ITS Academy
Adriano
Olivetti

TCP / IP - Livello di Trasporto

Modello TCP / IP – Livello di Trasporto (TCP - UDP)

- Il livello di trasporto ha il compito di svincolare il livello superiore (Applicativo) dai problemi di routing di cui si occupa il livello di rete.
- Fornisce alle applicazioni l'interfaccia per utilizzare il livello di rete
- Gestisce un sistema di indirizzamento all'interno del singolo host, basato su numeri chiamati porte (che vengono accodati all'indirizzo IP nella forma xxx.xxx.xxx.xxx : porta)
- Gestisce sia l'affidabilità che le prestazioni della comunicazione
- Sono due i protocolli disponibili:
  - TCP (Transmission Control Protocol) → Orientato all'affidabilità
  - UDP (User Datagram Protocol) → Orientato alle performance

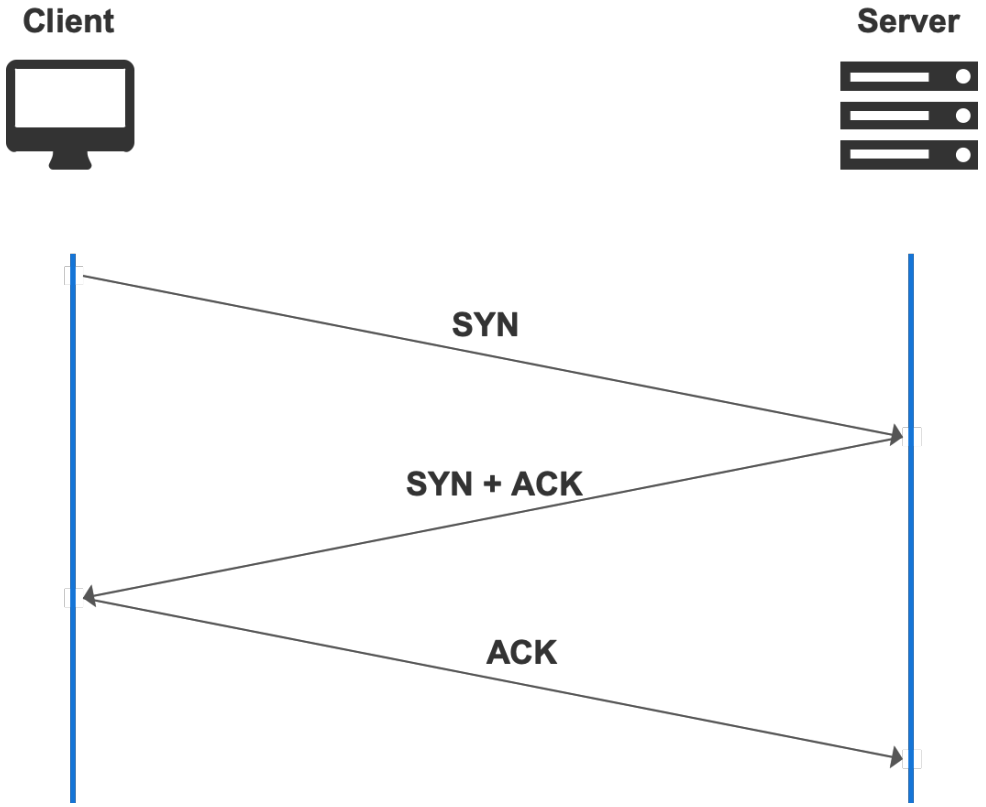
Modello TCP / IP – Livello di Trasporto - TCP

Caratteristiche principali di TCP:

- È orientato alla connessione: I due endpoint devono cioè stabilire una connessione prima di poter comunicare.
- È client-server: I due endpoint non sono equivalenti. Uno dei due (server) rimane passivo, in ascolto su una determinata porta, ed attende che l'altro (client) inizi il processo di connessione. La connessione rimane attiva per tutto il tempo necessario (il server impiega risorse per farlo)
- È affidabile: Verifica che i dati siano ricevuti in modo corretto
  - Riordinandoli se arrivati fuori sequenza
  - Chiedendo la ritrasmissione se alcuni non sono arrivati
  - Gestisce congestioni di rete
- Genera overhead: Consuma maggior banda di trasmissione

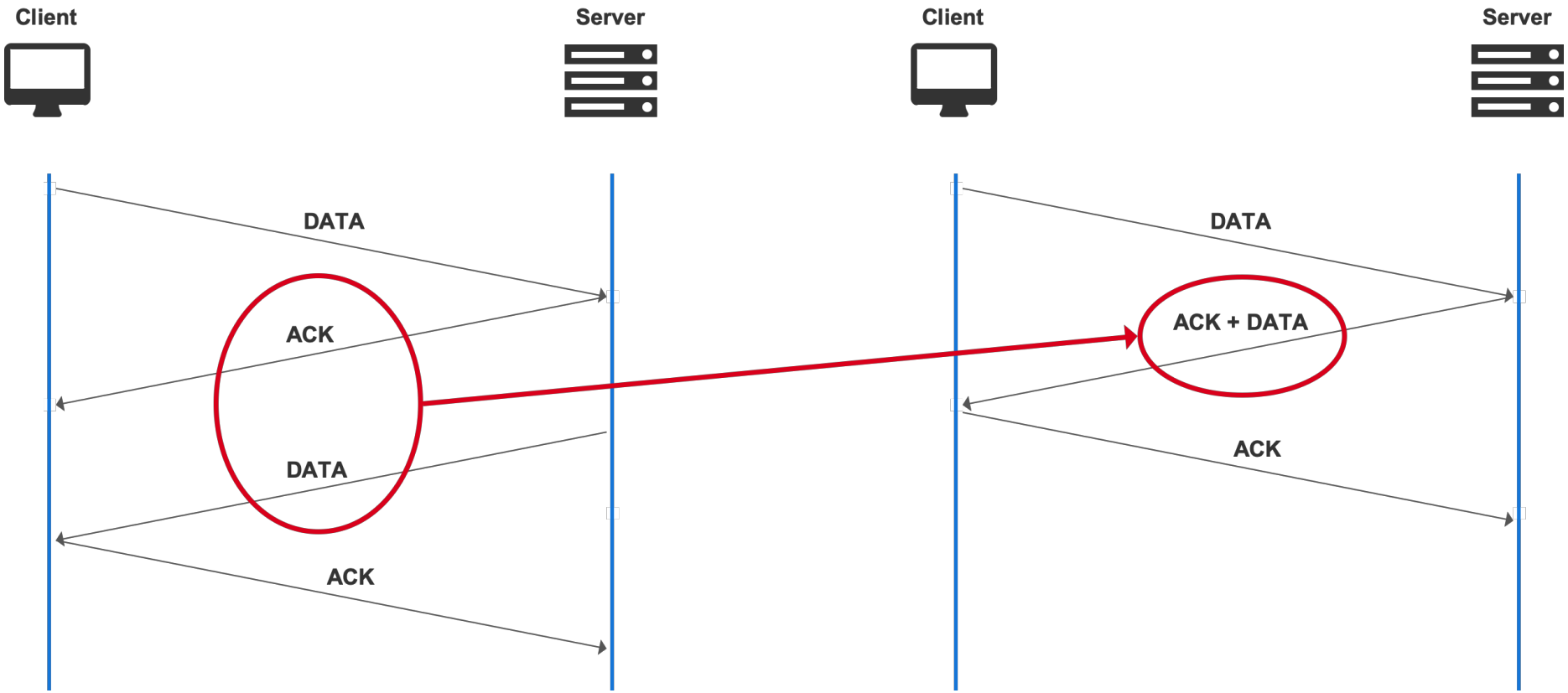
Modello TCP / IP – Livello di Trasporto - TCP

