FUNDAMENTOS DE REDES
CAPA DE TRANSPORTE

Daniel Barragán C.
daniel.barragan@correounivalle.edu.co
Lunes y Miércoles 3:00 pm a 5:00 pm – Edificio 331 Oficina 2114

Universidad del Valle

Capa de transporte

Agenda

- Introducción
- Servicios de la Capa de Transporte
- Multiplexación de Aplicaciones
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)

Introducción

Proporciona una comunicación lógica entre aplicaciones. Desde el punto de vista de las aplicaciones es como si estuvieran conectadas físicamente aunque se encuentren en distintos lugares del planeta

Servicios

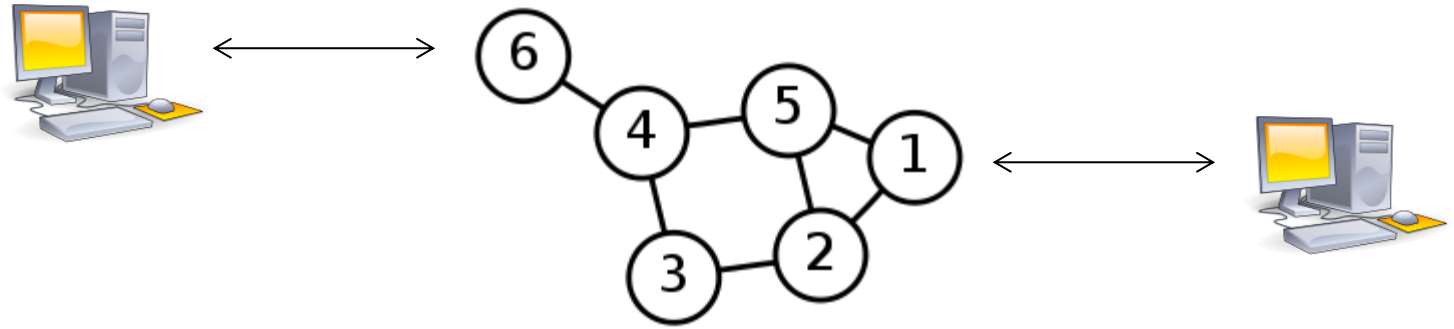
La capa de transporte provee los siguientes servicios:

- Multiplexación de aplicaciones
- Detección de error
- Servicio orientados y no orientados a conexión
- Entrega confiable: confirmaciones de envió y recepción
- Control de flujo: monitoreo de buffers en emisor y receptor
- Control de congestión

Multiplexación de Aplicaciones

Capa de red

IP se encarga de la comunicación de equipo a equipo. IP no entrega información entre aplicaciones.



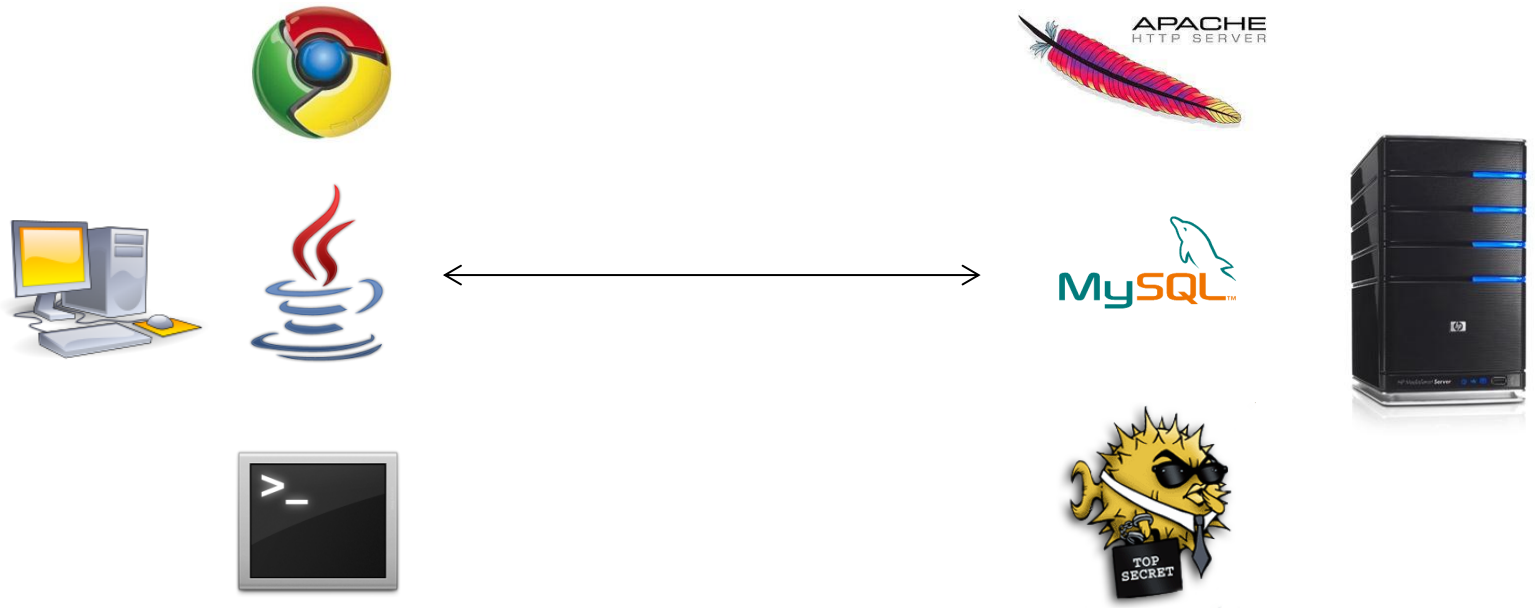
Multiplexación de Aplicaciones

Capa de transporte

La capa de transporte extiende la comunicación proporcionada por la capa de red para proporcionar conexión entre aplicaciones

Multiplexación de Aplicaciones

La capa de transporte recibe los segmentos de información de la capa de red y se encarga de entregarlos a la aplicación apropiada



Multiplexación de Aplicaciones

A cada aplicación se le asocia un numero de puerto, de esta manera se tiene una identificación única de las aplicaciones

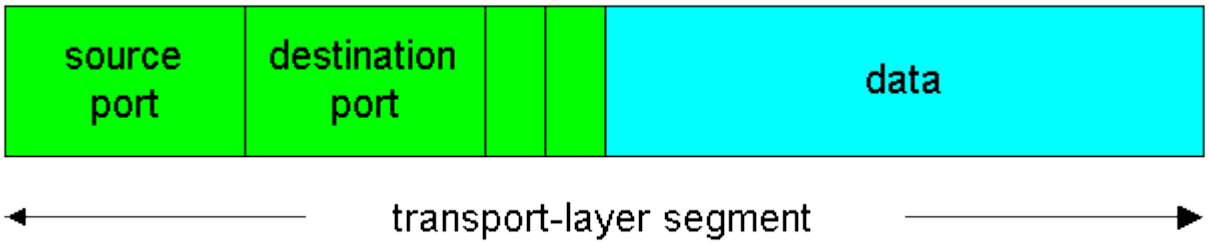
Multiplexación de Aplicaciones

Los puertos del 0 al 1023 se consideran puertos reservados. Los puertos del 1024 al 65535 son puertos de propósito general

Multiplexación de Aplicaciones

Segmento de información

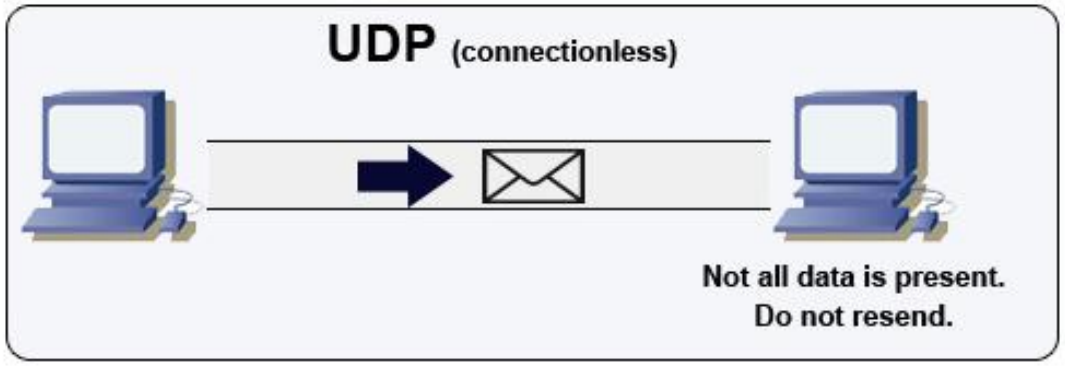
Los segmentos de información de la capa de transporte incluyen información del puerto origen y puerto destino de la comunicación



UDP

User Datagram Protocol

Protocolo No Orientado a Conexión.



