ECE458/ECE750T27: Computer Security

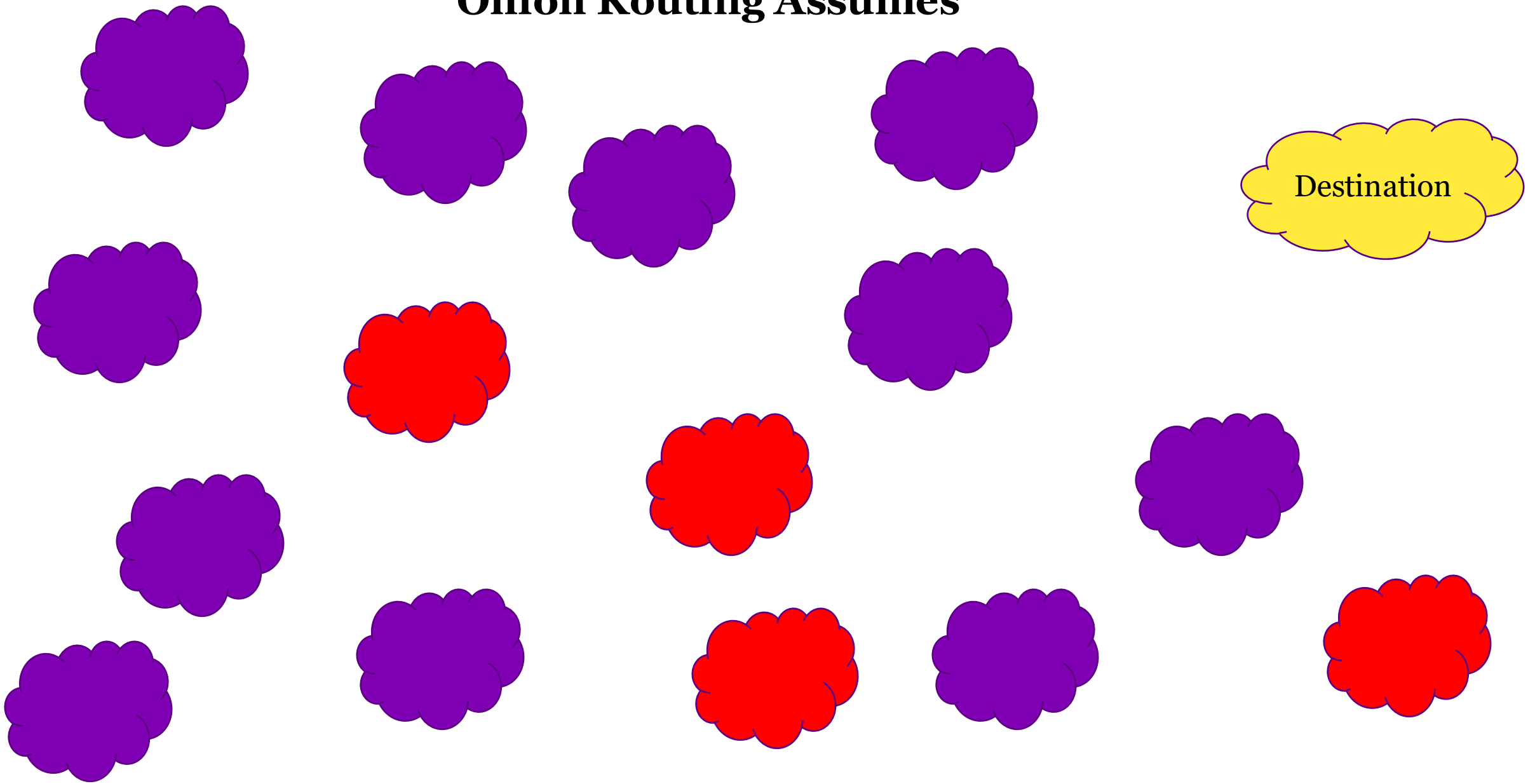
Networking Threats

Dr. Kami Vaniea
Electrical and Computer Engineering
kami.vaniea@uwaterloo.ca

ONION ROUTING

aka Tor

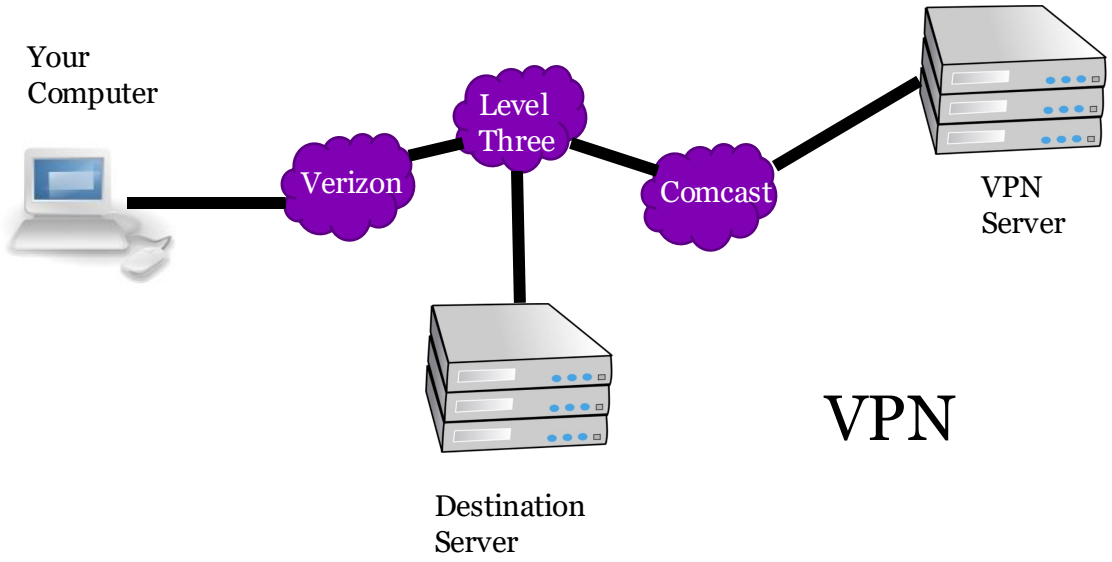
Onion Routing Assumes



Problem: Anonymity