La connessione TCP avviene tramite un 3 way handshake, nel quale client e server si scambiano pacchetti opportunamente configurati:

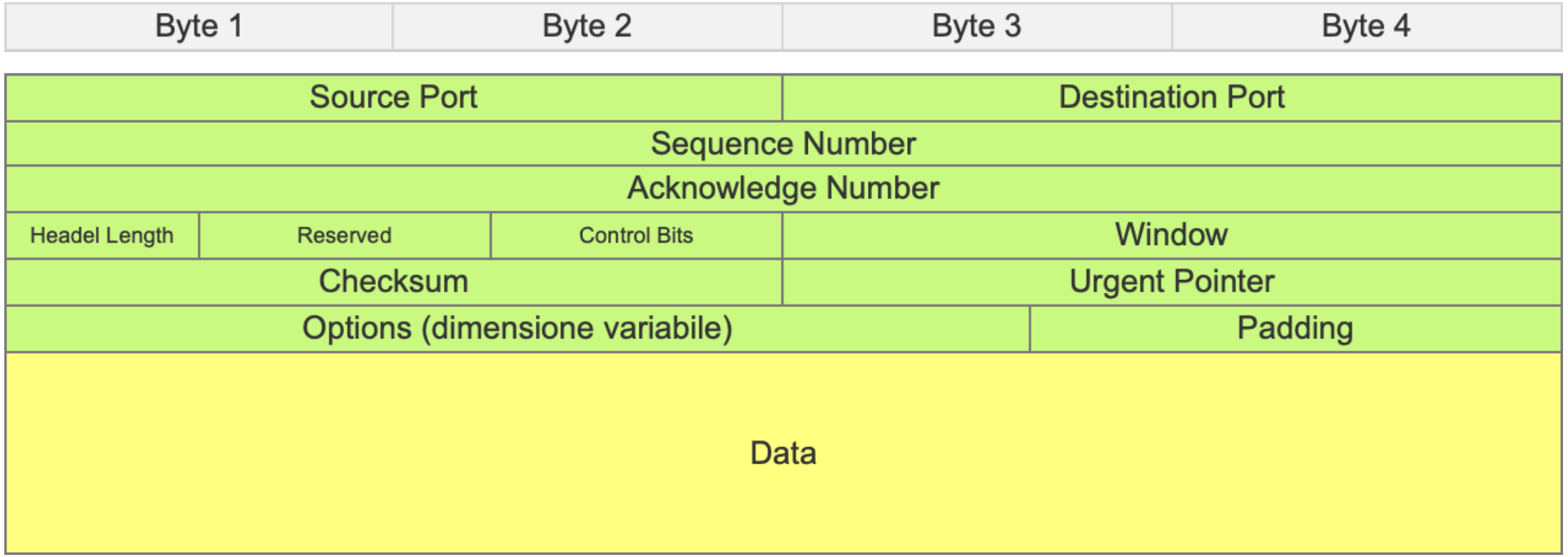


Modello TCP / IP – Livello di Trasporto - TCP

Dopo l'handshake tutti i trasferimenti sono seguiti da un pacchetto di SYN. In alcuni un pacchetto dati e un pacchetto di SYN possono essere uniti in un unico pacchetto.



Modello TCP / IP – Livello di Trasporto - TCP



Modello TCP / IP – Livello di Trasporto - TCP

- Source e Destination port: Numeri di porta del mittente e del destinatario. 16 bit ciascuno, pertanto le porte massimo sono 65536.
- Sequence Number: Numero di sequenza del primo byte del pacchetto. Usato per riassemblare i pacchetti ricevuti. Se SYN = 1 indica il numero di sequenza iniziale
- Acknowledge number: Numero di sequenza che il ricevitore si aspetta. Valido nel caso che il bit ACK valga 1. Il primo numero di ACK è dato dal numero di sequenza iniziale +1
- Header Length: Lunghezza dell'header espresso in parole da 32 bit. Il campo è di 4 bit.
- Reserved: 6 bit al momento non utilizzati

NB: I campi Sequence e Ack number non partono da zero ma da un numero randomico

Modello TCP / IP – Livello di Trasporto - TCP

- Controll bit: 6 bit ognuno per inserire una specifica sul pacchietto:
  - URG: Vale 1 se si deve considerare il campo urgent pointer
  - ACK: Vale 1 se si deve considerare il campo Acknowlege (quasi tutti i pacchetti lo hanno settato)
  - PSH: Vale 1 se si pacchetto contiene dati "push" da inoltrare all'applicazione immediatamente, senza passare per un buffer
  - RST: Vale 1 per resettare la connessione
  - SYN: Usato per stabilite le connessioni
  - FIN: Vale 1 per indicare che il mittente non ha dati da inviare

Modello TCP / IP – Livello di Trasporto - TCP

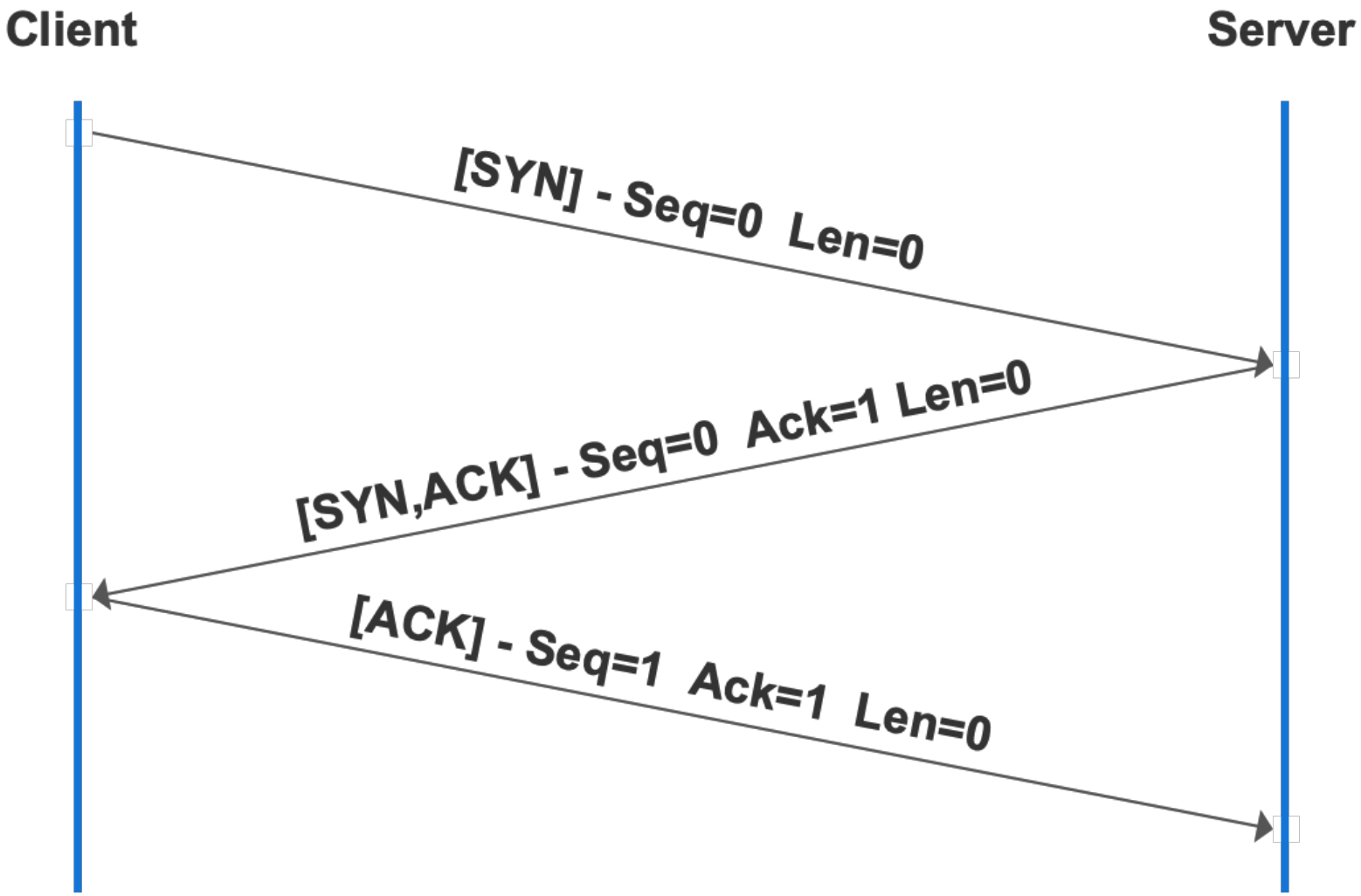
- Window: Indica il numero di byte che il ricevitore è in grado di ricevere con il prossimo invio. È un campo di 16 bit → Un pacchetto TCP può contenere al massimo 65535 bytes
- Checksum: Controllo dell'errore
- Urgent Pointer: Contiene il puntatore a dati urgenti, eventualmente presenti nel pacchetto
- Options: Contiene opzioni aggiuntive per la connessione
- Padding: Riempitivo per portare l'header a dimensione multipla di 32 bit.