UDP

Generalidades

- No hay establecimiento de la conexión. Los datos se envían directamente sin negociar la comunicación
- No se guarda estado de la conexión. No se guarda información para el control de flujo o confirmación
- Tiene una longitud de 64 bits (8 bytes) que se adicionan al mensaje de la capa de aplicación
- La tasa de envío de información es variable y depende del estado de la red (congestión)
- Provee detección de error (no recuperación)

UDP

Aplicaciones

Aplicación	Protocolo en la capa de aplicación	Protocolo en la capa de transporte
Remote file server	NFS	UDP
Streaming multimedia	Propietary	UDP
Internet telephony	Propietary	UDP
Network management	SNMP	UDP
Routing Protocol	RIP	UDP
Name Translation	DNS	UDP

UDP

DNS (Mensaje de aplicación vía UDP)

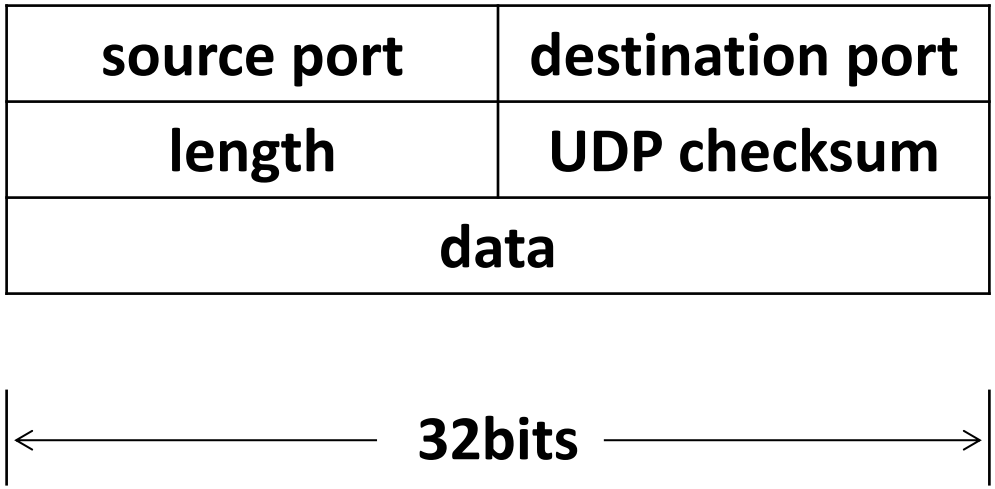
Cuando se requiere resolver un nombre de dominio ocurre lo siguiente:

1. Se crea un **mensaje** de solicitud DNS
2. UDP adiciona una cabecera al mensaje y forma un **segmento UDP**
3. Se encapsula el segmento UDP en un **datagrama IP**
4. Se envía el datagrama IP a un servidor de nombres (DNS)
5. De no haber respuesta se intenta con otros servidores (DNS)

UDP

Estructura

El segmento de información UDP posee una cabecera de 4 campos, cada una de ellas compuesta por 16bits



UDP

Estructura

source port y destination port: son usadas para multiplexar información hacia las aplicaciones de la capa superior

length: longitud del segmento UDP en bytes

UDP cheksum: empleado para detección de error

UDP

Detección de Error

El checksum de UDP proporciona detección de error. El checksum se calcula por medio de la suma de los campos de la cabecera del segmento UDP (también algunos campos de la cabecera IP) y el complemento a uno del resultado de la suma

UDP

Detección de Error

El checksum de UDP proporciona detección de error. El checksum se calcula por medio de la suma de los campos de la cabecera del segmento UDP (también algunos campos de la cabecera IP) y el complemento a uno del resultado de la suma

Pregunta: ¿Si existen protocolos en la capa de enlace como Ethernet que realizan detección y corrección de error por que UDP proporciona un mecanismo para detección de error?

UDP

Detección de Error (Ejemplo)

En el emisor

Contenido Cabecera UDP (1)	0110011001100110
Contenido Cabecera UDP (2)	0101010101010101
Suma Cabecera en el emisor (1,2)	1011101110111011
Contenido Cabecera UDP (3)	0000111100001111
Suma Cabecera en el emisor (1,2,3)	1100101011001010
Complemento a Uno	0011010100110101
Se transmite	0011010100110101

UDP

Detección de Error (Ejemplo)

En el receptor

Se recibe	0011010100110101
Suma Cabecera en el receptor (1,2,3)	1100101011001010
Suma de Comprobación	1111111111111111

UDP

Problema:

¿Cual es el valor del **checksum** que se transmite?

Contenido Cabecera UDP (1)	0110011001100111
Contenido Cabecera UDP (2)	0101010101010101
Suma Cabecera en el emisor (1,2)	
Contenido Cabecera UDP (3)	0000101100001110
Suma Cabecera en el emisor (1,2,3)	
Complemento a Uno	
Se transmite	

UDP

Solución:

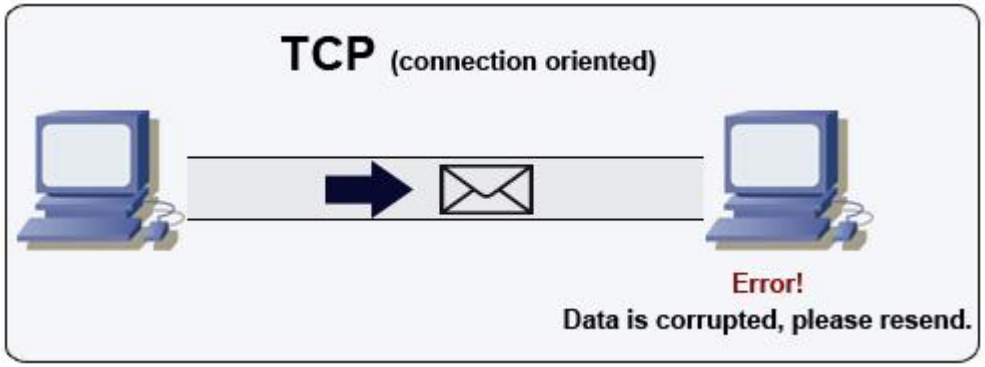
¿Cual es el valor del **checksum** que se transmite?

Contenido Cabecera UDP (1)	0110011001100111
Contenido Cabecera UDP (2)	0101010101010101
Suma Cabecera en el emisor (1,2)	1011101110111100
Contenido Cabecera UDP (3)	0000101100001110
Suma Cabecera en el emisor (1,2,3)	1100011011001010
Complemento a Uno	0011100100110101
Se transmite	0011100100110101

TCP

Transmission Control Protocol

Protocolo Orientado a Conexión



TCP

Generalidades

- Hay establecimiento de la conexión. Los datos se envían una vez se negocia la comunicación
- Se guarda estado de la conexión. Se guarda información para el control de flujo o confirmación
- Tiene una longitud de 160 bits (20 bytes) que se adicionan al mensaje de la capa de aplicación

TCP

Generalidades

- Es un protocolo punto a punto (no soporta multicasting)
- Es un protocolo **full-duplex**
- Provee detección de error (no recuperación)
- Puede tener en transito múltiples segmentos de datos sin haber recibido confirmación (Pipelining)

