



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Protocolli di Trasporto

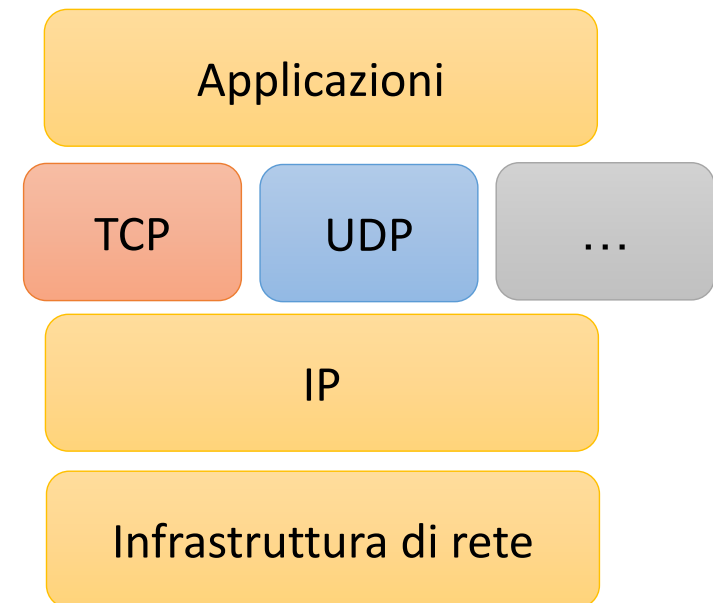
Franco CALLEGATI

Dipartimento di Informatica: Scienza e Ingegneria

Trasporto in Internet

- Architettura protocollare tradizionale di Internet:

- Tipologia di servizio: dati
- Due protocolli di trasporto:
 - TCP: connection oriented
 - UDP: connectionless



- Nuovi servizi multimediali:

- Emergono problematiche di trasporto real time
- Vengono definiti nuovi protocolli di trasporto
 - RTP, RTCP



Funzioni dello strato di trasporto

- Consente la **Multiplazione**:
 - Permette a più processi applicativi di utilizzare le sue funzioni di comunicazione in contemporanea
- Utilizza il numero di porta per distinguere flussi dati di applicazioni diverse
- Controlla il comportamento del canale di comunicazione end-to-end
 - L'obiettivo è quello di garantire la qualità del trasferimento dati a livello di trasporto richiesto dall'applicazione



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

User Datagram Protocol



User Datagram Protocol (UDP)

- Protocollo “connectionless”
 - Non esiste il concetto di “connessione”
 - Ogni messaggio è indipendente da tutti gli altri
- Pensato per
 - Invio di blocchi dati di limitate dimensioni
 - Comunicazione fra applicazioni che non richiede un controllo della qualità del trasporto
 - Esempi: e-mail, DNS, ...

Il messaggio UDP

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.10.199	137.204.59.1	DNS	Standard query A deisnet.deis.unibo.it
2	0.001714	137.204.59.1	192.168.10.199	DNS	Standard query response CNAME deis85.deis.unibo.it A 1
3	0.002763	192.168.10.199	137.204.57.85	TCP	nim > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_P
4	0.003487	137.204.57.85	192.168.10.199	TCP	http > nim [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1
5	0.003512	192.168.10.199	137.204.57.85	TCP	nim > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
6	0.003732				
7	0.005748				
8	0.009821				
9	0.011087				
10	0.011148				
11	0.014612				
12	0.015858				

1 0.000000 192.168.10.199 137.204.59.1 DNS Standard query A deisnet.deis.unibo.it

Frame 1: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)

Ethernet II, Src: DellComp_89:b3:e9 (00:06:5b:89:b3:e9), Dst: DellComp_ec:46:62 (00:b0:d0:ec:46:62)

Internet Protocol, Src: 192.168.10.199 (192.168.10.199), Dst: 137.204.59.1 (137.204.59.1)

User Datagram Protocol, Src Port: startron (1057), Dst Port: domain (53)

Source port: startron (1057)

Destination port: domain (53)

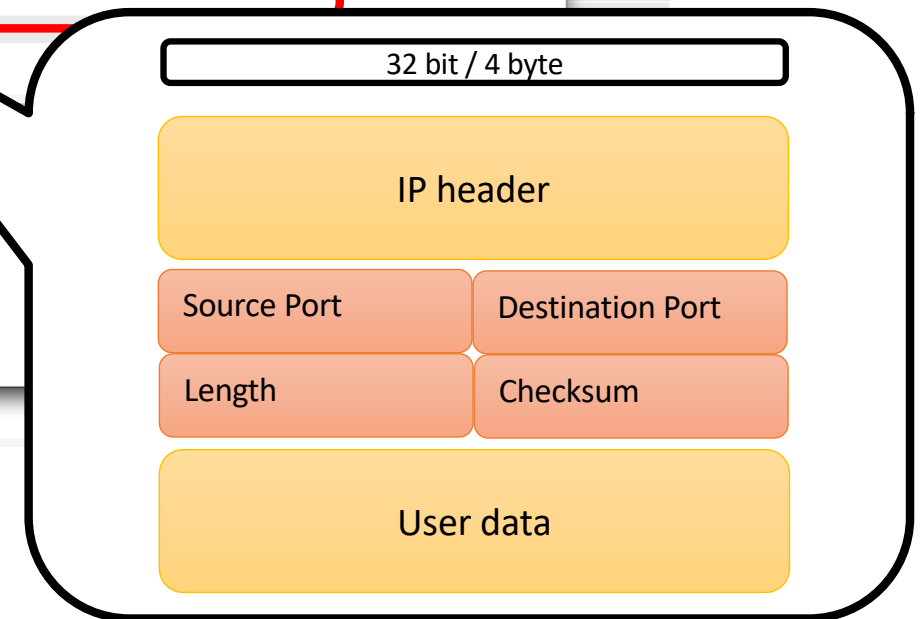
Length: 47

Checksum: 0x17eb [validation disabled]

Domain Name System (query)

```

0000  00 b0 d0 ec 46 62 00 06 5b 89 b3 e9 08 00 45
0010  00 43 01 83 00 00 80 11 00 00 c0 a8 0a c7 89
0020  3b 01 04 21 00 35 00 2f 17 eb 76 46 01 00 00 01
0030  00 00 00 00 00 00 07 64 65 69 73 6e 65 74 04 64
0040  65 69 73 05 75 6e 69 62 6f 02 69 74 00 00 01 00
0050  01
  
```





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Transmission Control Protocol



Il segmento TCP

- TCP incapsula i dati delle applicazioni in pacchetti detti “**segmenti**”
- Il segmento TCP prevede
 - Un header standard di 20 byte
 - Un header variabile per negoziare delle opzioni
 - Un payload di dimensione variabile contenente i dati di applicazione
- Il segmento TCP ha una dimensione massima detta Maximum Segment Size (MSS)
 - MSS corrisponde alla massima dimensione del blocco dati di applicazione che può essere contenuto nel segmento



Formato del segmento TCP (1)

32 bit

Source Port							Destination Port						
Sequence number													
Acknowledge number													
TCP header length	Reserved				U R G	A C K	P S H	R S S	S Y N	F I N	Window		
Checksum							Urgent Pointer						
Opzioni									Padding				
Dati													



Formato del segmento TCP (2)

