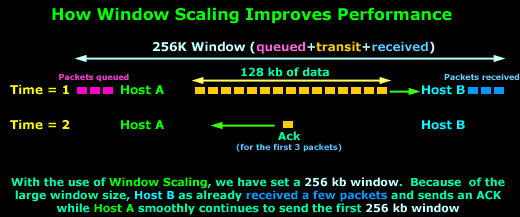


Stiamo utilizzando la dimensione massima del Window di 64 kb e poiché il collegamento WAN ha una latenza molto elevata, i pacchetti impiegano del tempo per arrivare all'host B. A causa dell'elevata latenza, l'host A ha interrotto la trasmissione dei dati in quanto vi sono 64 kb di dati inviati e non sono stati ancora riconosciuti.

Nel Time = 4, l'host B ha ricevuto i dati e invia il riconoscimento all'host A in modo da poter continuare a inviare i dati, ma il riconoscimento ci metterà un bel po’ per arrivare

Quindi, da Time = 1 fino a Time = 6, l'host A aspetta. Se dovessimo trasferire un file da 10 Mb, ci vorrebbero ore!

**Usando window scaling:**



La dimensione del Window è aumentata a 256 kb. Il che si traduce in più dati durante il transito, l'host B ha già ricevuto i primi pacchetti, mentre l'host A sta ancora inviando la prima finestra da 256 kb.

Nel Time = 2, l'host B invia un riconoscimento all'host A, che è ancora impegnato nell'invio dei dati. L'host A riceverà il riconoscimento prima che termini la finestra di 256 kb e continuerà quindi a inviare i dati senza interruzioni poiché riceverà presto un altro riconoscimento dall'host B.

**Selective Acknowledgment (SACK):**

Il destinatario manda un SACK informando al mittente se ci sono stati dei pacchetti persi o se non sono arrivati in ordine e chiede di rinviare i pacchetti mancanti.

**Esempio usando il SACK:**

Se abbiamo un client Windows 98 che attende il byte 4.268, ma l'opzione SACK mostra che il client Windows 98 ha ricevuto anche i byte da 7.080 a 8.486, è ovvio che mancano i byte da 4.268 a 7.079, quindi il server dovrebbe solo inviare nuovamente i 2.810 byte mancanti, anziché riavviare l'intero trasferimento al numero di byte 4.268.