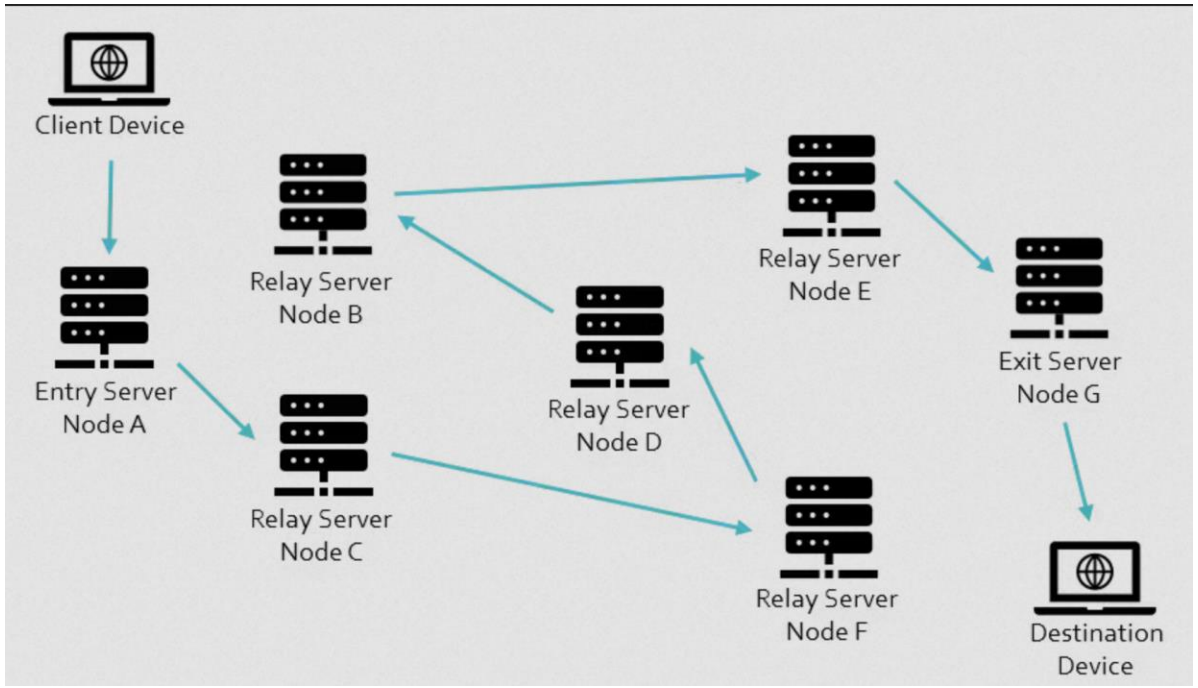
- The user wants to access parts of the internet without being tracked by:
  - Governments
  - Destination server
  - First-hop router
- Avoid being "tracked" means:
  - User's IP address not associated with traffic
  - User's identity cannot be associated with their traffic