- **Source (Destination) port:** numero della porta sorgente (destinazione)
- **Sequence number:** numero di sequenza del primo byte del pacchetto; se è presente il bit SYN questo è il numero di sequenza iniziale su cui sincronizzarsi
- **Acknowledge number:** se il bit ACK è a 1 allora questo numero contiene il numero di sequenza del blocco di dati che il ricevitore si aspetta di ricevere
- **TCP Header Length (4 bit):** numero di parole di 32 bit dell'intestazione TCP; indica dove iniziano i dati
- **Reserved:** sei bit riservati per uso futuro

Formato del segmento TCP (3)

- **Control bit:** sono 6 bit di controllo
 - **URG** posto a 1 se si deve considerare il campo Urgent Pointer
 - **ACK** posto a 1 se si deve considerare il campo Acknowledge
 - **PSH** posto a 1 serve per la funzione di push
 - **RST** posto a 1 per resettare la connessione
 - **SYN** posto a 1 per sincronizzare i numeri di sequenza
 - **FIN** posto a 1 per indicare la fine dei dati





Formato del segmento TCP (4)

- **Window:** finestra del ricevitore, cioè il numero di byte che il ricevitore è disposto a ricevere, partendo dal numero di sequenza di quello contenuto nel campo acknowledge
- **Checksum:** controllo dell' errore sul segmento
- **Urgent Pointer:** contiene puntatore a dati urgenti eventualmente presenti nel pacchetto (es. per abortire programma remoto in esecuzione), ha senso se il bit URG è posto ad 1
- **Options:** contiene opzioni per la connessione
- **Padding:** bit aggiuntivi per fare in modo che l' intestazione sia multipla di 32 bit



Dimensioni del segmento TCP

- MSS deve
 - Essere inferiore alla massima dimensione del payload IP meno un header TCP
 - $2^{16} = 65535 - 20 - 20 = 65495$ byte
 - Rispettare i limiti imposti ai pacchetti dalle reti che deve attraversare e che hanno una Maximum Transfer Unit (MTU)
 - Un tipico valore per MTU sono i 1500 byte imposti da Ethernet
- MSS dipende dall'implementazione
 - Normalmente è configurabile
- In generale non è possibile sapere la MTU di ogni rete intermedia che va attraversata
 - La rete ha il meccanismo per la frammentazione dei pacchetti
 - Questo può condurre a inefficienza, soprattutto in caso di una grande quantità di dati da trasmettere su una connessione
 - È stato definito un algoritmo detto "Path MTU discovery" (RFC 1191) basato su ICMP



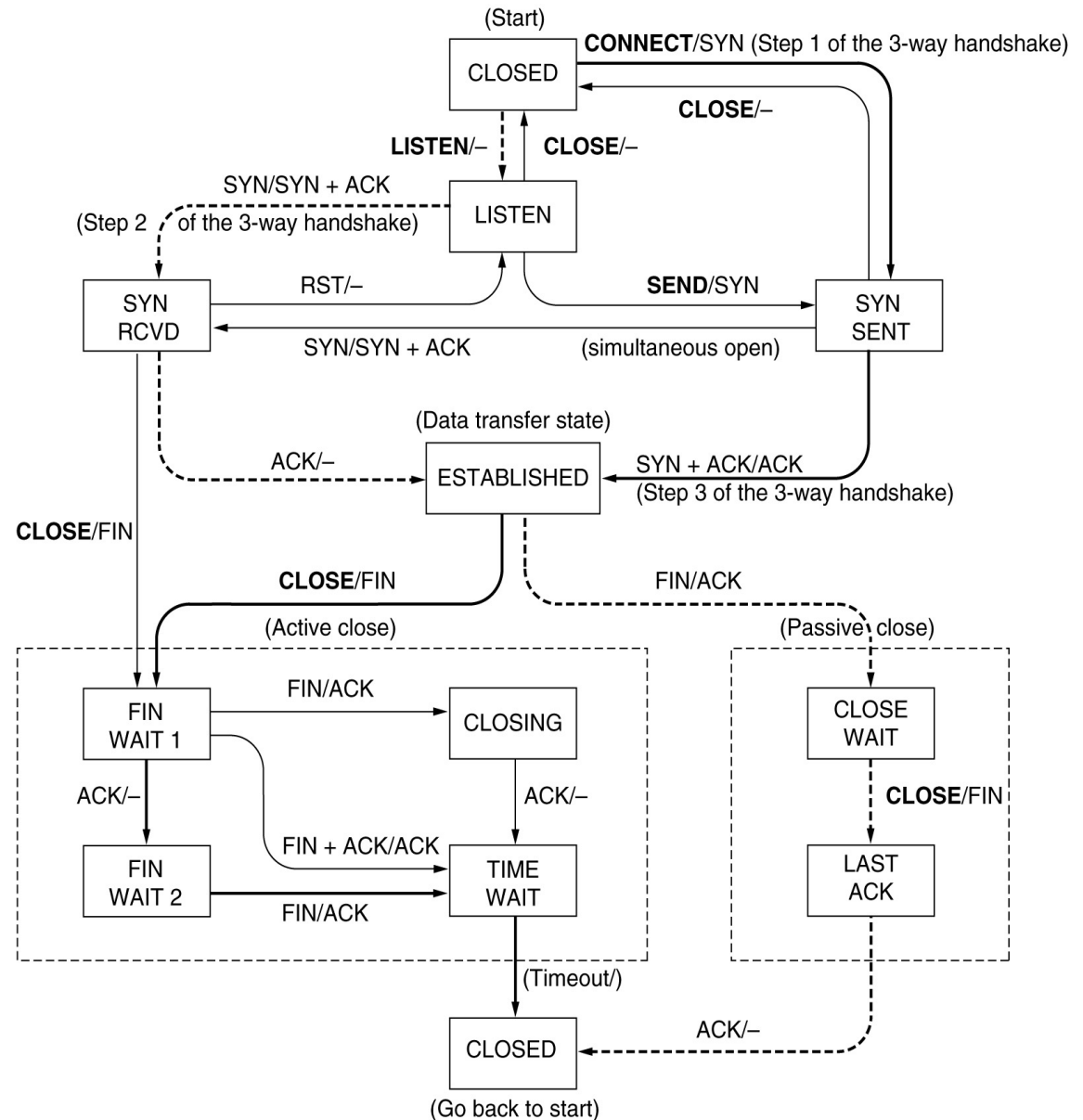
Checksum

- Viene calcolato applicando Internet Checksum a
 - Pseudo-header (psedo-intestazione)
 - Indirizzi IP sorgente e destinatario
 - Protocol
 - Lunghezza in byte del segmento TCP (payload IP)
 - Non viene trasmessa ma viene calcolata in trasmissione e in ricezione
 - Intestazione TCP
 - Con campo checksum posto a 0
 - Dati del segmento TCP
 - Se il numero di byte è dispari viene aggiunto un byte di padding con tutti i bit a 0
 - Il padding non viene trasmesso

La macchina a stati finiti del TCP

- Linee tratteggiate
 - Azioni tipiche di un server
- Linee nere
 - Azioni tipiche di un client
- Linee chiare
 - Eventi inusuali
- Transizioni
 - Causa/effetto

Da A.S. Tanenbaum, “Reti di Calcolatori”