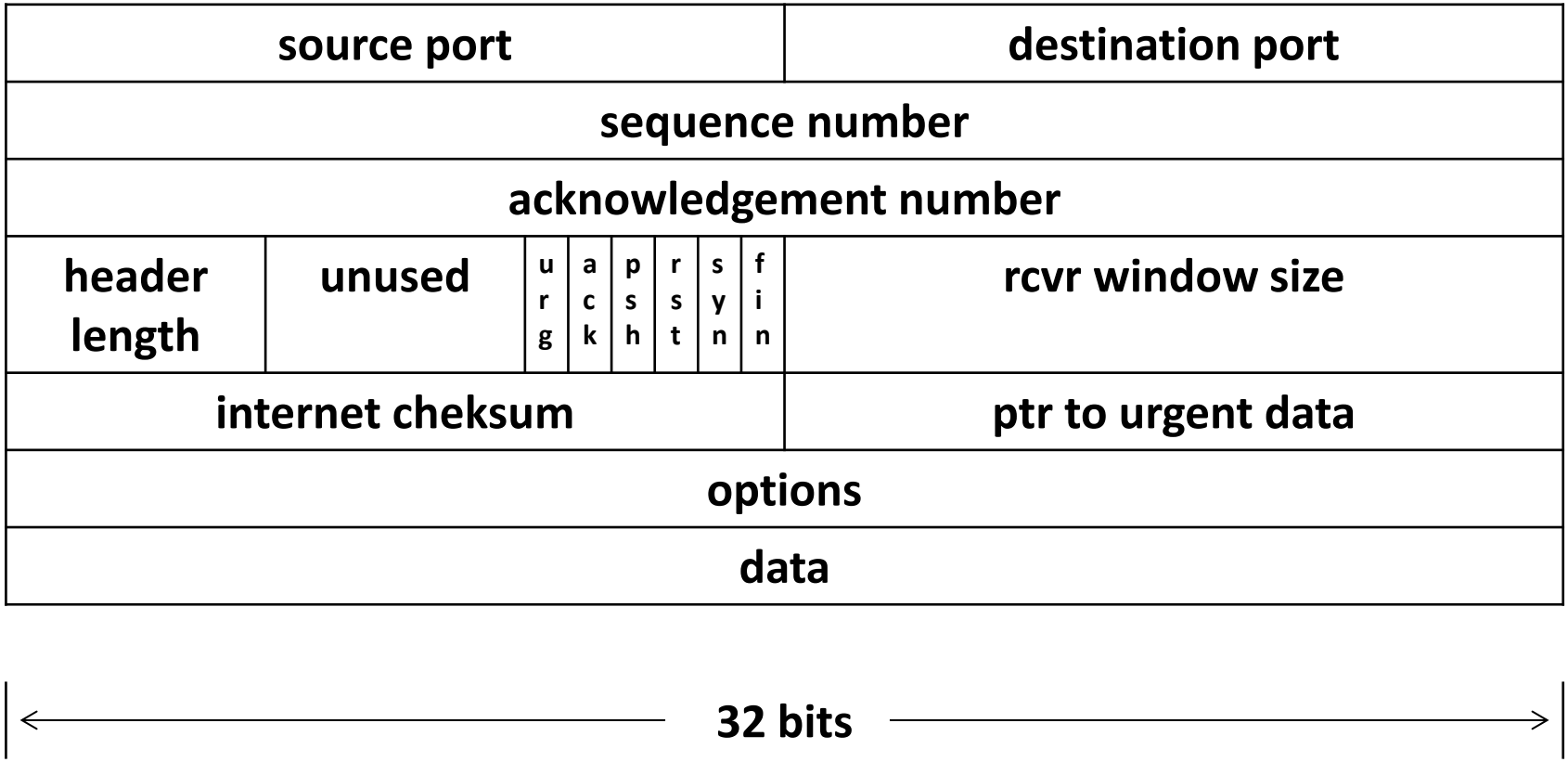
TCP

Aplicaciones

Aplicación	Protocolo en la capa de aplicación	Protocolo en la capa de transporte
Electronic mail	SMTP	TCP
Remote Terminal Access	Telnet	TCP
Web	HTTP	TCP
File Transfer	FTP	TCP

TCP

Estructura



TCP

Estructura

source port y destination port: son usadas para multiplexar información hacia las aplicaciones de la capa superior

sequence number y acknowledgement number: empleados para implementar transferencia de datos confiable

window size: es empleado para indicar el numero de bytes que el receptor esta dispuesto a aceptar

header length: longitud de la cabecera en unidades de 32 bits

options: es usado cuando se negocia el MSS y en redes de alta velocidad para incrementar el tamaño de la ventana

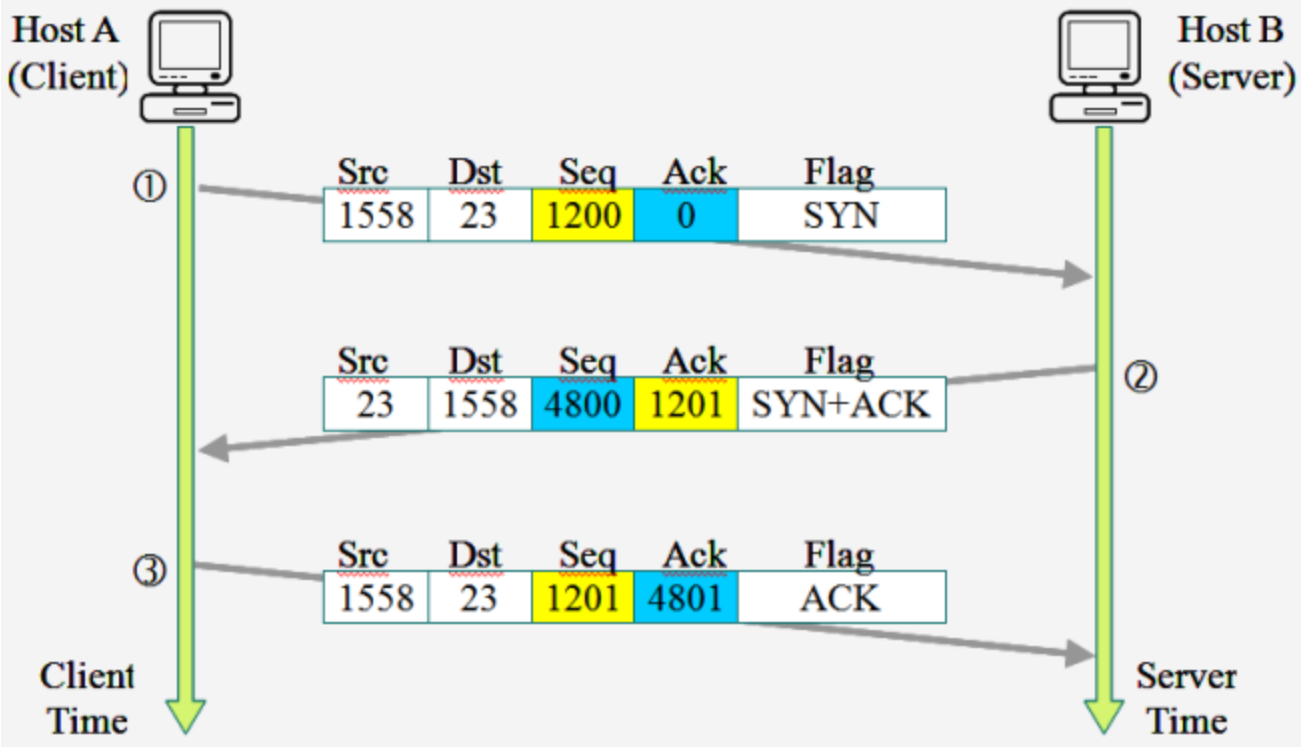
TCP

Three-way handshake

La comunicación en TCP involucra un emisor (proceso cliente) y receptor (proceso servidor). En el establecimiento de la conexión se dan tres pasos.

TCP

Three-way handshake



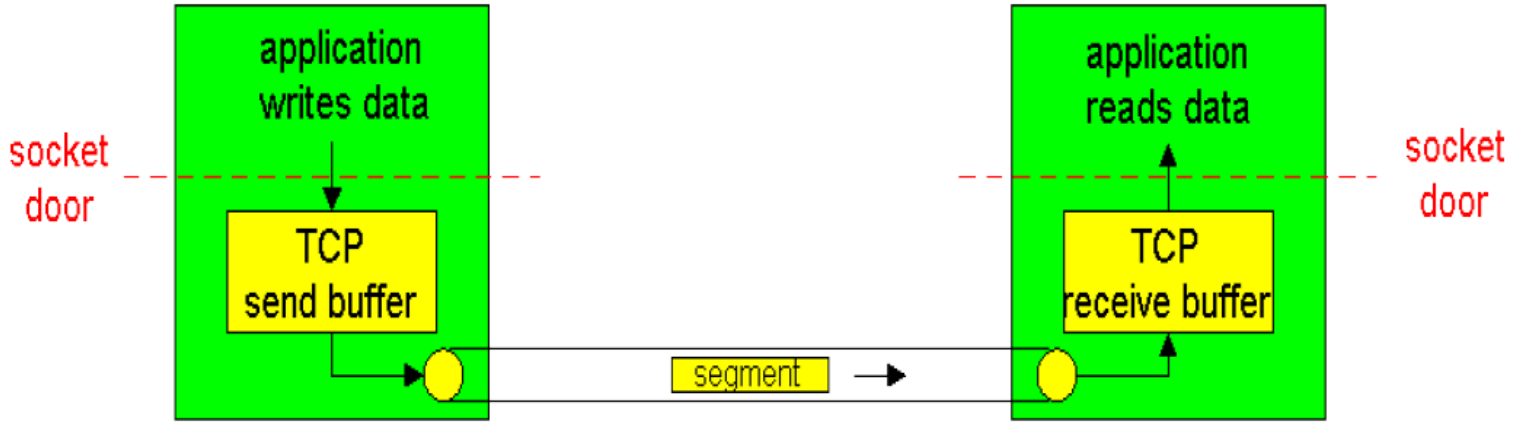
TCP

Three-way handshake

1. El emisor envía un segmento especial con el bit **SYN = 1** y un numero de secuencia inicial ***seq = client_isn***
2. El receptor envía un segmento especial con el bit **ACK = 1 y SYN = 1**, envía una confirmación ***ack = client_isn + 1*** y envía su propio número de secuencia inicial ***seq = server_isn***
3. El emisor envía un segmento especial con el bit **ACK = 1**, envía un ***seq = client_isn +1*** y **ack = *server_isn + 1***