- VPN solves some but not all
  - VPN knows who the user is (authentication) and what the traffic is
  - Destination server knows who the VPN is and could theoretically be compelled to provide data (Government attacker)



Onion Routing

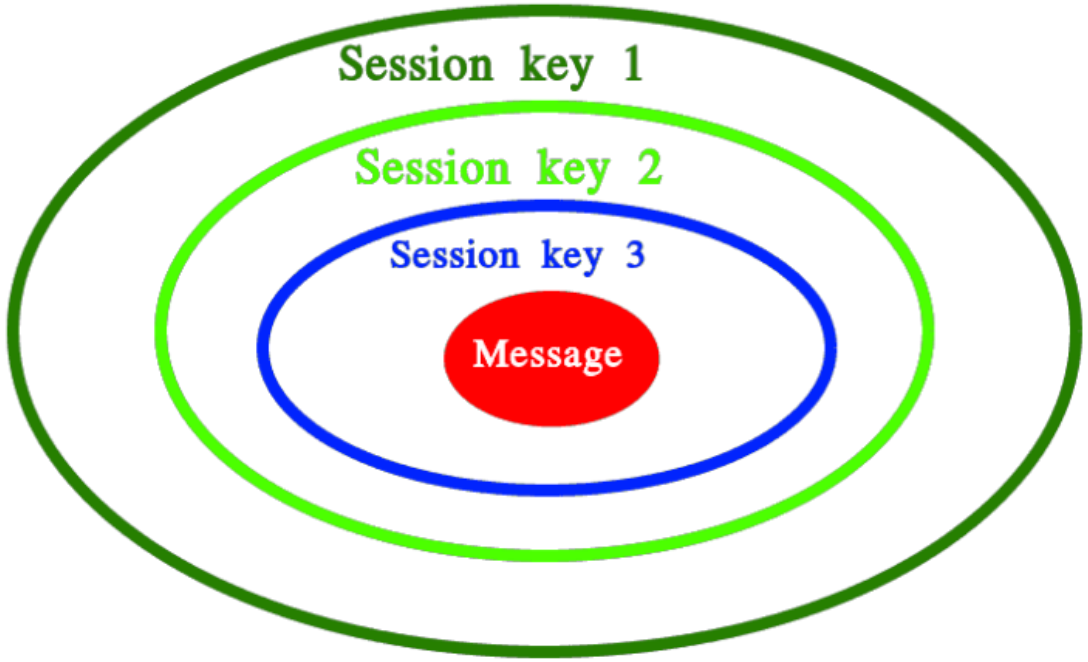
- Goal: allow users to access the internet such that their identity and traffic are not linked
- Each server knows where it got traffic and where it sent traffic, but it doesn't know the whole path, just its neighbors
- This approach protects the client from connecting their real IP address with their traffic
  - First node (Node A) has the real IP address
  - Last node (Node G) has the real traffic
  - Nodes A and G do not know they are carrying the same person's traffic