Modello TCP / IP – Livello di Trasporto - TCP

Per semplificare il calcolo dei numeri di sequenza e di ack possiamo usare la seguente astrazione:

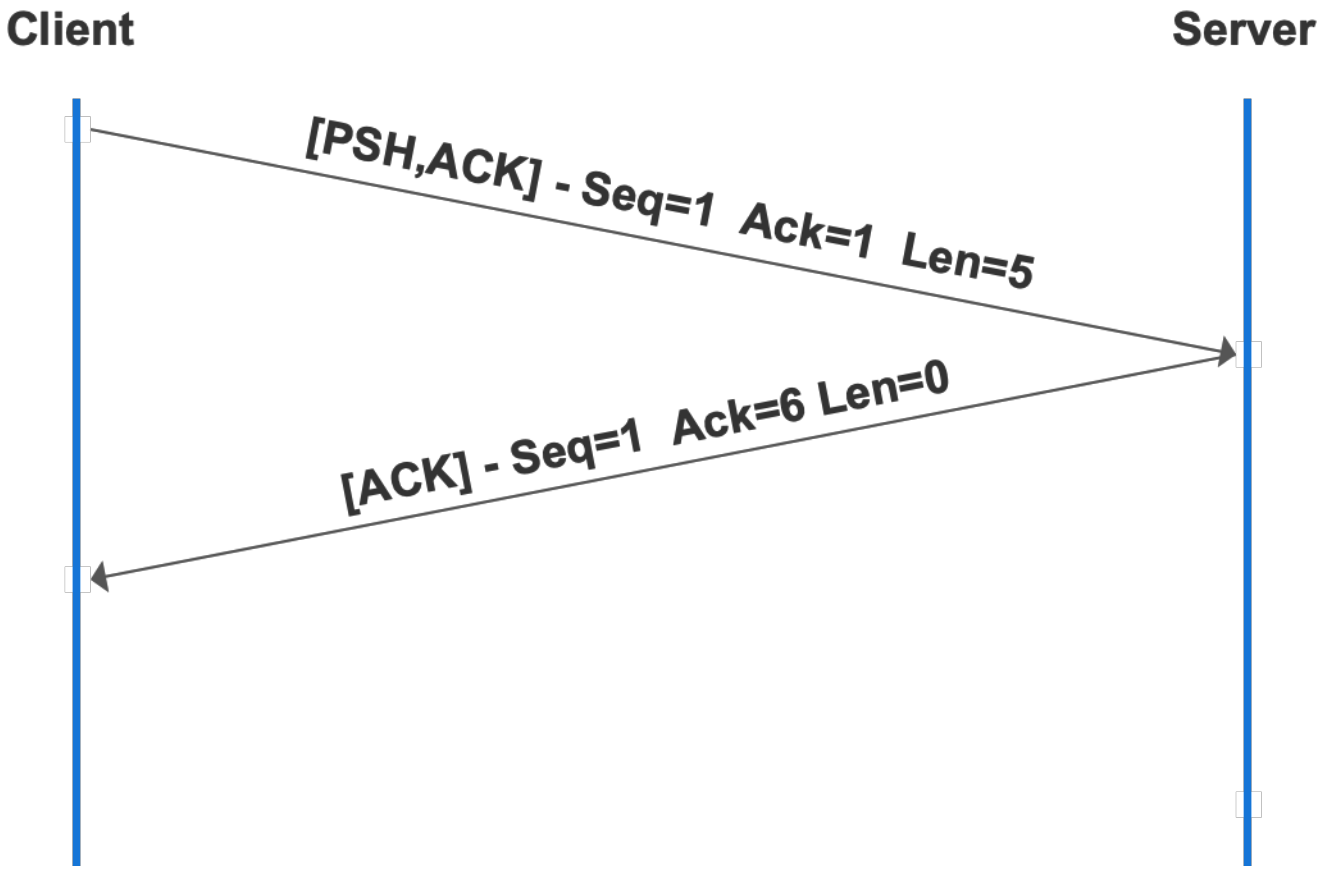
- Ogni host mantiene due contatori, uno per la sequenza e uno per l'ack
- Ogni host determina il numero di sequenza iniziale in modo randomico
- In ogni invio vengono inseriti i due contatori
- Dopo l'invio il numero di sequenza viene aumentato della quantità di dati inviati
- Quando si ricevono dati il numero di ack viene aumentato della quantità di dati ricevuti
- L'handshake usa regole differenti:
  - SYN → Si invia il numero di sequenza iniziale e lo si incrementa di uno per i successivi invii