Definizioni

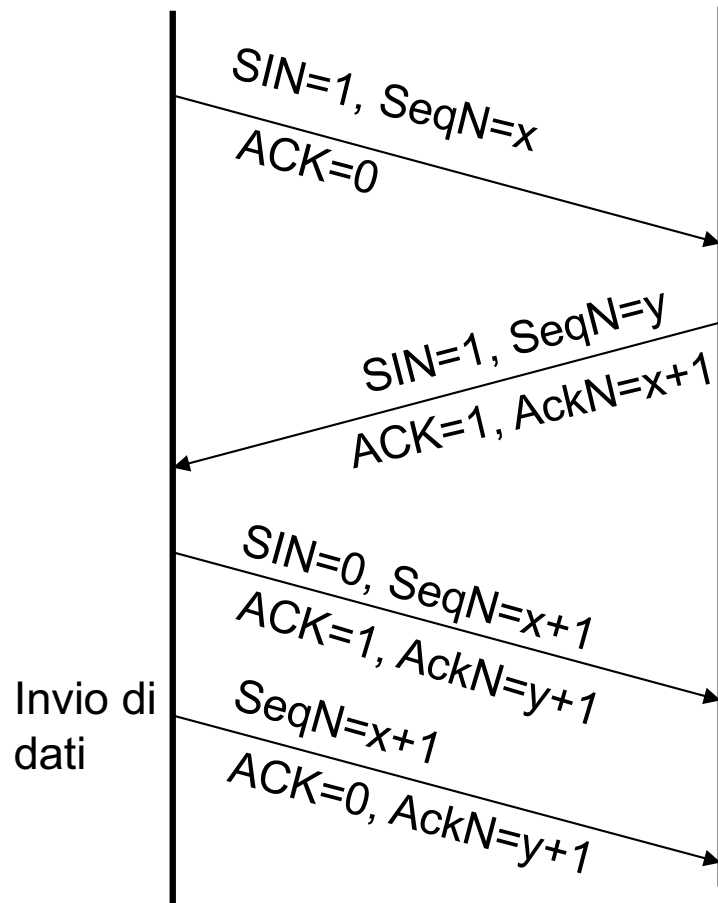
- **Stato:** Le condizioni che descrivono il software del protocollo in un particolare calcolatore in un determinato istante.
- **Transizione:** Il passaggio da uno stato ad un altro.
- **Evento:** Qualunque cosa che provochi una transizione di stato per il protocollo.
- **Azione:** Qualcosa che il software del protocollo fa in un dispositivo come risposta ad un evento prima di effettuare un transizione di stato



Dialogo su rete inaffidabile

- Se il mezzo di comunicazione è inaffidabile risulta sostanzialmente impossibile avere uno scambio di informazioni con conferma certa
 - Problema logico delle 3 armate
 - A invia un messaggio e B lo conferma
 - Se A non riceve la conferma non può sapere se B abbia ricevuto il messaggio o meno
 - Perdita del messaggio o della conferma?
 - Il ragionamento si può proseguire sulla conferma della conferma ecc.
- È necessario decidere dove fermarsi per raggiungere un determinato grado di affidabilità

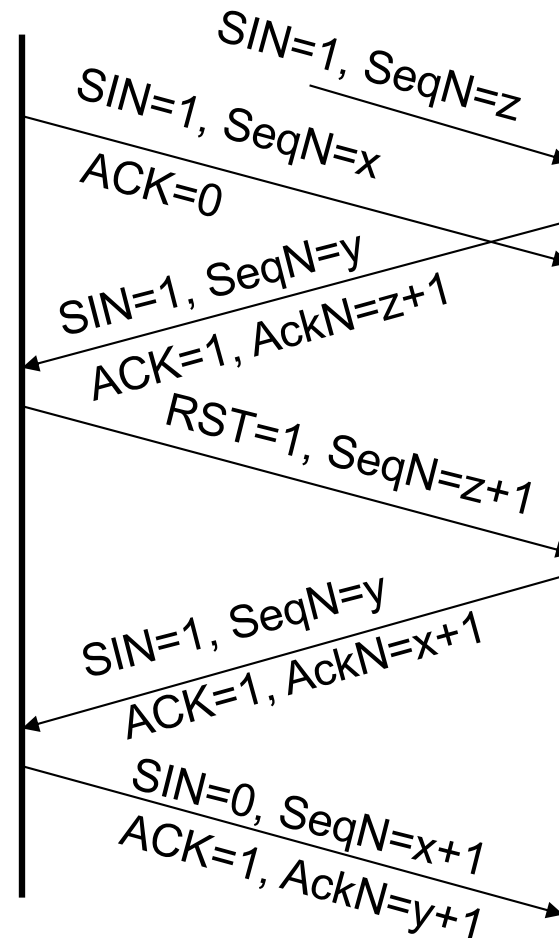
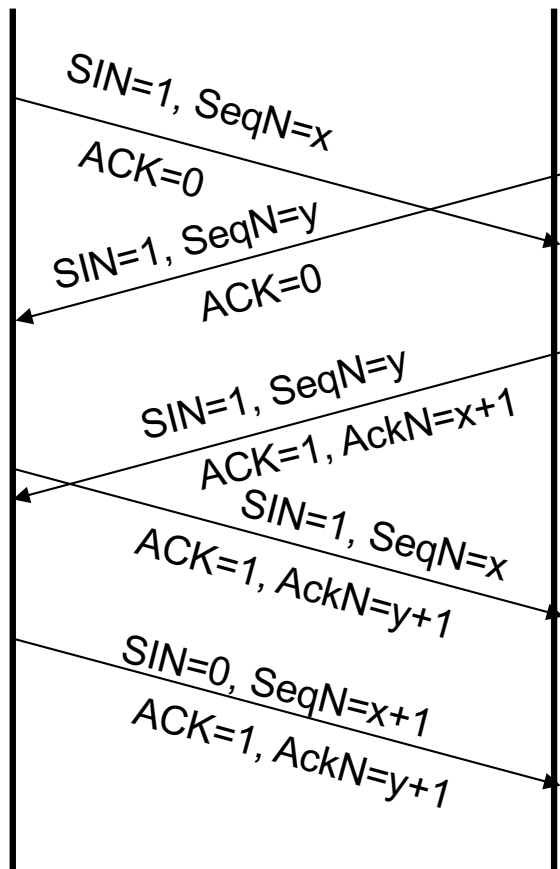
Apertura della connessione TCP



- L'apertura della connessione è critica
 - I segmenti di segnalazione possono essere persi, duplicati e ritardati
- **Three ways handshake**
 - Si è dimostrato molto robusto alla prova dei fatti
- Utilizza sinergicamente i bit di flag e quelli di numerazione
 - Si noti che il primo pacchetto dati ha numero di sequenza uguale all'ACK precedente (ACK non occupa spazio di numerazione)

Caratteristiche del TWH

- Il three-ways handshake
 - resiste alla instaurazione contemporanea di due connessioni
 - ignora pacchetti di apertura ritardatari



Il problema delle «incarnazioni»

- Connessione fra Host A e Host B
 - Host A viene riavviato
 - La connessione viene di fatto chiusa in modo errato
 - La connessione rimane attiva in Host B
- Dopo il riavvio l' applicazione vuole riprendere il dialogo e riprova ad aprire la connessione con Host B
 - Cosa accade se decide di utilizzare i medesimi numeri di porta (ossia riavviare la medesima connessione)
- Esistono a breve distanza di tempo diverse “**incarnazioni**” della medesima connessione logica



Ripresa dopo un problema

- Il three ways handshake per una nuova incarnazione
 - L'host B ritiene che la connessione precedente sia ancora aperta
 - L'ambiguità viene risolta