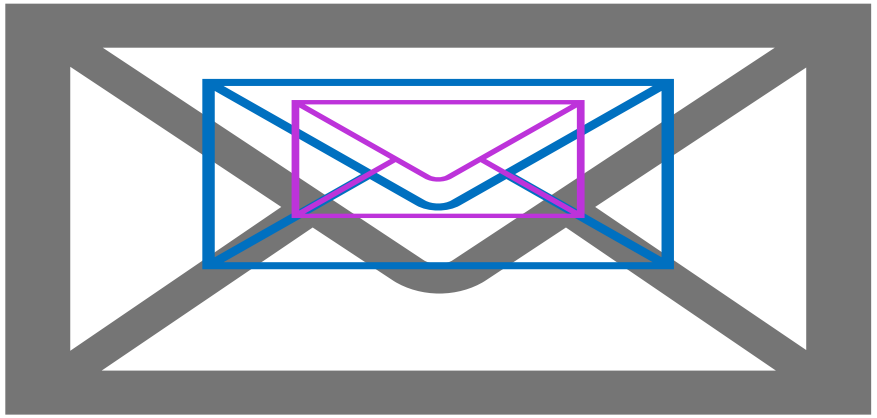


https://privacyhq.com/documentation/onion-routing-explained/

Onion Routing

- Encryption is also important to make all this work
- The client has a list of all onion routers in the network, they select a set and encrypt the message in concentric layers
- Each layer:
  - Encrypted with current node's public key
  - Address of next destination





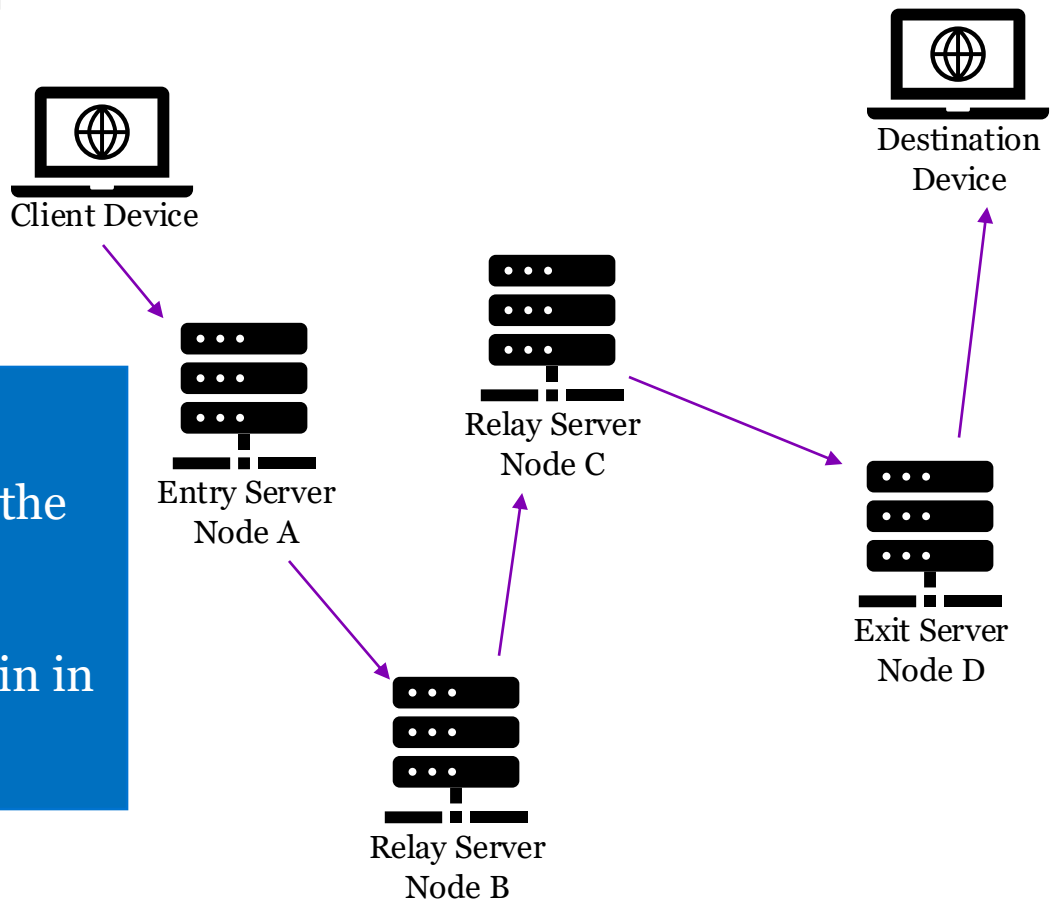
Onion Routing: Client->Destination

- M = Message
- $M_1 = E(M, Destination_{addr}, D_{pub})$
- $M_2 = E(M_1, D_{addr}, C_{pub})$
- $M_3 = E(M_2, C_{addr}, B_{pub})$
- $M_4 = E(M_3, B_{addr}, A_{pub})$

Original Slide
Over-simplified how the route is being built.