Modello TCP / IP – Livello di Trasporto - TCP



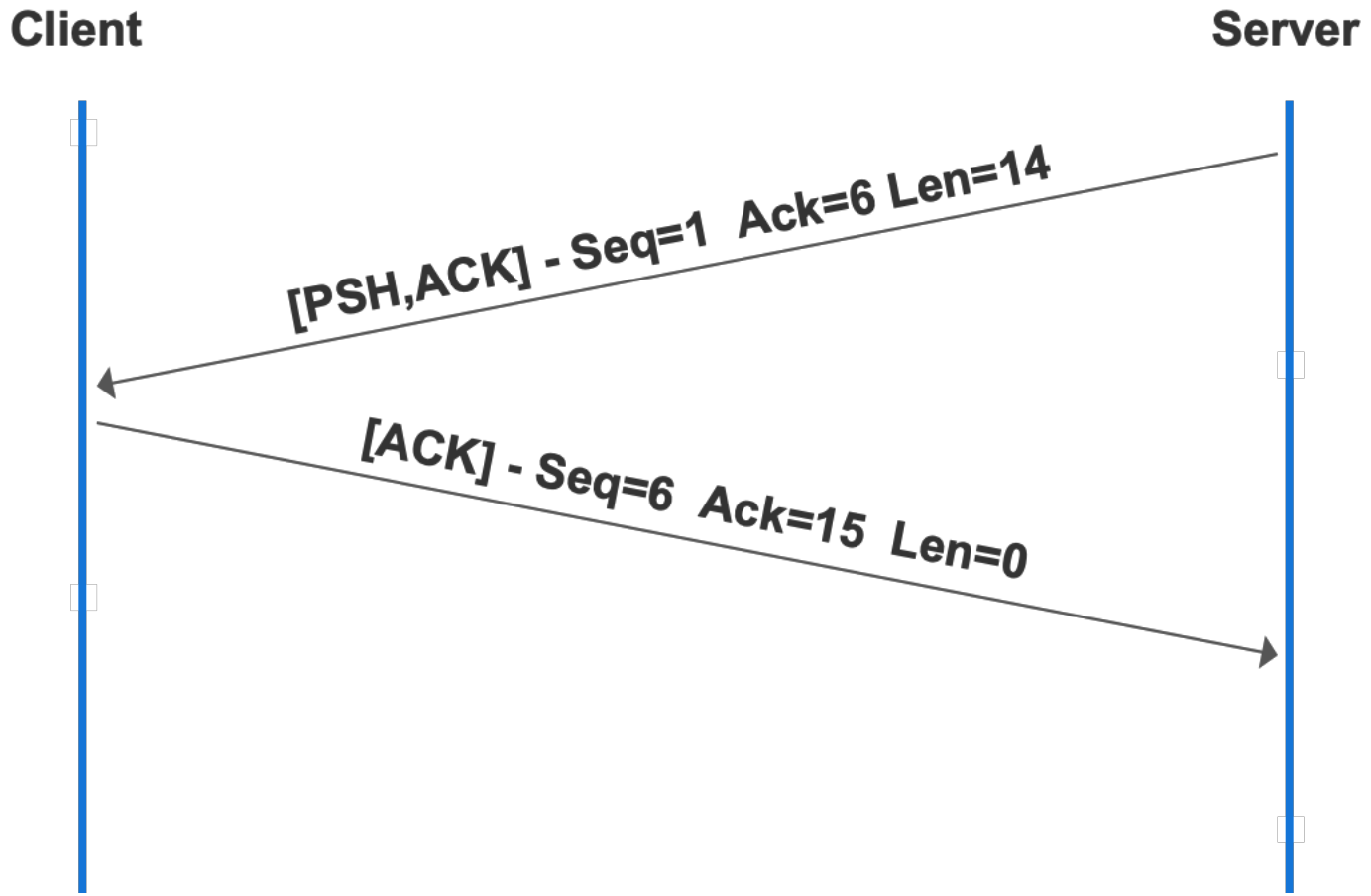
Modello TCP / IP – Livello di Trasporto - TCP