TCP

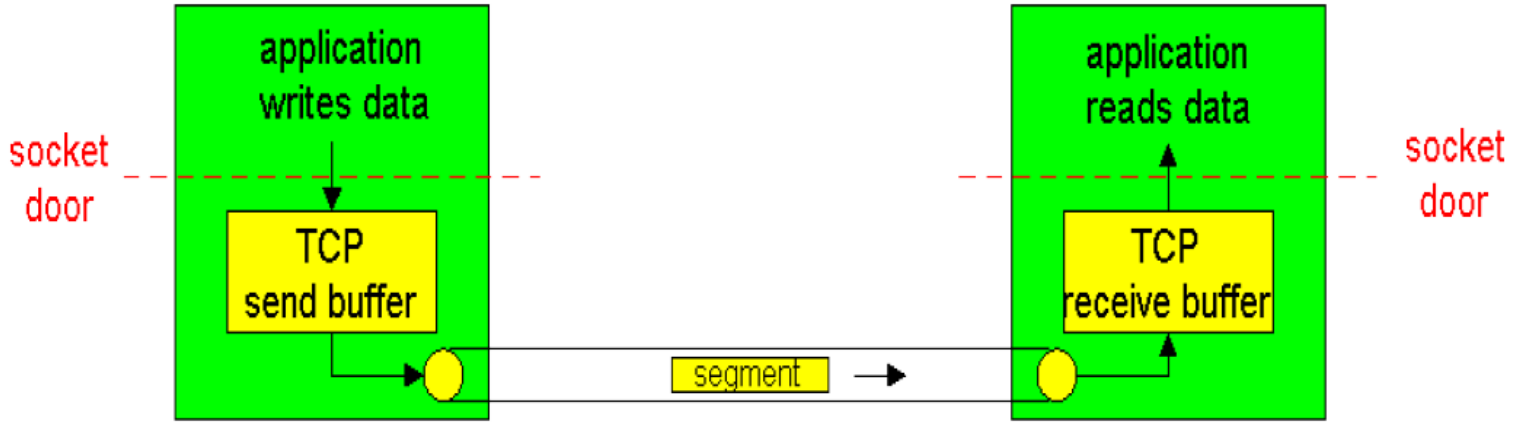
Transmisión de Información



1. El cliente envía un flujo de datos a través de un socket

TCP

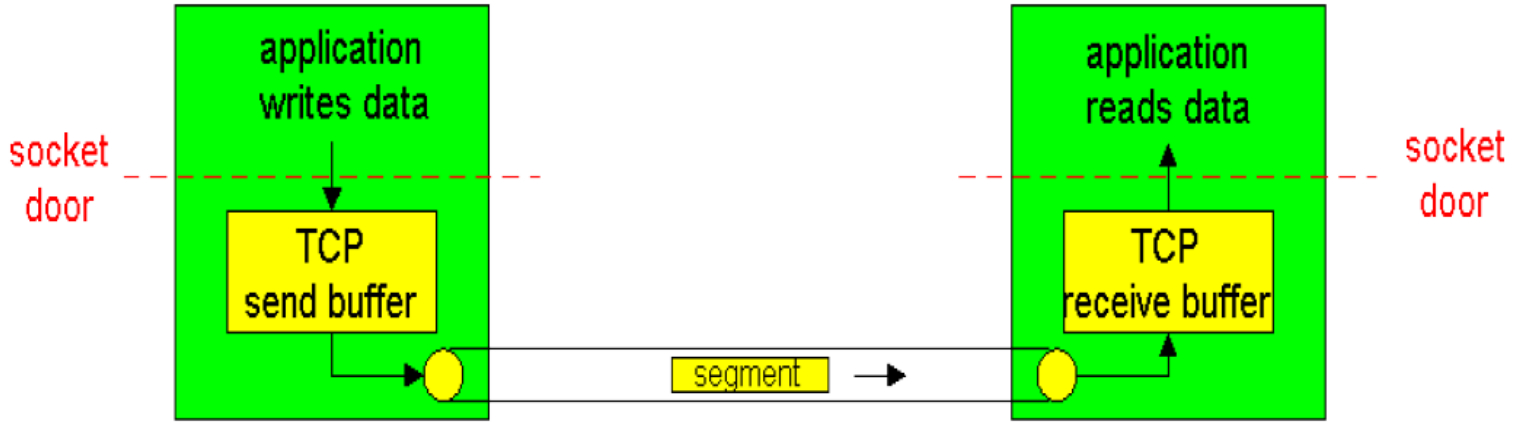
Transmisión de Información



2. TCP direcciona estos datos a un buffer.

TCP

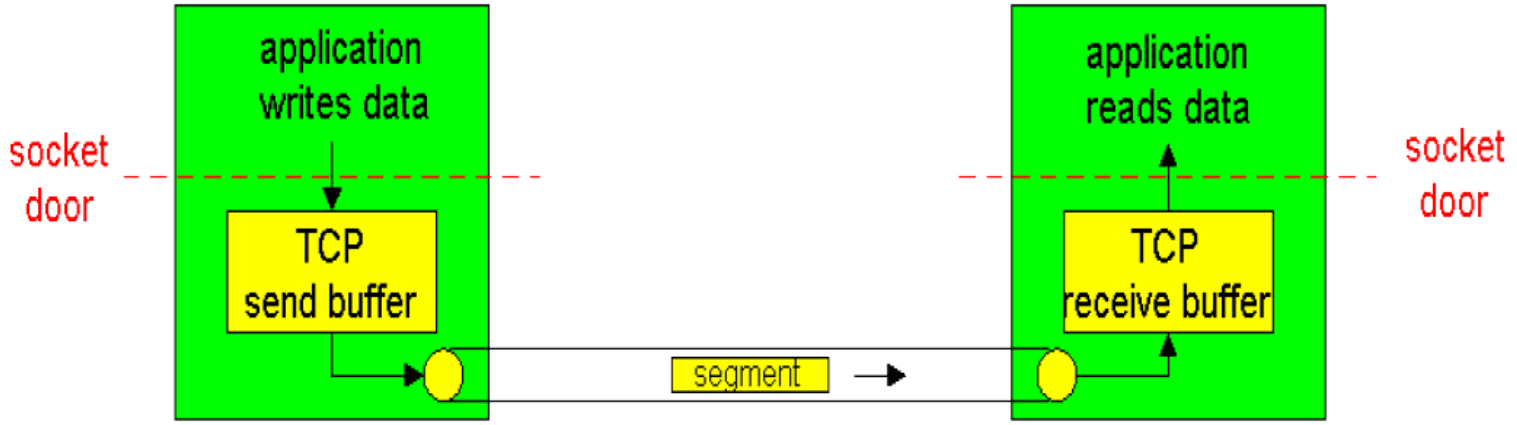
Transmisión de Información



3. TCP toma una cantidad MSS (Maximum Segment Size) de bytes del buffer. MSS es comúnmente de 1500, 536, 512 bytes

TCP

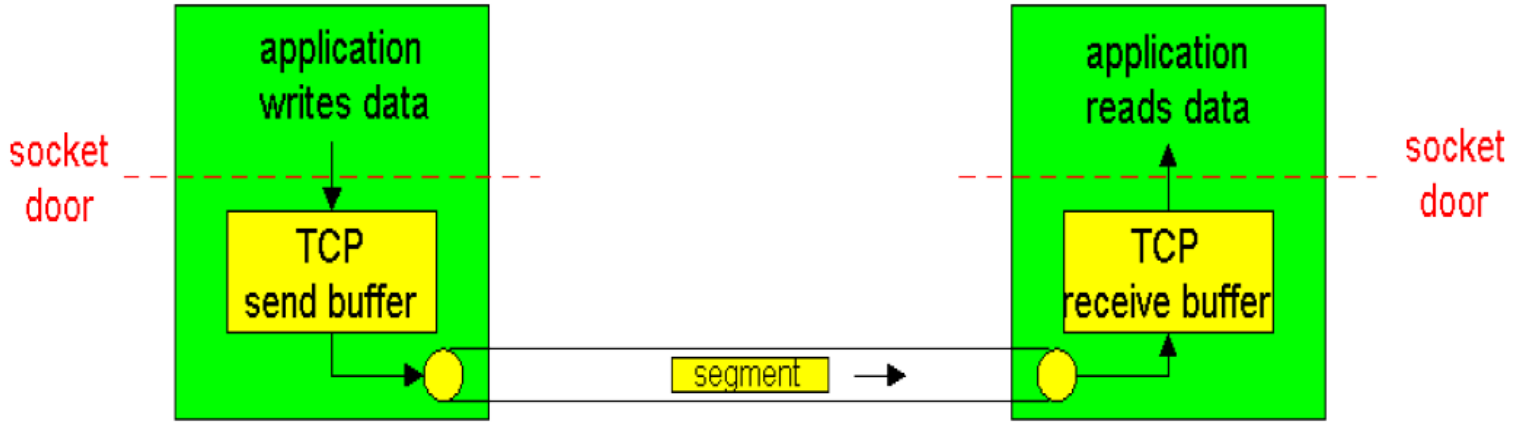
Transmisión de Información



4. TCP adiciona una cabecera TCP y entrega el segmento de información TCP (datos + cabecera) a la capa de red

TCP

Transmisión de Información



Nota: Si el tamaño de la información es superior al tamaño del MSS se realizan varios envíos

TCP

Transmisión de Información

¿Cual seria el tamaño del segmento TCP (datos+cabecera), para una cabecera TCP sin opciones?