**Esempio usando vecchi SACK:**

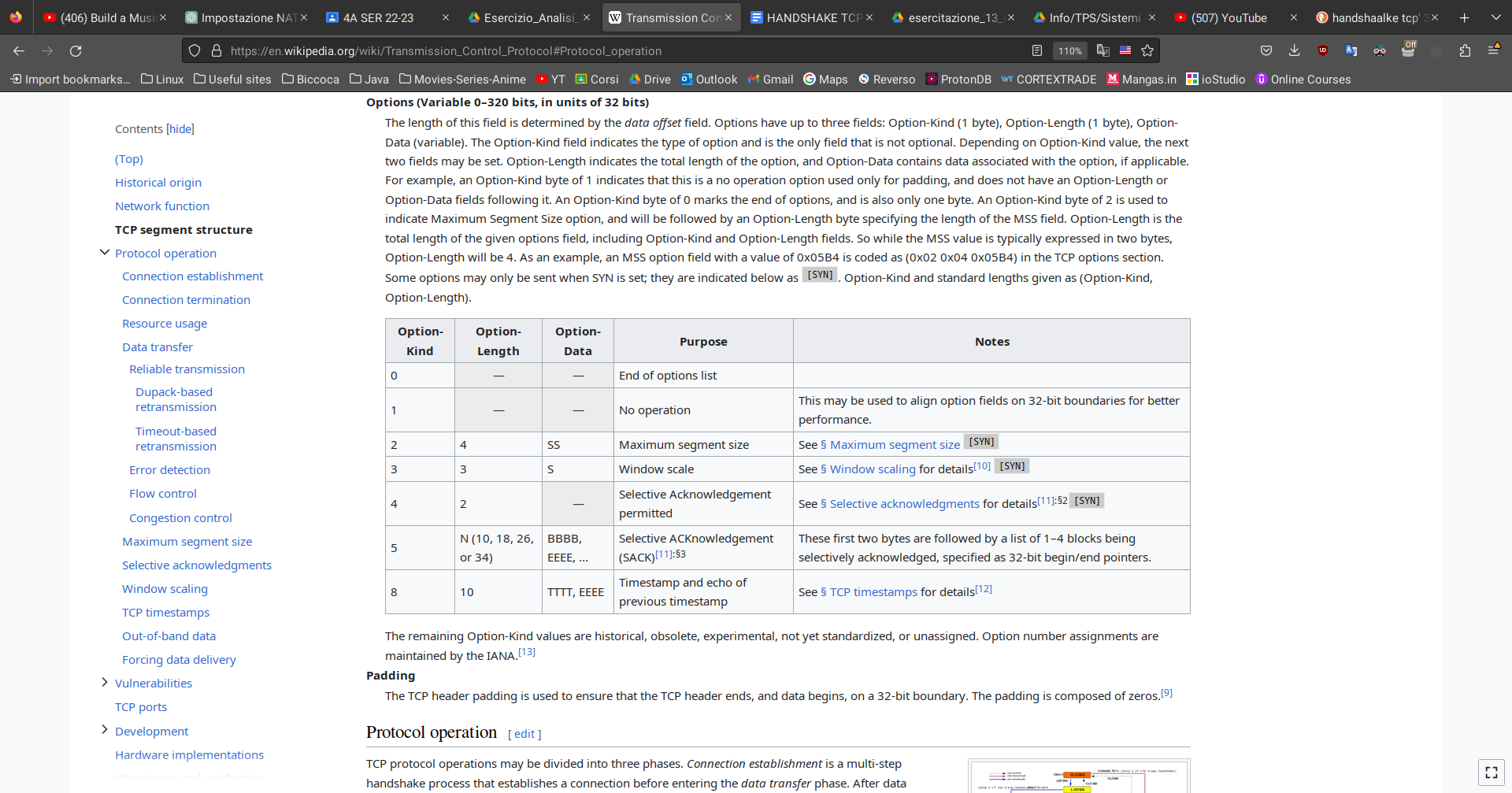
Il problema con i vecchi acknowledgment è che non ci sono meccanismi per un ricevitore per affermare "Sto ancora aspettando i byte da 20 a 25, ma ho ricevuto i byte da 30 a 35". Usando i 'classici' acknowledgment supportati da TCP, si può solo dire "Ho ricevuto tutto fino al byte 20". Il mittente deve quindi riconoscere che qualcosa è andato storto e inviare dal byte 20 in poi

**Timestamp:**

Calcola il tempo di consegna di andata e ritorno dei pacchetti in una rete. Il tempo di consegna determinerà con precisione per quanto tempo il TCP attenderà prima di tentare di ritrasmettere un segmento che non è stato riconosciuto. Ciò consente a TCP di adattarsi meglio alle congestioni e trasmettere dati in modo efficiente e affidabile su diversi tipologie di reti.

Il campo Timestamp è costituito da un campo Timestamp Echo e Timestamp Reply,

**Nop:**

Significa “Non utilizzata” e viene utilizzata per separare le diverse opzioni utilizzate all'interno del campo Options. Questo campo può variare a seconda del sistema operativo, per esempio se vengono utilizzate le opzioni MSS e SACK, Windows XP di solito inserirà due nop tra di esse. Per ogni Nop si usa 1 byte.

<https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Protocol_operation>

**Attacchi Dos/SYN Flood**

Ha lo scopo di ostacolare l’uso di un sistema online. Per farlo semplice, consiste nell’invio massivo di pacchetti inutili ad un ente. Le “caselle di posta” dell’ente sono piene e quindi l’ente non riuscirà più a ricevere pacchetti perché il traffico è collassato.

È un attacco di protocollo, questo tipo di attacchi mira a sfruttare le falle della connessione di rete.

1. L’aggressore invia al server un pacchetto SYN emaschera il suo indirizzo IP.
2. Il server crea una struttura di dati **Transmission Control Block (TCB)** per la connessione semi-aperta.
3. Il server invia un pacchetto SYN/ACK all’indirizzo IP mascherato dell’aggressore.
4. Poiché dal server dell’aggressore non entra alcun pacchetto ACK, per confermare la connessione, il server continua a inviare altri pacchetti SYN/ACK al presunto client **mantenendo la connessione semi-aperta.**
5. Mentre il server continua ad attendere una risposta, **arrivano nuovi pacchetti** SYN dell’aggressore e il **server crea altri TCB**
6. Da un momento preciso, **non ci sarà più spazio per altre connessioni semi-aperte**. Il server respingerà altri pacchetti SYN in entrata e non sarà più raggiungibile dall'esterno.