- Il client invia il messaggio "Hello", composto da 5 bytes

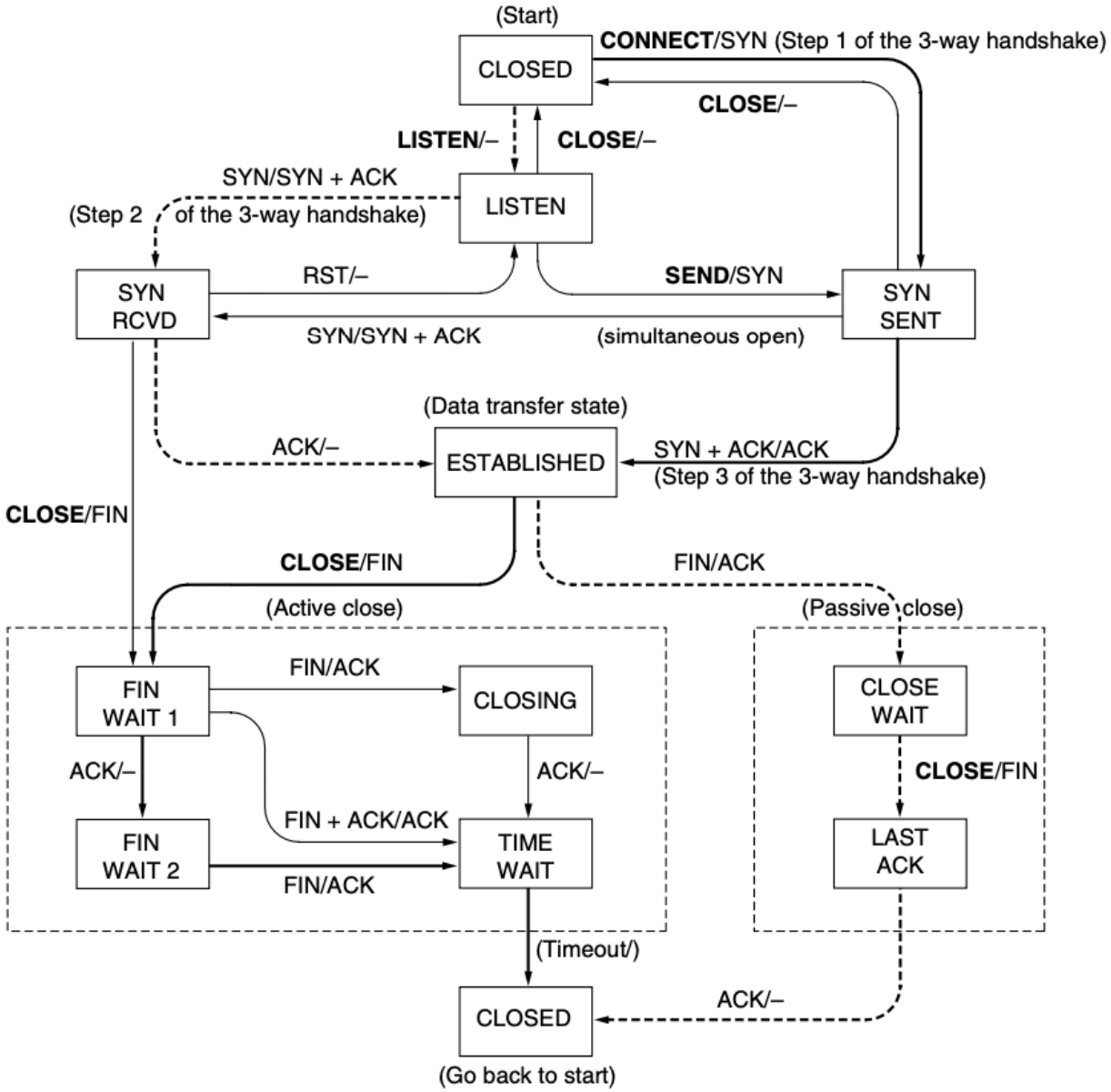


Modello TCP / IP – Livello di Trasporto - TCP

- Il server risponde con il messaggio "Echo di: Hello", composto da 14 bytes



Modello TCP / IP – Livello di Trasporto - TCP



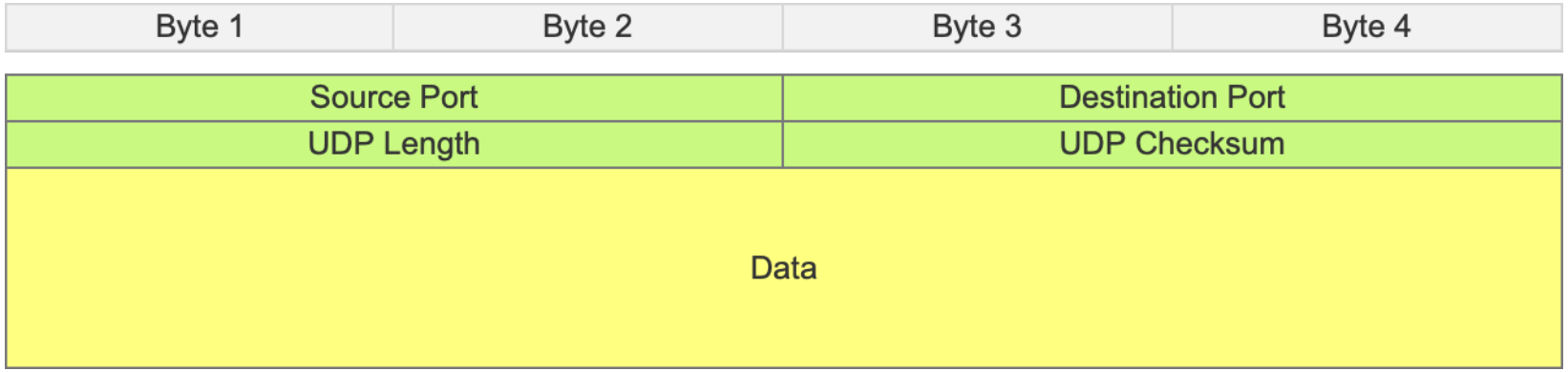
Modello TCP / IP – Livello di Trasporto – Numeri di porta

- Le porte disponibili sono 65535, in quanto il campo del pacchetto è a 16 bit.
- Le porte sono divise in tre categorie:
  - Numeri riservati – Da 1 a 1023: Possono essere utilizzati solamente dai server a fronte di servizi standardizzati. Per esempio la porta 80 è quella in cui i server HTTP si mettono in ascolto
  - Numeri Registrati – Da 1024 a 49151: Sono usati da alcuni server ma possono essere utilizzati anche dai client
  - Numeri Liberi – Da 49152 a 65535: Sono liberamente utilizzabili dai client

#	Protocol		Servizio
21	FTP-CONTROL	File Transfer Protocol	Trasferiento file (control)
20	FTP-DATA	File Transfer Protocol	Trasferimento files (dati)
23	TELNET		Accesso via terminale
25	SMTP		Trasferimento di posta elettronica
53	DNS	Domain Name System	Accesso al DNS
80	HTTP		Web server
109	POP2	Post Office Protocol (Version 2)	Lettura posta elettronica
22	SSH	Secure Socket	Accesso via terminale cifrato
110	POP3	Post Office Protocol (version 3)	Lettura posta elettronica
137	NETBIOS Name Service.		Servizio di rete per applicazioni in ambiente DOS (Windows)
138	NETBIOS Datagram Service.		
139	NETBIOS Session Service.		
443	HTTPS	HTTP over SSL/TLS	WEB cifrato

Modello TCP / IP – Livello di Trasporto - UDP

- Il protocollo UDP (User Datagram Protocol) è un protocollo connectionless che non prevede controlli sulla trasmissione.
- Si limita ad inviare pacchetti tra gli host, lasciando al livello applicativo eventuali controlli e gestioni del flusso
- È il modo più semplice con cui una applicazione può inviare un datagramma IP
- I pacchetti inviati tramite UDP sono anche detti segmenti



Modello TCP / IP – Livello di Trasporto - UDP

- Source e Destination port: Numeri di porta del mittente e del destinatario. 16 bit ciascuno, pertanto le porte massimo sono 65536.
- UDP Length: Lunghezza dell'intero segmento UDP (header + dati). Il valore minimo è 8 byte (dimensione dell'header).
- UDP CheckSum: Controllo dell'errore. Viene calcolato con un algoritmo particolare che tiene in considerazione anche alcuni campi del datagramma IP

Modello TCP / IP – Protocollo NAT

Il numero di indirizzi IP è limitato e non è sufficiente per assegnare un indirizzo IP differente (IP dedicato) ad ogni dispositivo.

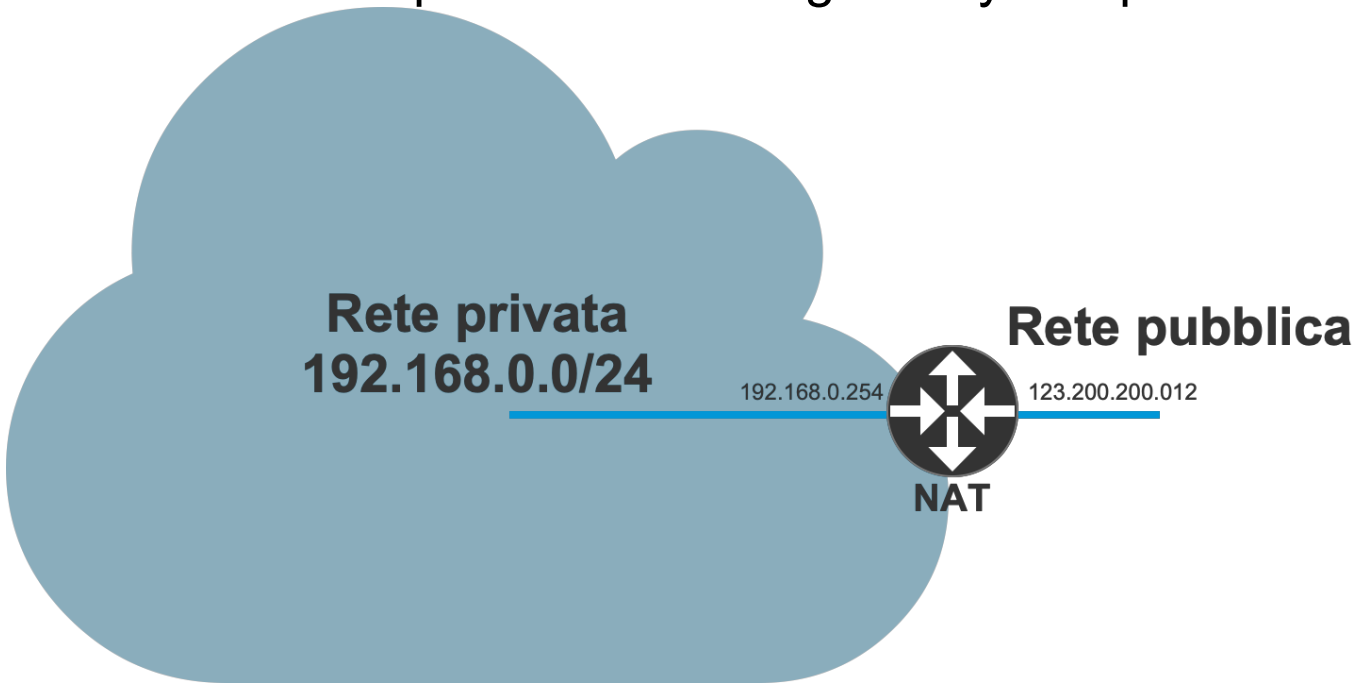
Sono pertanto state definite modalità per ovviare al problema.