TCP A		TCP B
1. (CRASH)		(send 300, receive 100)
2. CLOSED		ESTABLISHED
3. SYN-SENT -->	<SEQ=400><CTL=SYN>	--> (??)
4. (!!) <--	<SEQ=300><ACK=100><CTL=ACK>	<-- ESTABLISHED
5. SYN-SENT -->	<SEQ=100><CTL=RST>	--> (Abort!!)
6. SYN-SENT		CLOSED
7. SYN-SENT -->	<SEQ=400><CTL=SYN>	-->



Chiusura della connessione TCP

- **Soft release**
 - Il TCP cerca di realizzare la chiusura ordinata della connessione, garantendo che non vadano persi dati
- Anche questo problema non può essere risolto in modo rigoroso su una rete inaffidabile
- TCP sceglie di realizzare la chiusura con modalità “simplex”
 - Le due direzioni vengono rilasciate in modo **indipendente**
 - Il TCP che intende terminare la trasmissione emette un segmento con **FIN=1**
 - Quando questa entità riceve l' Ack la direzione si considera chiusa
 - Se dopo un certo tempo non arriva l' Ack il mittente del FIN rilascia comunque la connessione
 - L' altra direzione può continuare a trasmettere dati finché non decide di chiudere



Esempio di chiusura normale

- TCP A decide di chiudere
 - Inizia la procedura inviando un segmento con FIN=1
- TCP B rileva la richiesta di chiusura
 - Procede anche lui all' invio di un segmento con FIN=1

TCP A			TCP B		
1.	ESTABLISHED			ESTABLISHED	
2.	(Close)				
	FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN,ACK>	-->	CLOSE-WAIT	
3.	FIN-WAIT-2	<-- <SEQ=300><ACK=101><CTL=ACK>	<--	CLOSE-WAIT	
4.				(Close)	
	TIME-WAIT	<-- <SEQ=300><ACK=101><CTL=FIN,ACK>	<--	LAST-ACK	
5.	TIME-WAIT	--> <SEQ=101><ACK=301><CTL=ACK>	-->	CLOSED	
6.	(2 MSL)				
	CLOSED				



Chiusura contemporanea

- TCP A e B decidono in contemporanea di chiudere la connessione
 - La procedura funziona correttamente anche in questo caso

TCP A		TCP B
1. ESTABLISHED		ESTABLISHED
2. (Close)		(Close)
FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN,ACK>	... FIN-WAIT-1
	<-- <SEQ=300><ACK=100><CTL=FIN,ACK>	<--
	... <SEQ=100><ACK=300><CTL=FIN,ACK>	-->
3. CLOSING	--> <SEQ=101><ACK=301><CTL=ACK>	... CLOSING
	<-- <SEQ=301><ACK=101><CTL=ACK>	<--
	... <SEQ=101><ACK=301><CTL=ACK>	-->
4. TIME-WAIT		TIME-WAIT
(2 MSL)		(2 MSL)
CLOSED		CLOSED



Svolgimento del dialogo

- Protocollo ARQ per rendere affidabile il dialogo
- Rispetto ai protocolli di linea TCP affronta un problema molto più complesso
 - Le caratteristiche dei ricevitori possono essere estremamente variabili
 - I segmenti sono trasmessi utilizzando una rete connectionless e quindi non è garantito l'arrivo in sequenza
 - I percorsi in rete e le condizioni di congestione possono variare e quindi il ritardo di propagazione non è costante
 - La banda del canale dipende dal percorso



Lo stato ESTABLISHED

- Nello stato si trasferiscono i dati utilizzando il protocollo ARQ
 - Si numerano sequenzialmente i segmenti
 - Si conferma la corretta ricezione dei segmenti
 - Il recupero degli errori avviene tramite ritrasmissione
 - Il meccanismo di ritrasmissione è una sorta di Go-back-N modificato

Numerazione in TCP

- Per avere la massima **flessibilità** si sceglie di assegnare un numero non ai segmenti ma **ai singoli byte** trasportati nei segmenti
 - I dati trasportati sono pensati come un **unico flusso (stream)** di byte (**byte stream**)
 - Si comincia a numerare da un numero x scelto all'atto dell'apertura della connessione
 - Il campo Seq. number numera il primo byte del segmento
- La conferma di avvenuta ricezione viene data mettendo nel campo **Ack. Number** il numero del byte successivo all'ultimo ricevuto
 - primo byte che ci si aspetta di ricevere

Numerazione duplicata

- Possono facilmente aversi segmenti ritardati o duplicati
 - All'istante t_0 viene inviato il segmento S_0 con numero di sequenza X
 - S_0 viene duplicato nella rete
 - Una copia viene correttamente ricevuta e confermata
 - Una copia viene ritardata rispetto alle altre
 - La trasmissione continua
 - Si esaurisce lo spazio di numerazione e quindi si riutilizzano numeri già usati
 - All'istante $t_k > t_0$ viene inviato il segmento S_k che ha anch'esso numero di sequenza X
 - La copia ritardata di S_0 arriva poco prima di S_k
 - viene interpretata come segmento valido



Maximum Segment Lifetime

- I numeri di sequenza possono essere riutilizzati?
 - Solo se si è sicuri che non esistano più in rete vecchi segmenti numerati con tali numeri
- Lo spazio di numerazione finito rende necessario limitare il tempo di vita dei segmenti
- Il massimo tempo di vita dei segmenti (Maximum Segment Lifetime o MSL) deve essere noto
 - $MSL = 2 \text{ min}$ (RFC 793)
 - Su di un collegamento a 2 Megabits/sec servono 4.5 ore per esaurire lo spazio di numerazione di 2^{32} ottetti
 - Su di un collegamento a 100 Megabits/sec servono 5.4 minuti per esaurire lo spazio di numerazione di 2^{32} ottetti
 - Cosa accade per reti con velocità del Gbit/s e oltre?



Inizializzazione della sequenza

- All'apertura della connessione si deve scegliere il numero di sequenza iniziale (Initial Sequence Number o ISN)
 - Numero prefissato uguale per tutti
 - Numero puramente casuale
 - Numero legato al valore di un contatore
- ISN
 - Deve garantire che non ci sia duplicazione nell'uso dei numeri di sequenza
 - Qualora non sia prefissato e costante deve essere concordato fra i due host che aprono la connessione