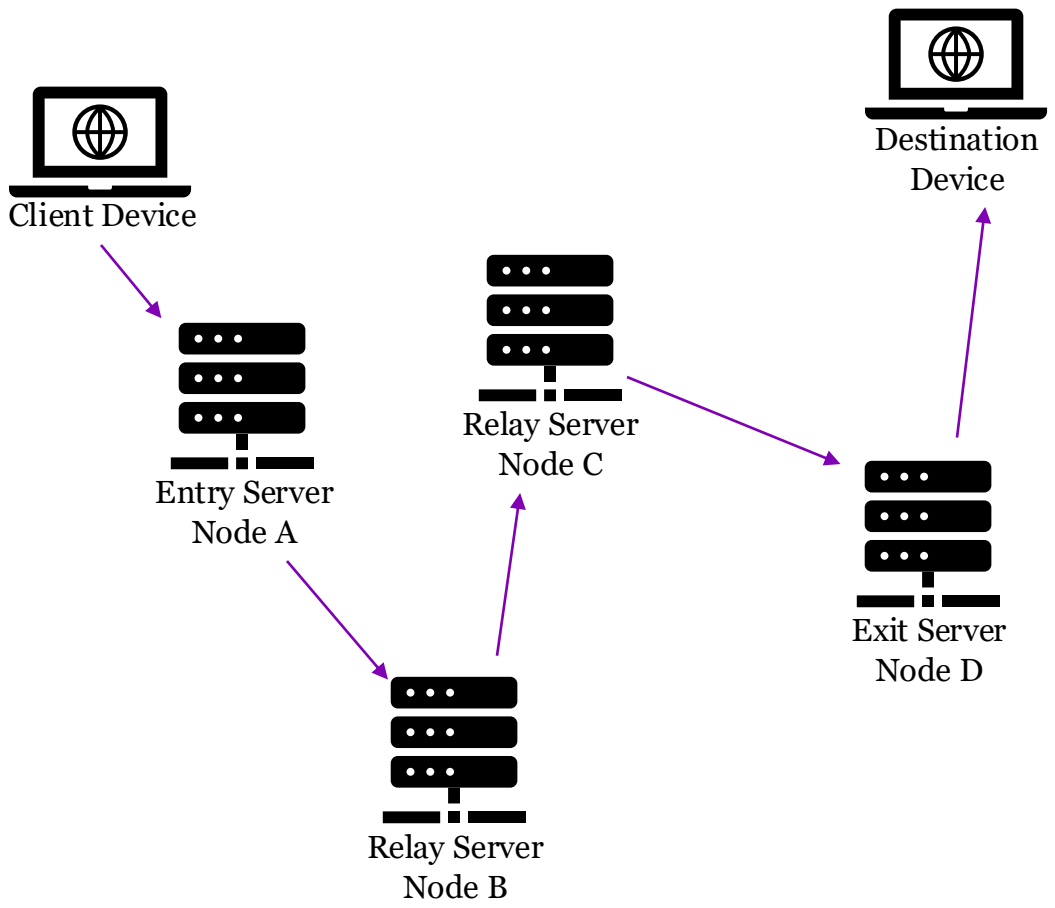
Next few slides explain in more detail.

- Client sends M_4 to Node A



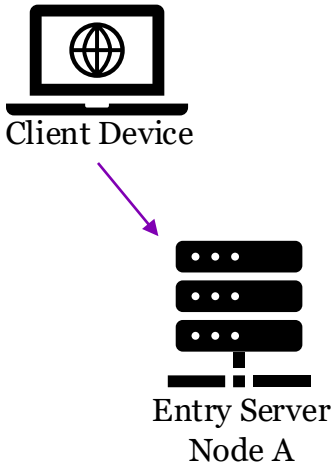
Onion Routing: Client->Destination

- Route iteratively built by client.
- For each node:
  - Send an initiation request including creating a new session (symmetric) key.