1. Assegnando IP dinamici: Quando un dispositivo vuole accedere collegarsi ad internet riceve un IP. Quando il dispositivo si scollega quell'IP viene assegnato ad un altro host.
2. Utilizzando IP privati: Creando Network IP utilizzando i range di IP privati, garantendo però che vengano "rimossi" dai datagrammi inoltrati verso gli IP pubblici.
    1. Il protocollo NAT (Network Address Translation) si occupa di questo compito

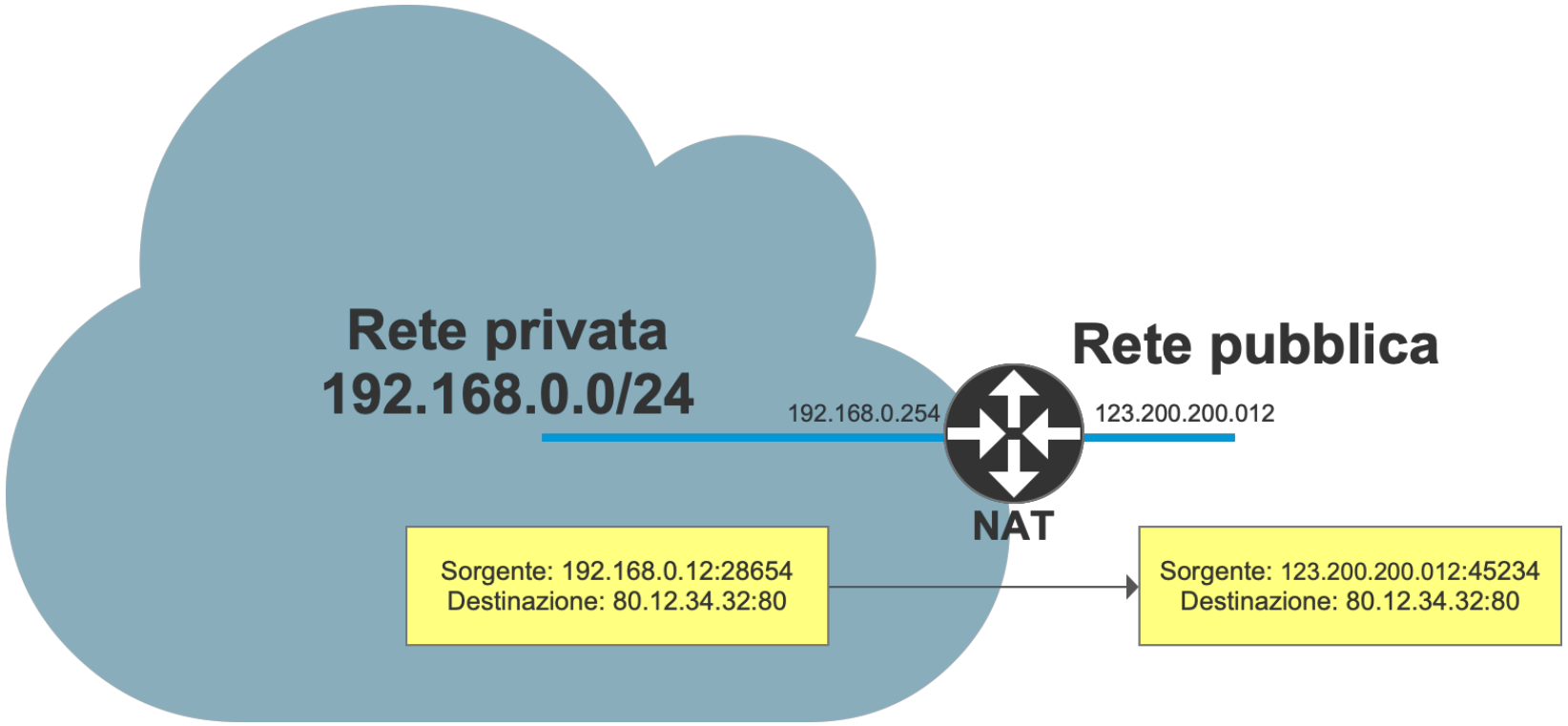
Modello TCP / IP – Protocollo NAT

Il NAT deve sostituire gli indirizzi privati quando i datagrammi vengono inoltrati verso IP pubblici. Pertanto il ruolo deve essere affidato ad un gateway, che si interpone tra una rete privata e una rete pubblica (o altre rete privata).

La sostituzione avverrà con l'indirizzo pubblico di cui il gateway è in possesso.



Modello TCP / IP – Protocollo NAT



Modello TCP / IP – Protocollo NAT

Il NAT deve essere in grado di consegnare all'host interno i pacchetti che arrivano all'indirizzo pubblico. Deve essere cioè possibile risalire all'IP interno per il quale sta arrivando il datagramma.

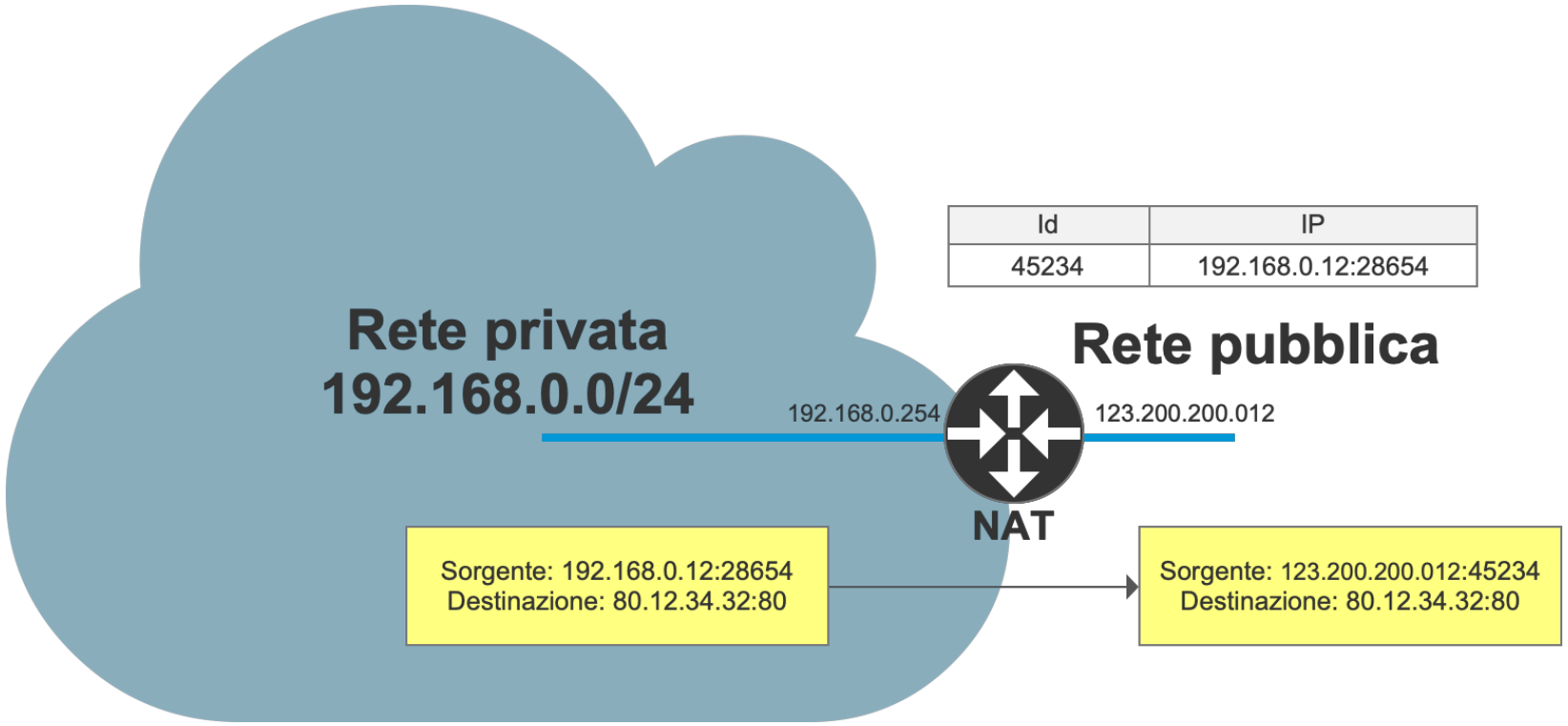
NAT mantiene una tabella di associazione con i seguenti campi:

- Id
- Indirizzo host interno (ip privato + porta)

Quando i pacchetti sono in uscita NAT inserisce nel numero di porta l'id della tabella di corrispondenza.

Quando arriva un pacchetto in entrata NAT preleva il numero di porta e lo usa per risalire all'host

Modello TCP / IP – Protocollo NAT



ITS Academy
Adriano
Olivetti

TCP / IP - Livello di Trasporto

HANDS ON SESSION

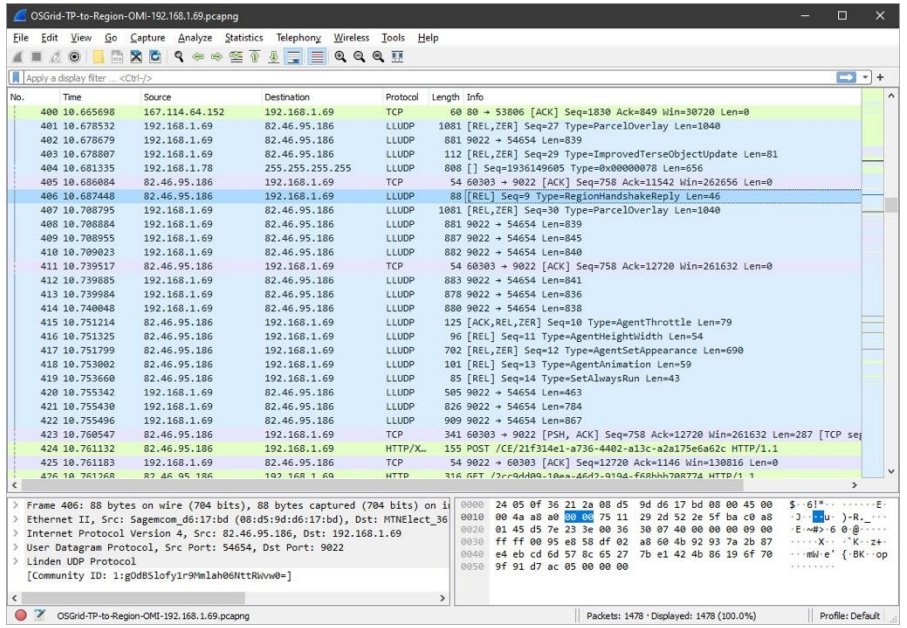
Wireshark

Per analizzare il traffico di rete TCP / UDP utilizzeremo Wireshark, il più utilizzato network analyzer con interfaccia utente (https://www.wireshark.org)

È open source e disponibile per Windows, Mac, Linux (è preinstallato in Kali)

È uno strumento estremamente potente ma anche estremamente semplice nel suo utilizzo base:

- Scelgo l'interfaccia di rete su cui catturare il traffico
- Avvio la cattura
- Analizzo in real-time il traffico catturato
- Termino la cattura quando ho finito



Client e Server TCP in Python

Useremo un semplice Echo Server, e relativo client, scritti in phyton. Potete scaricarli da Classroom o clonando il repository https://github.com/dluppoli/Demo-TCP-UDP

```
import socket
HOST = '127.0.0.1'  # Loopback address
PORT = 65432  # Porta arbitraria non utilizzata dal sistema operativo

# Socket TCP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        # Associazione a indirizzo e porta
        server_socket.bind((HOST, PORT))
        # Avvio server in ascolto
        server_socket.listen()

        print(f"Server TCP in ascolto su {HOST}:{PORT}")

        # Connessione in entrata
        conn, addr = server_socket.accept()
        with conn:
            print(f"Connesso a {addr}")

            while True:
                # Ricezione dati dal client
                data = conn.recv(1024)
                if not data:
                    break

                print(f"Ricevuto: {data.decode()}")

                # Invio messaggio di Echo
                conn.sendall(("Echo di: " + data.decode()).encode())
```

```
import socket

HOST = '127.0.0.1'  # Loopback address
PORT = 65432        # La stessa porta su cui il server è in ascolto

# Creazione socket TCP/IP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
        # Connessione al server
        client_socket.connect((HOST, PORT))

        while True:
            # Lettura messaggio da inviare
            message = input("Inserisci il messaggio da inviare (o 'exit' per uscire): ")

            if message.lower() == 'exit':
                break

            # Invio dati al server
            client_socket.sendall(message.encode())

            # Ricezione risposta dal server
            data = client_socket.recv(1024)

            # Stampa risposta
            print(f"Risposta dal server: {data.decode()}")
```

Client e Server UDP in Python

Echo Server realizzato utilizzando UDP

```
import socket
HOST = '127.0.0.1'
PORT = 65434

# Socket UDP
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_socket:
        # Associazione a indirizzo e porta
        server_socket.bind((HOST, PORT))

        print(f"Server UDP in ascolto su {HOST}:{PORT}")

        while True:
                # Ricezione dati dal client
                data, client_address = server_socket.recvfrom(1024)
                print(f"Ricevuto: {data.decode()} da {client_address}")

                # Invio messaggio di Echo
                server_socket.sendto(("Echo di: " + data.decode()).encode(),
client_address)
```

```
import socket
HOST = '127.0.0.1'
PORT = 65434

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as client_socket:

        while True:
                # Lettura messaggio da inviare
                message = input("Inserisci il messaggio da inviare (o 'exit' per uscire)
")

                if message.lower() == 'exit':
                        client_socket.close()
                        break

                # Invio dati al server
                client_socket.sendto(message.encode(), (HOST, PORT))

                # Ricezione risposta dal server
                data, server_address = client_socket.recvfrom(1024)

                # Stampa risposta
                print(f"Ricevuto: {data.decode()} da {server_address}")
```

NMAP

NMAP è un software open source per effettuare scansioni di rete, analizzandone i risultati.
Alcune funzionalità:

- Identificazione degli host presenti sulla rete (scansione a livello IP)
- Identificazione delle porte aperte su ogni host (scansione a livello TCP / UDP)
- Fingerprint dei servizi esposti sulle porte aperte
- Ricerca vulnerabilità note su servizi identificati
- Fingerprint del sistema operativi utilizzato
- Alcune tipologie di attacchi (p.e. bruteforce su servizi con password)
- …

NMAP

La sintassi di base è: `nmap <parametri> <rangeIp>`

Il range di IP da scansionare può essere fornito in più modi:

- Elenco di singoli IP separato da spazi → 192.168.2.3 192.168.2.7
- Range di indirizzi IP consegutivi → 192.168.2.3-9
- Tutti gli ip di una rete, usando la notazione CIDR → 192.168.2.0/24
- Importandolo da un file di testo → tramite il parametro -iL

NMAP

I parametri di utilizzo sono molteplici. I più usati sono:

- Selezione delle porte da scansionare:
  - Singola porta –p 80
  - Porte multiple –p 80,443 oppure –p 0-1024 oppure –p-
  - Porte principali (circa 100) –F
- Modalità di scansione principali:
  - Scansione ICMP (ping) –sn
  - Scansione TCP full handshake –sT
  - Scansione UDP -sU
- Velocità di scansione, da –T1 a –T4 con:
  - T1 modalità più lenta (e più accurata) e T4 modalità più rapida (e meno accurata)
- Specifica del file di output: -oX <nomefile>

NMAP

NMAP è uno strumento "molto rumoroso" le cui attività vengono tipicamente identificate dai sistemi di protezione (quali IDS / IPD).