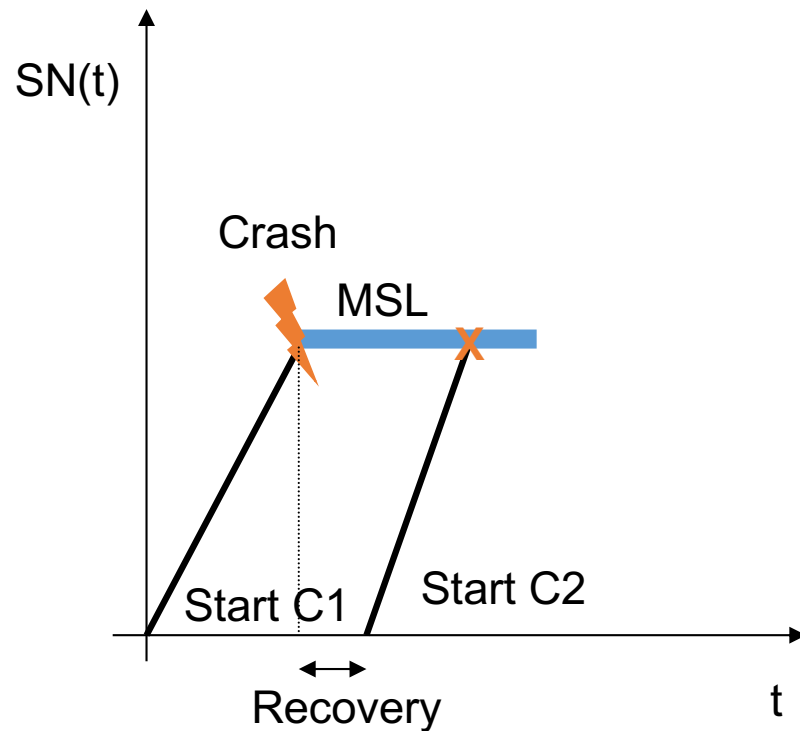


Se qualcosa va male

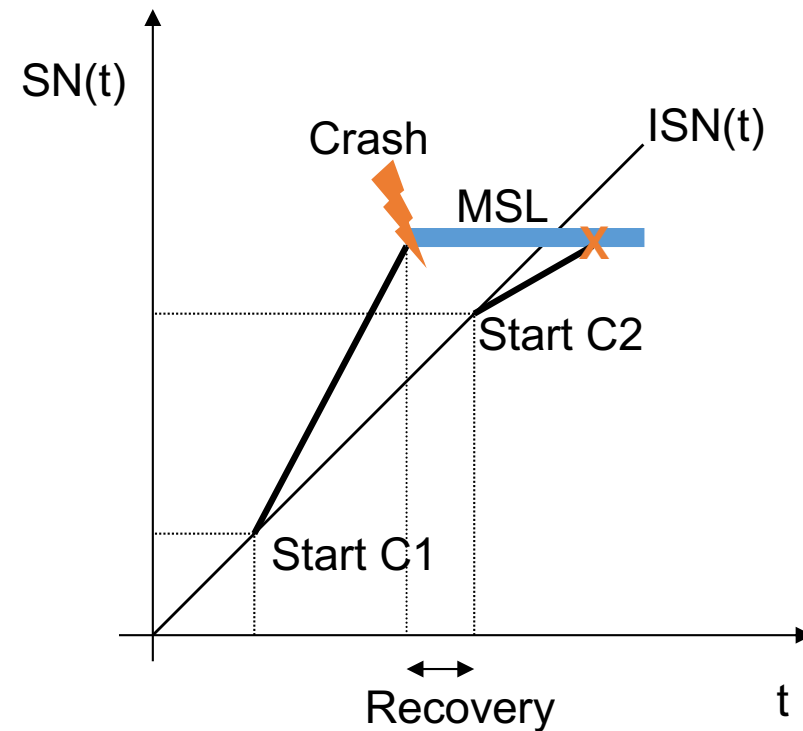
- Un host viene riavviato a causa di un problema?
 - Vengono ricreate nuove “*incarnazioni*” di vecchie connessioni
 - Vengono scelti nuovi ISN
 - Una nuova incarnazione può finire per usare numeri già utilizzati dall’ incarnazione precedente
 - Segmenti ritardati della prima incarnazione possono essere ricevuti dalla nuova incarnazione con numero di sequenza apparentemente corretto

Esempio

- ISN sempre uguale
 - $ISN = 0$



- $ISN(t) = t$
 - ISN uguale al valore di un contatore





ISN nella RFC 793

- ISN è funzione del tempo utilizzando un sistema di conteggio
 - Contatore a 32 bit
 - Incrementato ogni 4 μ sec
 - Il contatore ripete la sequenza ogni $2^{32} \cdot 4 \mu\text{sec} = 4.77 \text{ h}$
- TCP quiet time
 - Dopo un qualunque riavvio un host attende almeno un MSL prima di riaprire le connessioni TCP

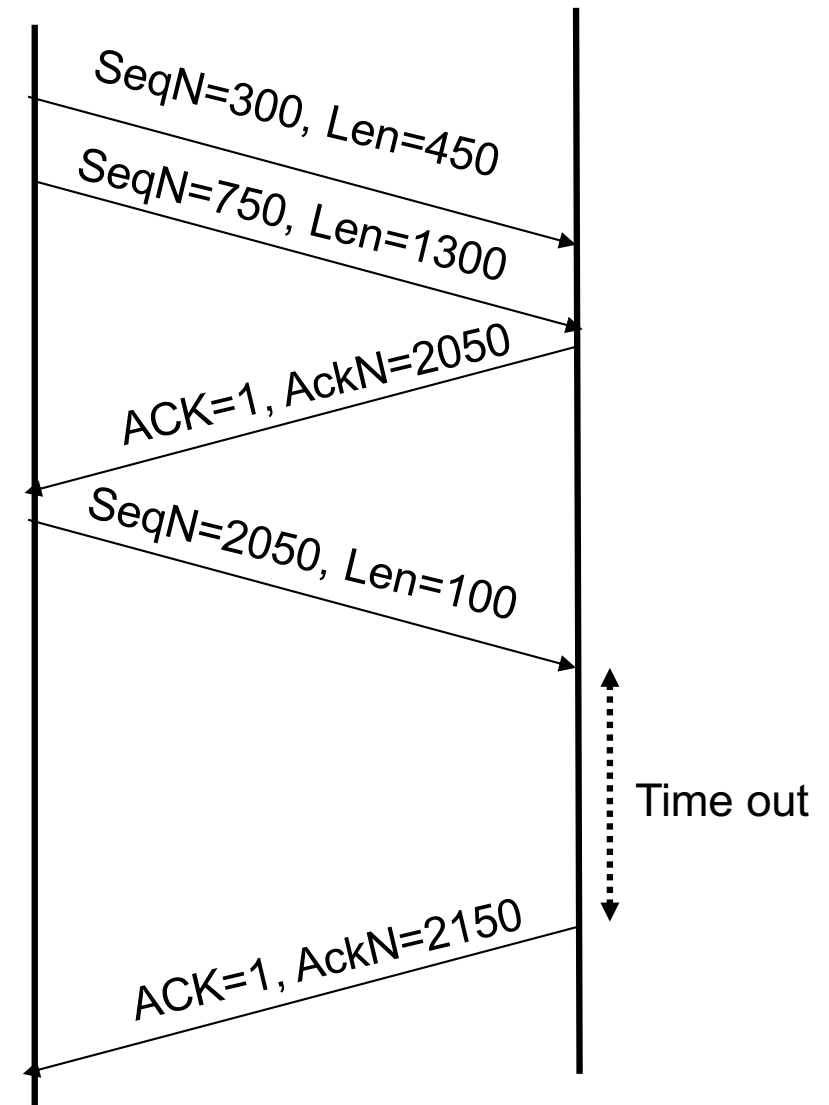


Messaggi di conferma (ACK)

- Gli Ack sono **cumulativi**
- In piggybacking
 - La conferma ha la forma di un normale messaggio TCP con il flag **ACK=1**
 - Può contenere dati oppure no
 - Se non contiene dati ne risulta un datagramma IP di 40 byte
 - 20 byte = min. header IP + 20 byte = min. header TCP
- Di default la conferma è esplicita
 - Il ricevitore trasmette un ACK per ogni segmento ricevuto
- ACK **ritardati**
 - Al momento della ricezione corretta di un segmento il ricevitore può inviare subito un ACK oppure ritardarlo
 - L'obiettivo è quello di minimizzare il numero di ACK
 - Se si ritarda troppo possono scattare i time-out