**Le varianti:**

* **SYN Flood diretto**: l'aggressore fa partire l’attacco usando il proprio indirizzo IP. Per essere sicuro di respingere i pacchetti in entrata SYN-ACk, l'aggressore configura il suo firewall per respingerli
* **SYN Flood con IP mascherato:** l’aggressore usa un indirizzo IP falsificato nel campo del mittente dei pacchetti SYN. Di solito si usano indirizzi IP che risultano liberi al momento dell'attacco per assicurarsi che i sistemi scelti a caso non rispondano con un pacchetto RST alle risposte SYN/ACK dei server attaccati terminando in questo modo la connessione.
* **SYN Flood Distributed Denial of service (DDoS):** l’attacco parte contemporaneamente da diversi computer. Si tratta di una **botnet** cioè di una rete di computer sotto il controllo dell’aggressore
* **SYN Flood reflection:** Di solito un server risponde a un singolo pacchetto SYN con più pacchetti SYN/ACK. E quindi l'aggressore maschera l’indirizzo IP della vittima e fa partire un SYN Flood DDoS contro uno o più server estranei. Ciascuno dei server risponderà a ogni pacchetto SYN in entrata con più pacchetti SYN/ACK che saranno inviati alla vittima provocando una **moltiplicazione del traffico di rete**

**Misure difensive:**

### **Aumentare la capacità del backlog del SYN:**

Il backlog sarebbe una sorta di tabella, ciascuna casella contiene le informazioni per creare la singola connessione TCP. Solo dopo aver stabilito la connessione tramite il handshake a 3 vie, viene inoltrato alle applicazioni in ascolto sulle porte e rimosso dal backlog del SYN.

Quindi la soluzione sarebbe di aumentare la capacità del backlog del SYN.

**Riciclare le connessioni TCP semi-aperte più vecchie:**

Consiste nel **cancellare le connessioni semi-aperte più vecchie dal backlog del SYN** così **si libera lo spazio per una nuova connessione semi-aperta..** Ma è inefficace nel caso di un attacco volumetrico.

**SYN cache e SYN cookies:**

**TCB:** viene **creato** quando un’entità TCP apre una connessione TCP econtiene lo stato della connessione, il **numero di sequenza locale** e il **numero di sequenza inviato dal client remoto**

**SYN cache:** il server tiene traccia delle connessioni semi-aperte che non hanno ricevuto una risposta in una cache. Quando non si riceve una risposta ACK da parte del client dopo un periodo di tempo, la richiesta SYN viene rimossa. Lo svantaggio è che può consumare risorse da parte del server.

**SYN cookies:** il server include un **parametro speciale nella sequenza numerica del pacchetto SYN-ACK** inviato al client il quale deve **includere questo parametro nella sequenza numerica del pacchetto ACK** per stabilire la connessione.

Un client legittimo risponde al pacchetto SYN/ACK con un pacchetto ACK e ricorre a una **sequenza numerica particolare.** Il **server** utilizza la sequenza numerica del pacchetto ACK per **verificare** la **crittografia** **della connessione e stabilire la connessione**.

Entrambe le tecniche si usano in combinazione

**Mitigation service di cloud computing:**

L’idea è di **distribuire il flusso di dati DDoS** in entrata **su tanti sistemi singoli** permettendo di disperdere il carico totale dell'attacco

**Le porte**

Sono dei punti di accesso che sono associati a un processo (mail server, comunicazione tra app, accesso ai social network). Sono **identificati** da un **numero** intero positivo **in 16 bit**, da **0-65 535**:

* **Well Known Ports (0-1023)**
* **Registered Ports (1024-49 151)**
* **Private Ports (49 152-65 535)**

La IANA ha assegnato le stesse porte sia per UDP sia per TCP

**Le socket:**

Quando un programma desidera comunicare su una rete, crea un socket, lo associa a un indirizzo IP e un numero di porta specifici e lo utilizza per inviare e ricevere dati.

Deve contenere 5 informazioni principali (**quintupla**) detta Association:

**Protocollo, Ind. IP sorgente, Porta sorgente, Ind. IP dest, Porta dest**

In realtà la quintupla è **divisa in due parti**, uno logico e l’altro remoto, per esempio:

* TCP, IP logico, Porta logica
* TCP, IP remoto, Porta remota