Per diminuire il "rumore" sono disponibili altre tipologie di scansione TCP

- Half tcp scan (SYN): -sS
- Null scan: -sN
- FIN scan: -sF
- Xmas scan: -sX

NMAP

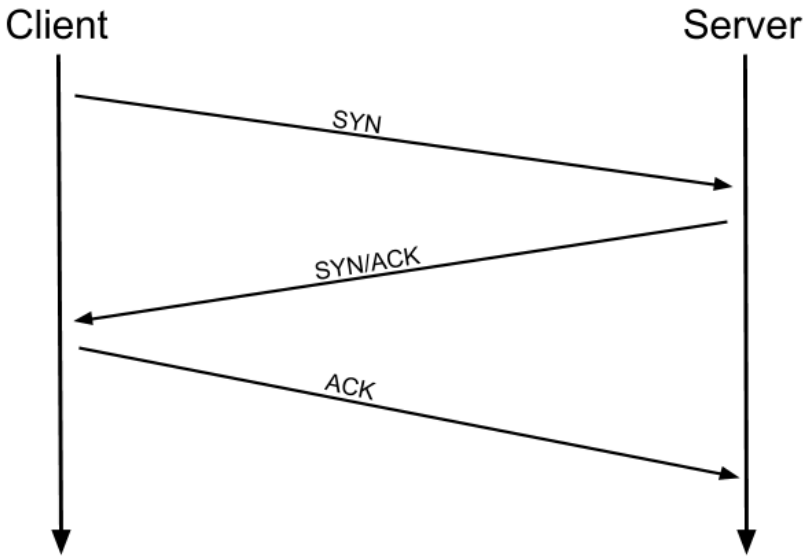
Altre opzioni:

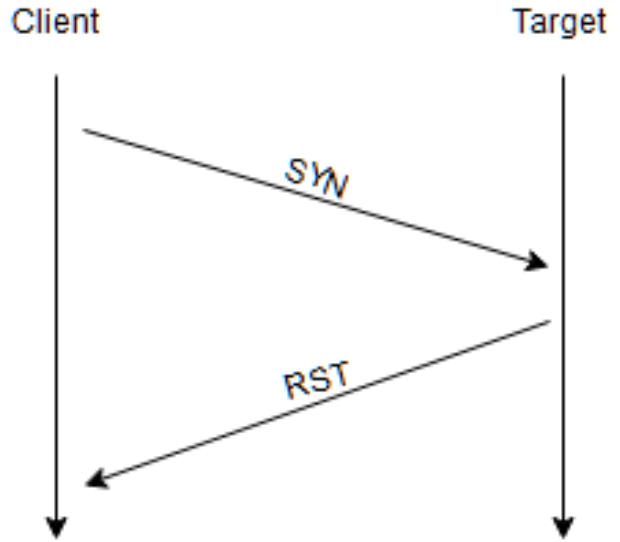
- Verbosità dell'output –v –vv
- Os Detection –O
- Service version Detection –sV
- Vulnerabilità note –script vuln
- Aggressive (Os detection + Service version + Vuln) -A

NMAP– Full TCP Scan

Nella scansione full TCP la connessione alla porta oggetto del test è completa, ovvero copre l'intero 3-way handshake.

In caso di porta aperta l'host risponderà con il flag SYN/ACK mentre nel caso di porta chiusa risponderà con il flag RST

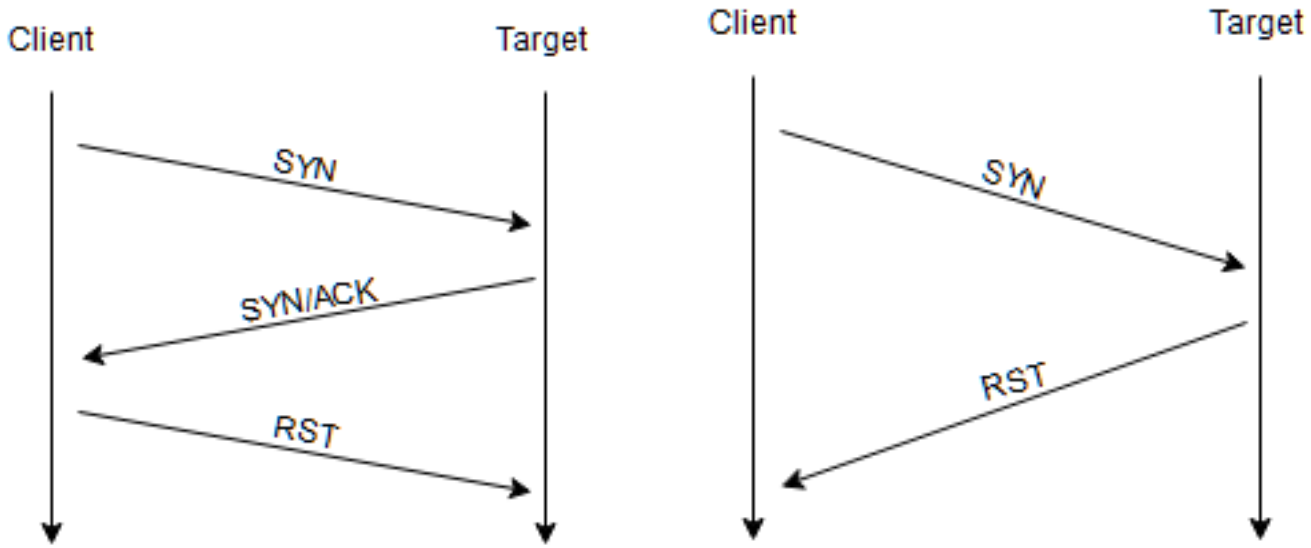




NMAP– Half TCP Scan

Nella scansione half TCP, anche detta SYN scan, la connessione viene abortita dopo aver ricevuto il pacchetto SYN/ACK. Questa modalità ha alcuni vantaggi:

- Permette di bypassare alcuni sistemi IDS/IPS
- Diminuisce la probabilità che le azioni vengano loggata. Questa modalità è anche detta stealth
- È più rapida rispetto al full scan



Playgroung

- Nmap (https://tryhackme.com/room/furthernmap)
- Nmap live host discovery (https://tryhackme.com/room/nmap01)
- Nmap basic port scan (https://tryhackme.com/room/nmap02)
- Nmap advanced port scan (https://tryhackme.com/room/nmap03)
- Nmap post port scan (https://tryhackme.com/room/nmap04)