ACK ritardati

- Un ACK e' ritardato fino a che
 - è stato ricevuto un ulteriore pacchetto
 - Scatta un time out
 - Uguale a 200 ms nelle principali implementazioni
- Si riduce il traffico di ack
- Normalmente si produce un ACK ogni due segmenti



In ricezione

- Il ricevitore ha ricevuto fino a $\text{SeqN} = N$
 - Attende un segmento con $\text{SeqN} = N+1 \bmod M$
 - Riceve un segmento con $\text{SeqN} = X \neq N+1 \bmod M$
 - Se X è precedente ad N il segmento viene considerato un duplicato ritardato e viene scartato
 - Se X è successivo ad N il segmento è fuori sequenza (manca qualcosa)
 - Cosa può essere accaduto?
 - Uno o più segmenti sono andati persi
 - Un segmento trasmesso dopo un altro lo ha superato a causa dei diversi percorsi possibili e dei ritardi variabili in rete
 - Cosa fare?
 - Il ricevitore memorizza il segmento se X è entro W_R
 - Ritrasmette l'ultima conferma inviata (ACK duplicato)
 - Quando riceve $\text{seqN} = N$ a completamento della sequenza conferma $\text{ackN} = X+1$



In trasmissione

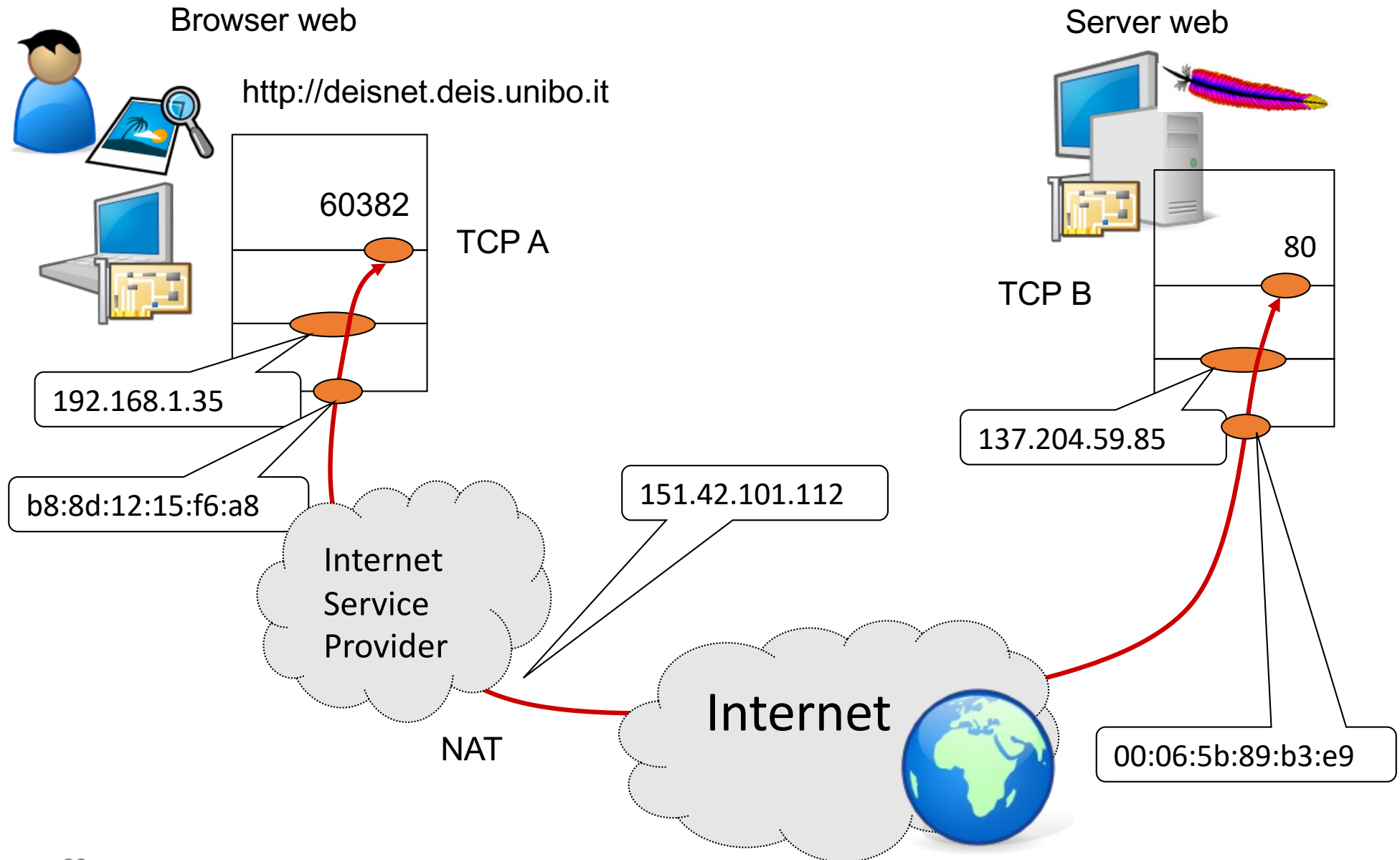
- Il trasmettitore invia il segmento $\text{seqN} = N$
 - Se riceve un ACK con $\text{ackN} = N+1$ toglie il segmento dalla memoria e fa scorrere la finestra di trasmissione
 - Se non riceve $\text{ackN} = N+1$ allo scadere di RTO ritrasmette il segmento
 - Ignora eventuali ACK duplicati



ACK duplicati

- Il TCP ricevente ritrasmette l'ACK per l'ultimo segmento ricevuto nella corretta sequenza generando un **ACK duplicato** (duplicate ACK)
 - Le implementazioni classiche di TCP ignorano gli ACK duplicati
 - Le implementazioni recenti prendono specifiche azioni se ricevono dei "duplicate ACK"

Topologia e indirizzi



Apertura: lato client – porte e flag

No	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=460
5038	69.185181	137.204.57.85	192.168.1.35	80	60382	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5039	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=13480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data (PDU)]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data (PDU)]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=13480
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data (PDU)]
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP	124	HTTP/1.1 200 OK (GIF89a)
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=13480
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112

Frame 5026: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)

Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)

Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)

Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 0

Source Port: 60382 (60382)

Destination Port: http (80)

[Stream index: 150]

[TCP Segment Len: 0]

Sequence number: 3874871719

Acknowledgment number: 0

Header Length: 44 bytes

... 0000 0000 0010 = Flags: 0x002 (SYN)

Window size value: 65535

[Calculated window size: 65535]

Porta sorgente = 60382

Porta destinazione = 80

Flag SYN impostato a 1

Apertura: lato client - ISN

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=460
5038	69.185181	137.204.57.85	192.168.1.35	80	60382	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5039	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=131480
5090	69.187182	192.168.1.35	137.204.57.85	60382	80	HTTP	703	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Len=0
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data len 1490, window 65535]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data len 1490, window 65535]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=131480
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of data len 1490, window 65535]
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP	124	HTTP/1.1 200 OK
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=131480
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112