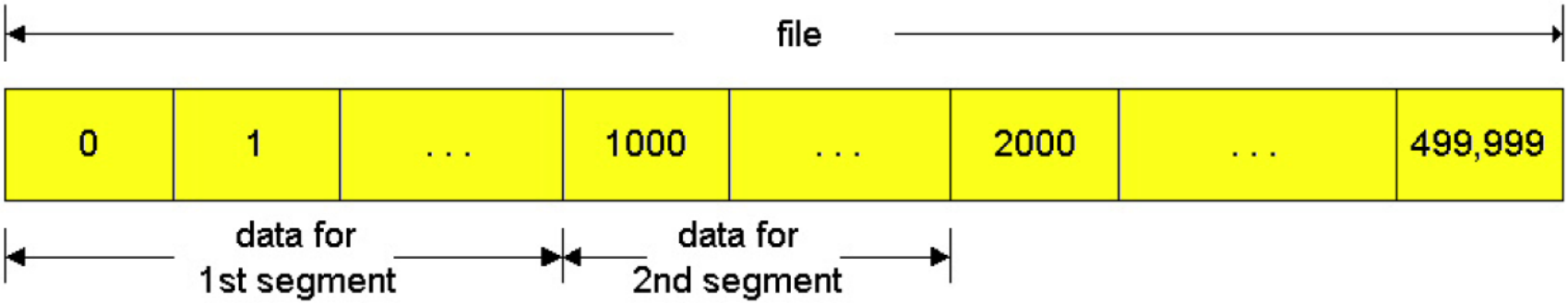
TCP

Números de Secuencia y Confirmación

La información esta conformada por un flujo de bytes

Cada byte esta asociado a un numero que indica su posición en el flujo de información

El número de secuencia corresponde a la posición del byte que inicia cada segmento



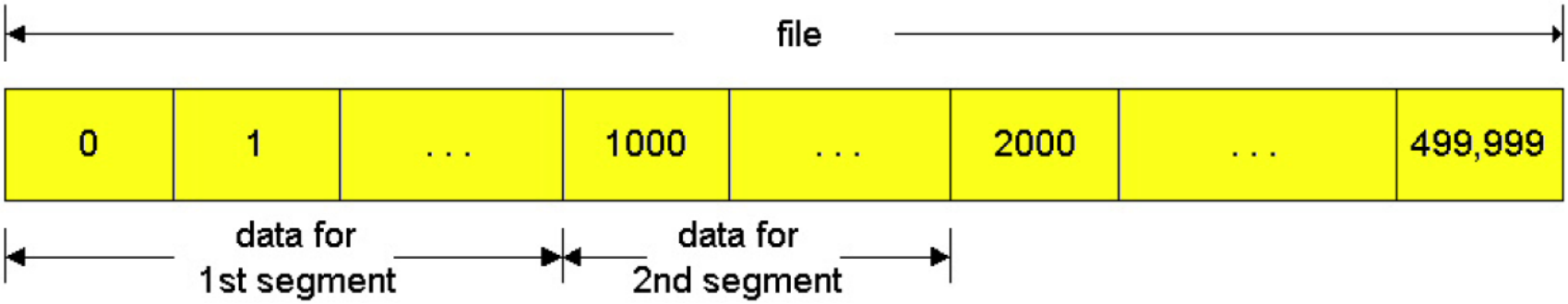
TCP

Números de Secuencia y Confirmación

A partir de la gráfica se observa que para el segmento 1, el número de secuencia que emplea el emisor es 0

Para el segmento 2 el número de secuencia es 1000

Para el segmento 3 el número de secuencia es 2000

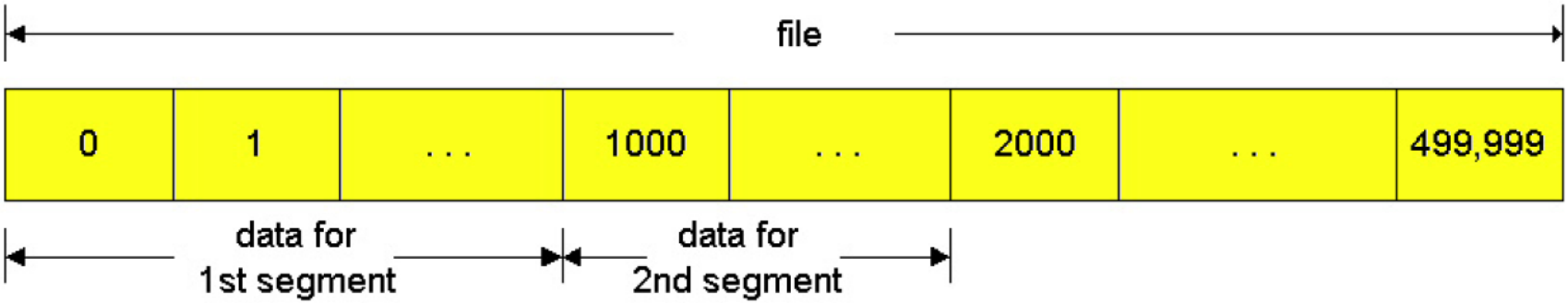


TCP

Números de Secuencia y Confirmación

El número de confirmación corresponde al siguiente byte del flujo de datos que se espera recibir

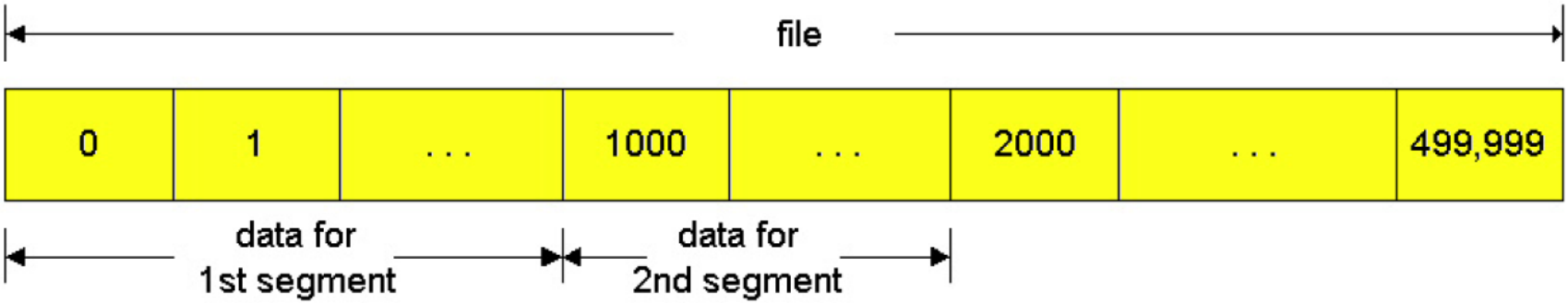
Las confirmaciones TCP son acumulativas (si se pierde el **ack** del segmento1 pero se recibe el **ack** del segmento 2, se asume que el receptor recibió el segmento 1 y 2)



TCP

Números de Secuencia y Confirmación

A partir de la gráfica para confirmar que se ha recibido el segmento 1 con los bytes desde el 0 hasta el 999, el receptor envía el número de confirmación 1000