Three ways handshake

```

Frame 5026: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)
Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)
Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 0
  Source Port: 60382 (60382)
  Destination Port: http (80)
  [Stream index: 150]
  [TCP Segment Len: 0]
  Sequence number: 3874871719
  Acknowledgment number: 0
  Header Length: 44 bytes
  ... 0000 0000 0010 = Flags: 0x00000010 (SYN)
  Window size value: 65535
  [Calculated window size: 65535]
0000 00 18 4d c0 7f be b8 8d 12 15 f6 a8 08 00 00 00
0010 00 40 41 be 40 00 40 06 74 0d c0 a8 01 23 80 00 00
0020 39 55 eb de 00 50 e6 f5 d9 a7 00 00 00 00 b0 00 00
0030 ff ff b6 c2 00 00 02 04 05 b4 01 03 03 05 01 00 00
0040 08 0a 11 ce 3c b3 00 00 00 00 04 02 00 00 00 00
  
```

TCP A invia il SYN e propone il suo ISN

SYN è il primo messaggio della connessione, non si sa nulla dell'altro e quindi ACK-N = 0



Invio richiesta HTTP

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.185181	137.204.57.85	192.168.1.35	80	60382	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Len=
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP		Win=129632
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP		Win=129568
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP		
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP		Win=130112
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP		
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP		
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP		
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=8

Terminata l'apertura della
connessione TCP A chiede l'invio della
pagina web

▶ Frame 5090: 763 bytes on wire (6104 bits), 763 bytes captured (6104 bits)
▶ Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: Netgear_c0:7f:be (00:18:4d:c0:7f:be)
▶ Internet Protocol Version 4, Src: 192.168.1.35 (192.168.1.35), Dst: 137.204.57.85 (137.204.57.85)
▶ Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871720, Ack: 847290583, Len: 697

▼ Hypertext Transfer Protocol

▶ GET / HTTP/1.1\r\n

Host: deisnet.deis.unibo.it\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n

Connection: keep-alive\r\n

▶ [truncated]Cookie: __utma=83985245.1349407932.1432415213.1432484513.3

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/600.5.17

Accept-Language: it-it\r\n

Content-Type: application/javascript

Il segmento TCP contiene 697 byte di
dati (payload) che sono le
informazioni di livello applicativo

0020 39 55 eb de 00 50 e6 f5 d9 a8 32 80 a0 d7 80 18 9U...P... ..2...
0030 10 2c 44 ff 00 00 01 01 08 0a 11 ce 3d 46 43 62 ..D..... ..=F...
0040 0b 1c 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 ..GET / HTTP/1...
0050 0d 0a 48 6f 73 74 3a 20 64 65 69 73 6e 65 74 2e ..Host: deisnet...
0060 64 65 69 73 2e 75 6e 69 62 6f 2e 69 74 0d 0a 41 deis.unibo.it...
0070 63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c ccept: text/html...
0080 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 ,application/xml...
0090 6d 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 ml+xml,application/...

Il segmento TCP contiene un messaggio HTTP
di tipo GET che richiede l'invio della pagina di
default del sito web



Invio richiesta HTTP

Numero di sequenza iniziale (ISN) concordato con il three ways handshake è
3874071720

Il segmento contiene 697 byte numerati sequenzialmente da 3874071720 a 3874072416 (in totale 697 numeri)
Il primo byte del prossimo segmento deve essere quello successivo: 3874072417

L'analizzatore di protocollo calcola il prossimo numero di sequenza e lo indica (non è incluso nell'intestazione TCP)

No.	Time	Source	Destination
-----	------	--------	-------------

5026	69.032250	192.168.1.35	137.204.57.8
5088	69.185181	137.204.57.85	192.168.1.35
5089	69.185319	192.168.1.35	137.204.57.8
5090	69.187162	192.168.1.35	137.204.57.8
5128	69.328866	137.204.57.85	192.168.1.35
5136	69.344413	137.204.57.85	192.168.1.35
5137	69.344939	137.204.57.85	192.168.1.35
5138	69.345002	192.168.1.35	137.204.57.85
5207	69.524222	137.204.57.85	192.168.1.35
5208	69.524362	137.204.57.85	192.168.1.35
5209	69.524458	192.168.1.35	137.204.57.85
5476	70.329117	192.168.1.35	137.204.57.85
5504	70.471930	137.204.57.85	192.168.1.35
5505	70.472011	192.168.1.35	137.204.57.85
5525	70.886308	137.204.57.85	192.168.1.35
5526	70.886409	192.168.1.35	137.204.57.85

```
Frame 5090: 763 bytes on wire (6104 bits), 763 bytes captured (6104 bits) on interface 0
Ethernet II, Src: Apple_15:f6:a8 (b8:8d:12:15:f6:a8), Dst: 08:00:27:00:00:00 (08:00:27:00:00:00)
Internet Protocol Version 4, Src: 192.168.1.35, Dst: 137.204.57.85
Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: 80 (80)
  Source Port: 60382 (60382)
  Destination Port: http (80)
  [Stream index: 150]
  [TCP Segment Len: 697]
  Sequence number: 3874871720
  [Next sequence number: 3874872417]
  Acknowledgment number: 847290583
  Header Length: 32 bytes
  ... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
  Window size value: 4140
```

```
0020 39 55 eb de 00 50 e6 f5 d9 a8 32 80 a0 d7 80 18 9U...P...
0030 10 2c 44 ff 00 00 01 01 08 0a 11 ce 3d 46 43 62 ..D.....
0040 0b 1c 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 .GET / HTTP/1.1
0050 0d 0a 48 6f 73 74 3a 20 64 65 69 73 6e 65 74 2e ..Host: deisnet.
0060 64 65 69 73 2e 75 6e 69 62 6f 2e 69 74 0d 0a 41 deis.uni bo.it..A
0070 63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c ccept: t ext/html
0080 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 , applica tion/xht
0090 6d 6c 20 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 ml+xml, a pplicati
```

```
535 Len=0 MSS=1460
k=3874871720 Win=57
7290583 Win=132480
290583 Ack=3874872417 Win=7188 Le
ed PDU]
PDU]
17 Ack=847293463 Win=129632
PDU]
17 Ack=847294961 Win=129568
HTTP/1.1
17 Ack=847295896 Win=130112
200 OK (GIF89a)
p [ACK] Seq=3874873117 Ack=8
```



Il primo ACK

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382→http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.185181	137.204.57.85	192.168.1.35	80	60382	TCP	74	http→60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382→http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http→60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]

Acknowledge number porta il numero di sequenza del primo byte che ci si aspetta di ricevere
(indica implicitamente che tutti quelli fino a quel numero sono stati ricevuti correttamente)

In risposta alla richiesta del client il server risponde per prima cosa con un ACK

Conferma la ricezione della richiesta

L'acknowledge ha lunghezza 0 poiché non contiene dati dell'applicazione ma solamente l'intestazione TCP

Il flag ACK è a 1

Il flag ACK è a 1



La pagina web

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.185181	137.204.57.85	192.168.1.35	80	60382	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.185319	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.187162	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.328866	137.204.57.85	192.168.1.35	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.344413	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5137	69.344939	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847293463 Win=129632
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP	124	HTTP/1.1 200 OK (text/html)
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847294961 Win=129568
5476	70.3	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.4	192.168.1.35	137.204.57.85	60382	80	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.4	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.8	192.168.1.35	137.204.57.85	60382	80	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.8	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=8
5527	70.8	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
.....								
2:15:f6:a8)								
192.168.1.35)								
7290583, Ack: 3874872417, Len: 1440								
Source Port: http (80)								
Destination Port: 60382 (60382)								
[Stream index: 150]								
[TCP Segment Len: 1440]								
Sequence number: 847290583								
[Next sequence number: 84729023]								
Acknowledgment number: 3874872417								
Header Length: 32 bytes								
.....								
b8 8d 12 15 f6 a8 00 18 4d c0 7f be 08 00 45 00								
0010 05 d4 ab 98 40 00 32 06 12 9f 89 cc 39 55 c0 a8								
0020 01 23 00 50 eb de 32 80 a0 d7 e6 f5 dc 61 80 10								
0030 07 05 b6 a3 00 00 01 01 08 0a 43 62 0b 44 11 ce								
0040 3d 46 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f								
0050 4b 0d 0a 44 61 74 65 3a 20 53 75 6e 2c 20 32 34								
0060 20 4d 61 79 20 32 30 31 35 20 31 36 3a 32 33 3a								
0070 32 33 20 47 4d 54 0d 0a 53 65 72 76 65 72 3a 20								

L'ultima trama MAC contiene quanto rimasto per completare la pagina web, complessivamente composta da $4378 = 1440 \cdot 3 + 58$ byte

Il client inizia a ricevere segmenti con dati dal server.
Sono 4 segmenti di lunghezza 1506, 1506, 1506 e 124 byte
La pagina web inviata dal server è stata suddivisa in 4 blocchi per riempire 3 datagrammi IP e 3 trame MAC al massimo possibile
Di questi 1506 byte 1440 sono relativi al protocollo HTTP ossia dati dell'applicazione server

Ritrasmissione: lato client

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
5026	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1460
5088	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
5089	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
5090	69.032250	192.168.1.35	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
5128	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
5136	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	1506	[TCP segment of a reassembled PDU]
5137	69.032250	192.168.1.35	137.204.57.85	60382	80	TCP	1506	[TCP segment of a reassembled PDU]
5138	69.345002	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847293463 Win=129632
5207	69.524222	137.204.57.85	192.168.1.35	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
5208	69.524362	137.204.57.85	192.168.1.35	80	60382	HTTP	124	HTTP/1.1 200 OK (text/html)
5209	69.524458	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847294961 Win=129568
5476	70.329117	192.168.1.35	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
5504	70.471930	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
5505	70.472011	192.168.1.35	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=130112
5525	70.886308	137.204.57.85	192.168.1.35	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
5526	70.886409	192.168.1.35	137.204.57.85	60382	80	TCP	66	[TCP Window Update] 60382->http [ACK] Seq=3874873117 Ack=8

Source Port: 60382	Destination Port: 60382 (60382)
[Stream index: 150]	[TCP Segment Len: 1440]
Sequence number: 847292023	[Next sequence number: 847293463]
Acknowledgment number: 3874872417	Header Length: 32 bytes
0000 0001 0000 = Flag: 0x010 (ACK)	

0000	b8 8d 12 15 f6 a8 00 18	4d c0 7f be 08 00 45 00 M.....E.
0010	05 d4 ab 99 40 00 32 06	12 9e 89 cc 39 55 c0 a8	...@.2.9U..
0020	01 23 00 50 eb de 32 80	a6 77 e6 f5 dc 61 80 10	..#.P..2. ..w...a..
0030	07 05 e8 66 00 00 01 01	08 0a 43 62 0b 44 11 ce	...f.... ..Cb.D..
0040	3d 46 6d 65 6e 74 6f 20	44 45 49 3c 2f 61 3e 3c	=Fmento DEI<
0050	2f 6c 69 3e 0a 3c 2f 75	6c 3e 0a 3c 2f 64 69 76	/li>.</u l>.</div
0060	3e 0a 0d 0a 0d 0a 3c 64	69 76 20 69 64 3d 22 63	>.....<d iv id="c
0070	6f 6e 74 61 69 6e 65 72	4d 65 6e 75 43 6f 6e 74	ontainer MenuCont



Lato server

No.	Time	Source	Destination	Src Port	Dst Port	Protocol	Length	Info
1883	54.742790	151.42.79.161	137.204.57.85	60382	80	TCP	78	60382->http [SYN] Seq=3874871719 Win=65535 Len=0 MSS=1452
1884	54.742816	137.204.57.85	151.42.79.161	80	60382	TCP	74	http->60382 [SYN, ACK] Seq=847290582 Ack=3874871720 Win=57
1892	54.877566	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874871720 Ack=847290583 Win=132480
1893	54.886652	151.42.79.161	137.204.57.85	60382	80	HTTP	763	GET / HTTP/1.1
1894	54.886673	137.204.57.85	151.42.79.161	80	60382	TCP	66	http->60382 [ACK] Seq=847290583 Ack=3874872417 Win=7188 Le
1897	54.899808	137.204.57.85	151.42.79.161	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
1898	54.899843	137.204.57.85	151.42.79.161	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
1903	55.080073	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847293463 Win=129632
1904	55.080095	137.204.57.85	151.42.79.161	80	60382	TCP	1506	[TCP segment of a reassembled PDU]
1905	55.080103	137.204.57.85	151.42.79.161	80	60382	HTTP	124	HTTP/1.1 200 OK (text/html)
1918	55.217302	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874872417 Ack=847294961 Win=129568
1980	56.028035	151.42.79.161	137.204.57.85	60382	80	HTTP	766	GET /images/english_flag.gif HTTP/1.1
1981	56.028436	137.204.57.85	151.42.79.161	80	60382	HTTP	1001	HTTP/1.1 200 OK (GIF89a)
2012	56.442888	137.204.57.85	151.42.79.161	80	60382	HTTP	1001	[TCP Retransmission] HTTP/1.1 200 OK (GIF89a)
2017	56.580028	151.42.79.161	137.204.57.85	60382	80	TCP	66	60382->http [ACK] Seq=3874873117 Ack=847295896 Win=131072
2423	71.032219	137.204.57.85	151.42.79.161	80	60382	TCP	66	http->60382 [FIN, ACK] Seq=847295896 Ack=3874873117 Win=85

▶ Frame 1883: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
▶ Ethernet II, Src: Cisco_8f:21:47 (00:16:47:8f:21:47), Dst: Intel_5e:3f:1c (00:0e:0c:5e:3f:1c)
▶ Internet Protocol Version 4, Src: 151.42.79.161 (151.42.79.161), Dst: 137.204.57.85 (137.204.57.85)
▼ Transmission Control Protocol, Src Port: 60382 (60382), Dst Port: http (80), Seq: 3874871719, Len: 0

Source Port: 60382 (60382)
Destination Port: http (80)
[Stream index: 44]
[TCP Segment Len: 0]
Sequence number: 3874871719
Acknowledgment number: 0
Header Length: 44 bytes

▶ 0000 0000 0010 = Flags: 0x002 (SYN)

Window size value: 65535

```
0000 00 0e 0c 5e 3f 1c 00 16 47 8f 21 47 08 00 45 00  ...^?... G.!G..E.
0010 00 40 41 be 40 00 32 06 5d 0d 97 2a 4f a1 89 cc  .@A.@.2. ]..*0...
0020 39 55 eb de 00 50 e6 f5 d9 a7 00 00 00 00 b0 02  9U...P.. ....
0030 ff ff 91 ca 00 00 02 04 05 ac 01 03 03 05 01 01  ....
0040 08 0a 11 ce 3c b3 00 00 00 00 04 02 00 00  ....<.. .....
```