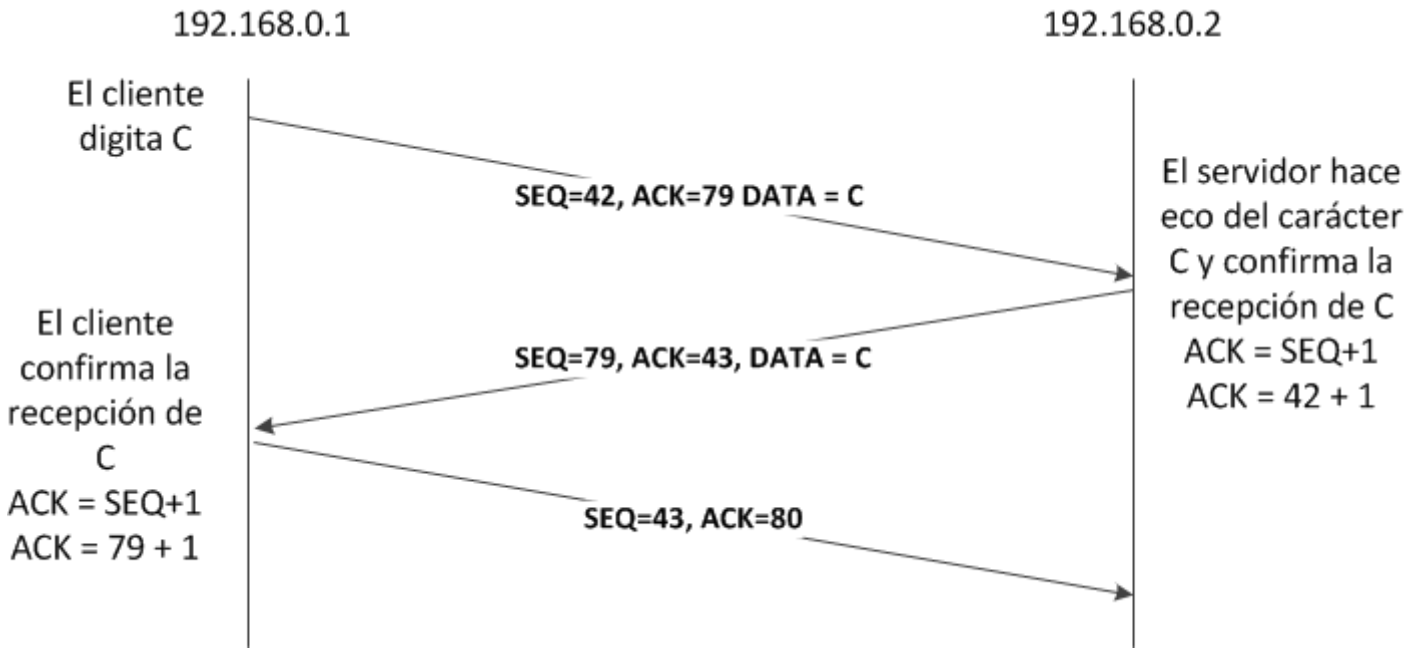
TCP

Problema:

Por medio de un grafico represente el envío de segmentos TCP en una conexión de Telnet. Tenga en cuenta que un emisor envía el carácter **C** y el receptor realiza un eco del carácter **C**. Emplee como número de secuencia el 42 y como número de confirmación 79

Nota: Telnet envía un solo carácter por cada transmisión

TCP


192.168.0.1
192.168.0.2
El cliente digita C
SEQ=42, ACK=79 DATA = C
El servidor hace eco del carácter C y confirma la recepción de C
ACK = SEQ+1
ACK = 42 + 1
El cliente confirma la recepción de C
ACK = SEQ+1
ACK = 79 + 1
SEQ=79, ACK=43, DATA = C
SEQ=43, ACK=80

TCP

Transferencia Confiable

Para analizar la transferencia confiable en TCP se deben tener en cuenta los siguientes aspectos:

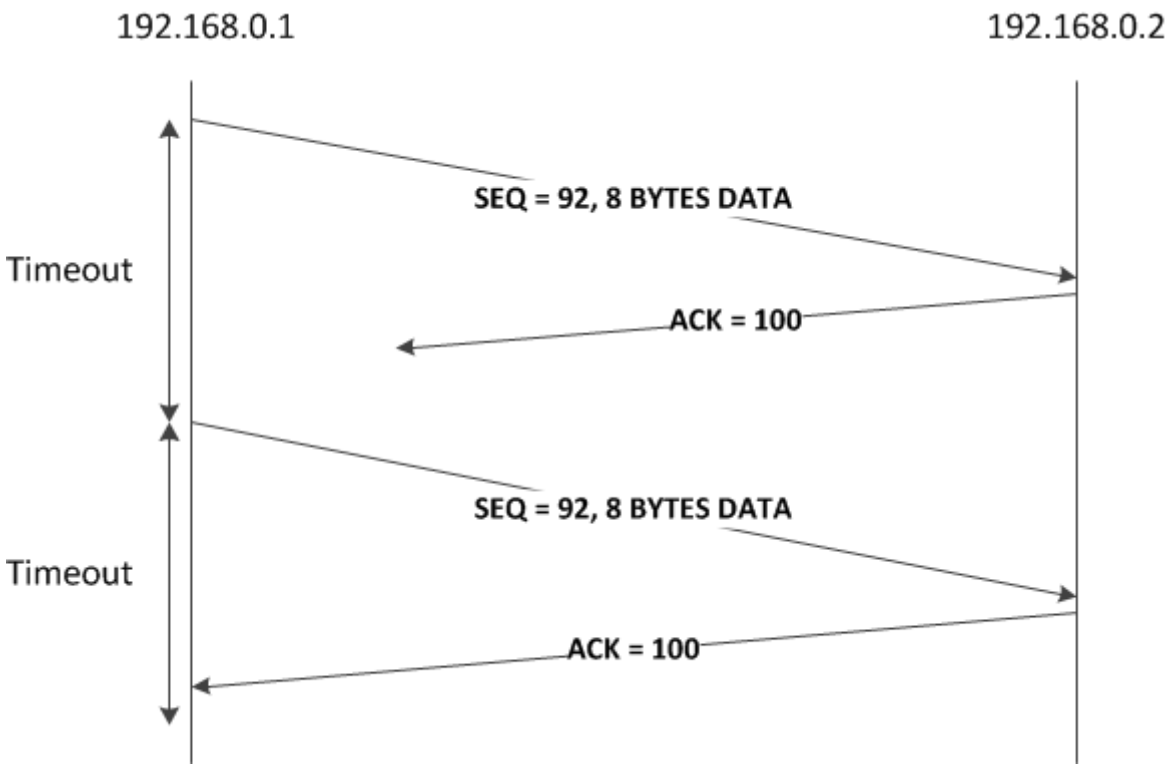
- TCP divide la información de la aplicación en segmentos de acuerdo al valor del MSS
- TCP inicia un temporizador de cuenta regresiva (**timeout**) con cada envío de un segmento de información
- TCP recibe mensajes de confirmación por cada segmento que recibe el receptor.

TCP

Transferencia Confiable

TCP inicia un temporizador de cuenta regresiva (**timeout**) con cada envío de un segmento de información. Cuando el temporizador expira y no se ha recibido confirmación del receptor, se realiza un reenvió del segmento de información

TCP

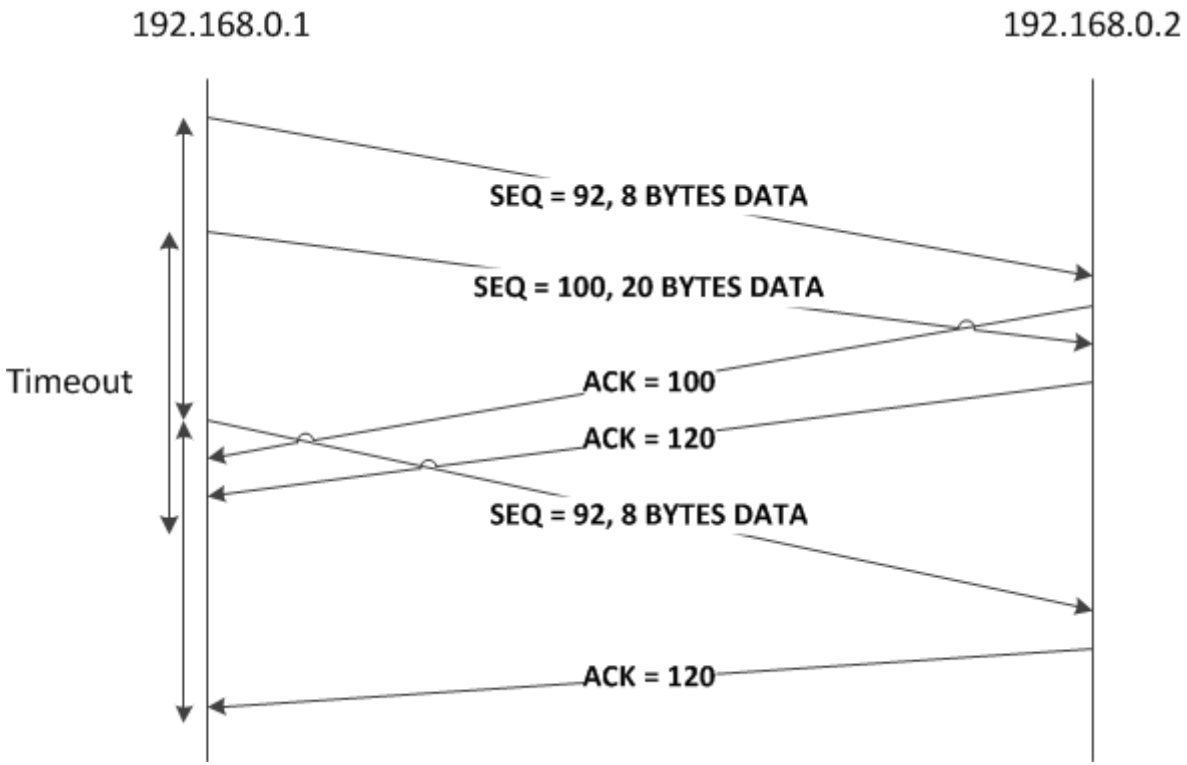

192.168.0.1
192.168.0.2
SEQ = 92, 8 BYTES DATA
Timeout
ACK = 100
SEQ = 92, 8 BYTES DATA
Timeout
ACK = 100

TCP

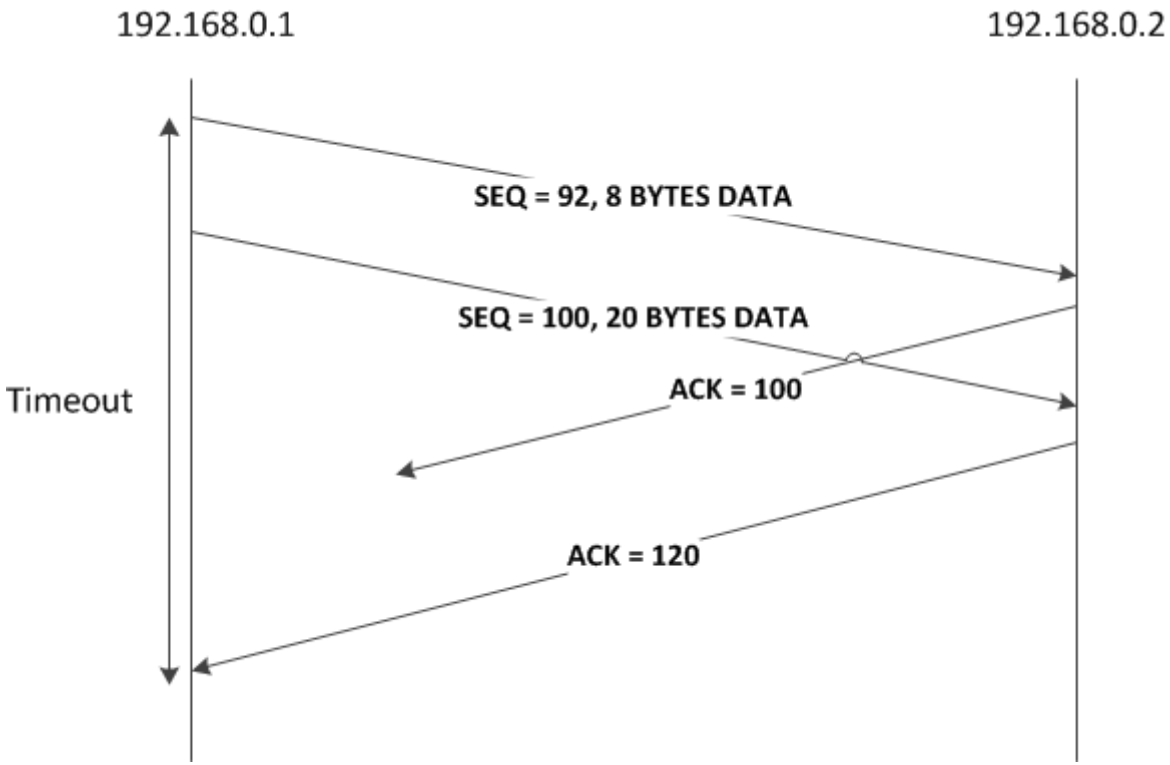
Transferencia Confiable

Los números de confirmación en TCP (**acknowledgement number**) son acumulativos. Si se pierde una confirmación de recepción pero se recibe la que corresponde a un segmento posterior, es debido a que el receptor recibió ambas correctamente

TCP


192.168.0.1
192.168.0.2
SEQ = 92, 8 BYTES DATA
SEQ = 100, 20 BYTES DATA
Timeout
ACK = 100
ACK = 120
SEQ = 92, 8 BYTES DATA
ACK = 120

TCP

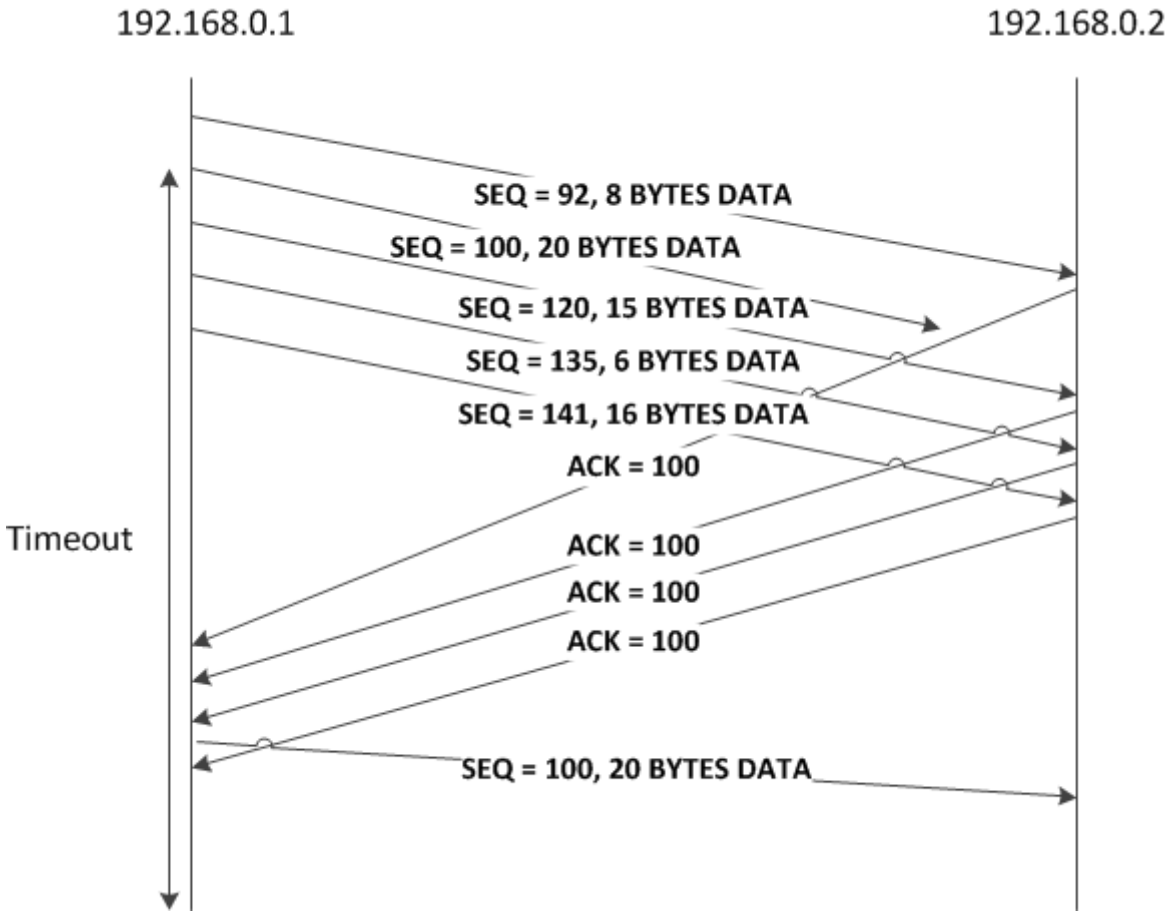

192.168.0.1
192.168.0.2
SEQ = 92, 8 BYTES DATA
SEQ = 100, 20 BYTES DATA
Timeout
ACK = 100
ACK = 120

TCP

Transferencia Confiable

El RFC 0793 correspondiente a TCP no especifica la acción a realizar cuando se reciben segmentos fuera de orden. Es decisión del desarrollador descartar los paquetes fuera de orden o almacenarlos temporalmente y reordenar al obtener los segmentos faltantes

TCP


192.168.0.1
192.168.0.2
SEQ = 92, 8 BYTES DATA
SEQ = 100, 20 BYTES DATA
SEQ = 120, 15 BYTES DATA
SEQ = 135, 6 BYTES DATA
SEQ = 141, 16 BYTES DATA
ACK = 100
Timeout
ACK = 100
ACK = 100
ACK = 100
SEQ = 100, 20 BYTES DATA

TCP

Recomendaciones ACK (RFC 2581, RFC 1122)

Event	TCP receiver action
Arrival of in-order segment with expected sequence number. All data up to up to expected sequence number already acknowledged. No gaps in the received data.	Delayed ACK. Wait up to 500 ms for arrival of another in-order segment. If next in-order segment does not arrives in this interval, send an ACK
Arrival of in-order segment with expected sequence number. One other in-order segment waiting for ACK transmission. No gaps in the received data.	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment with higher-than expected sequence number. Gap detected.	Immediately send duplicate ACK, indicating sequence number of next expected byte
Arrival of segment that partially or completely fills in gap in received data	Immediately send ACK, provided that segment starts at the lower end of gap.

TCP

Control de Flujo

El servicio de control de flujo de TCP permite evitar la saturación del buffer del receptor.

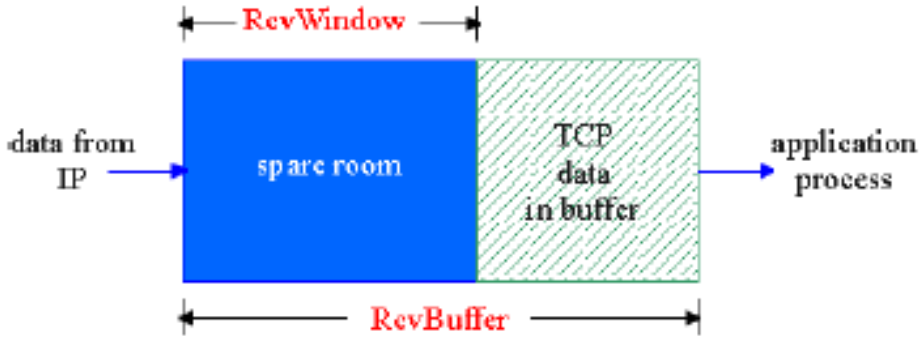
TCP

Control de Flujo

TCP provee control de flujo a través de una variable en el emisor llamada ventana del receptor

TCP

Control de Flujo



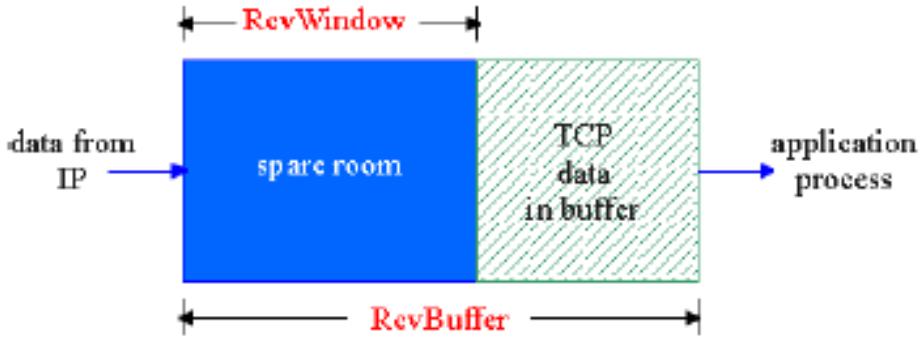
- En el receptor:

LastByteRcvd - LastByteRead <= RcvBuffer

***RcvWindow** = RcvBuffer – (LastByteRcvd – LastByteRead)*

TCP

Control de Flujo



- En el emisor:

LastByteSend – LastByAcked <= RcvWindow

TCP

Control de Flujo

El valor del temporizador (**timeout**) de un segmento de datos se debe calcular en relación al RTT (Round Trip Time) de acuerdo a las siguientes consideraciones:

- No debe ser mucho mas corto que el RTT
- No debe ser mucho mas grande que el RTT

TCP

Control de Flujo

El valor del temporizador (**timeout**) de un segmento de datos se debe calcular en relación al RTT (Round Trip Time) de acuerdo a las siguientes consideraciones:

- No debe ser mucho mas corto que el RTT
  - Genera retransmisiones innecesarias
- No debe ser mucho mas grande que el RTT
  - Generar retardos amplios en la retransmisión

TCP

Control de Flujo

EstimatedRTT = (1-x)EstimatedRTT + x SampleRTT

TCP

Control de Flujo

Timeout *= EstimatedRTT + 4*Deviation*

Deviation = (1-x) Deviation + x | SampleRTT – EstimatedRTT |

TCP

Control de Flujo

Un valor comúnmente usado para x es 0.1

TCP

Control de Congestión

TCP provee mecanismos para el control de la congestión en la red. La congestión en la red se produce cuando la cantidad de información a enviar es cercana ó excede la capacidad del canal de comunicación

Bibliografía

Computer Networking: A Top-Down Approach
Sexta Edición (2012)
James F. Kurose and Keith W. Ross

Using Snort and Ethereal to Master The 8 Layers Of An Insecure Network
Primera Edición (2006)
Michael Gregg, Stephen Watkins, George Mays, Chris Ries, Ronald M. Bandes, Brandon Franklin

Enlaces Adicionales

Herramientas

http://nmap.org/

http://www.hping.org/

http://drjohnstechtalk.com/blog/2012/06/compiling-hping-on-centos/

Páginas WEB

http://nmap.org/movies/

http://lamiradadelreplicante.com/2011/12/17/detectar-intrusos-en-la-red-con-nmap-a-lo-trinity/

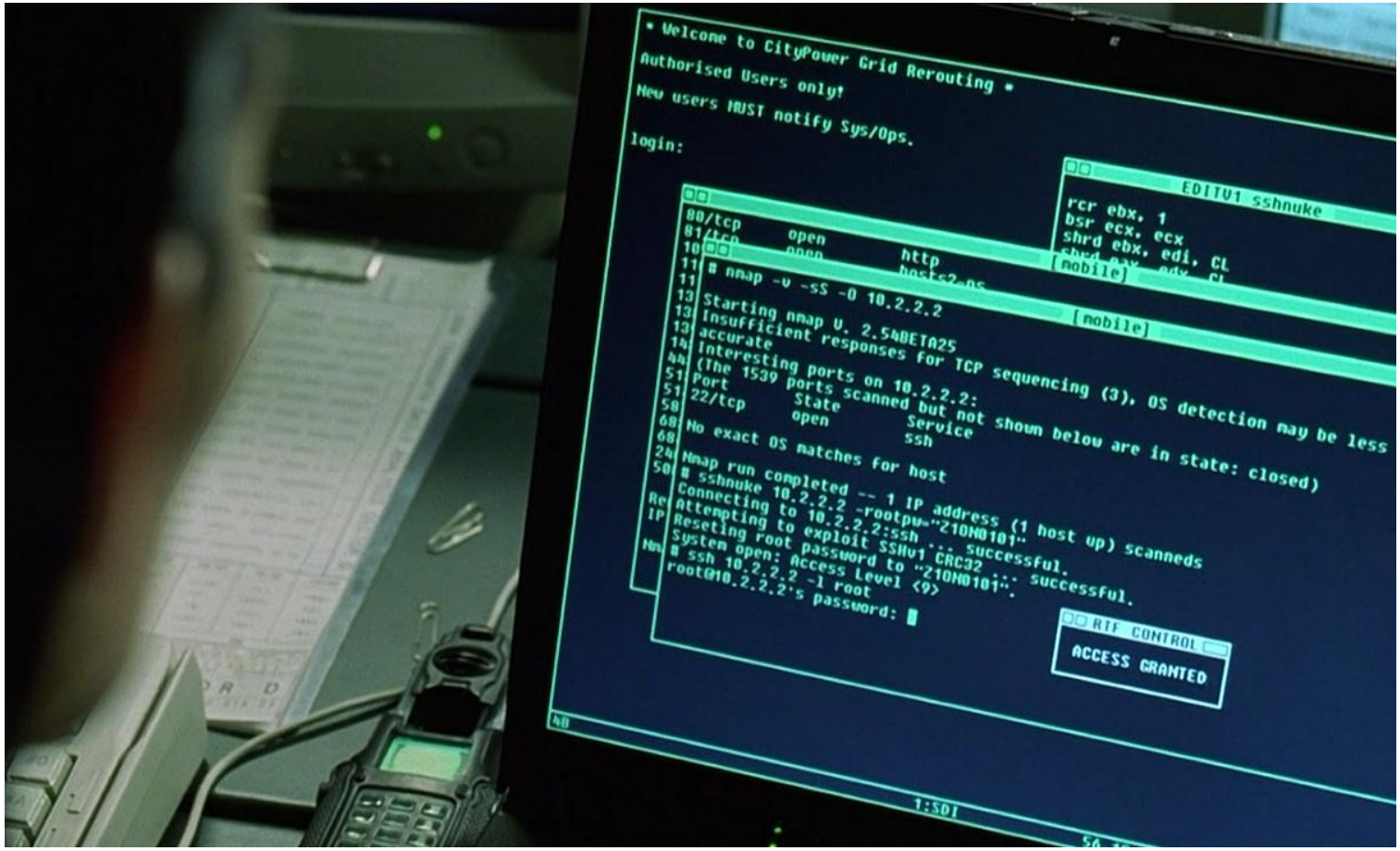
http://lamiradadelreplicante.com/2012/01/24/ataque-ddos-syn-flood-con-hping3/

Asesorías

daniel.barragan@correounivalle.edu.co

Edificio 331 – Oficina 2114

Lunes y Miércoles 3:00 pm – 5:00 pm


* Welcome to CityPower Grid Rerouting *
Authorised Users only!
New users MUST notify Sys/Ops.
login:
EDITV1 sshnuke
rcr ebx, 1
bsr ecx, ecx
shrd ebx, edi, CL
[nobile]
80/tcp open http
nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may be less accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanned
sshnuke 10.2.2.2 -rootpw="Z1ON0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Resetting root password to "Z1ON0101".
System open: Access Level <9>
ssh 10.2.2.2 -l root
root@10.2.2.2's password:
RTF CONTROL
ACCESS GRANTED
1:SDI