Onion Routing: Client->Destination

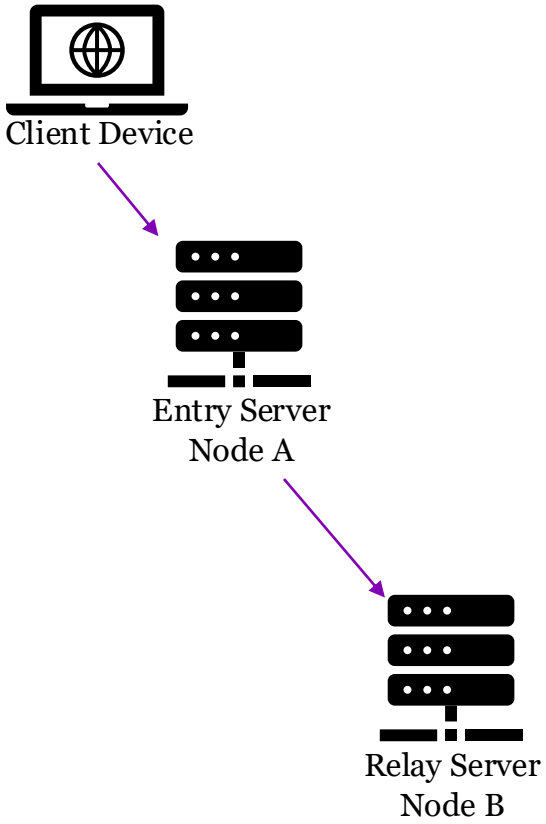
- Client selects a set of nodes and picks a set of session keys such that each node has a different associated session key
  - Session keys could also be negotiated via Diffy-Hillman type approach, pre-selected just easier to explain
- Client then builds the path one node at a time



- M = Start onion route using $A_{session}$
- M_1 = PublicKeyE(M, A_{pub})
- Client sends M_1 to Node A
- Node A decrypts using A_{priv} and now has $A_{session}$ key for future communication

Onion Routing: Client->Destination

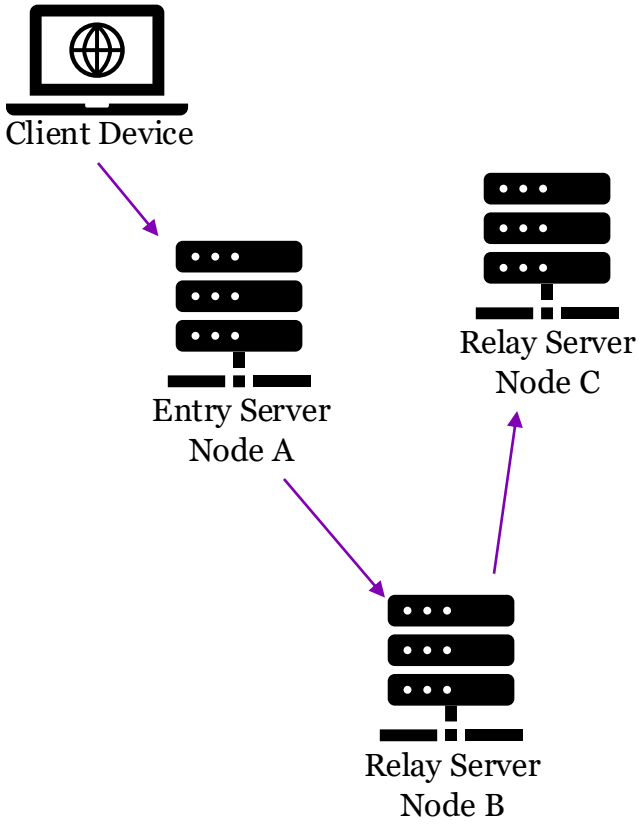
- Client then extends the path to the next planned node B



- M = Start onion route using $B_{session}$
- M_1 = PublicKeyE(M, B_{pub})
- M_2 = SymmetricE(M_1 + B_{addr}, $A_{session}$)
- Client sends M_2 to Node A
- Node A decrypts using $A_{session}$ that was sent earlier and forwards M_1 to B_{addr}
- Node B decrypts using B_{priv} and now has $B_{session}$

Onion Routing: Client->Destination

- Client then extends the path to the next planned node C

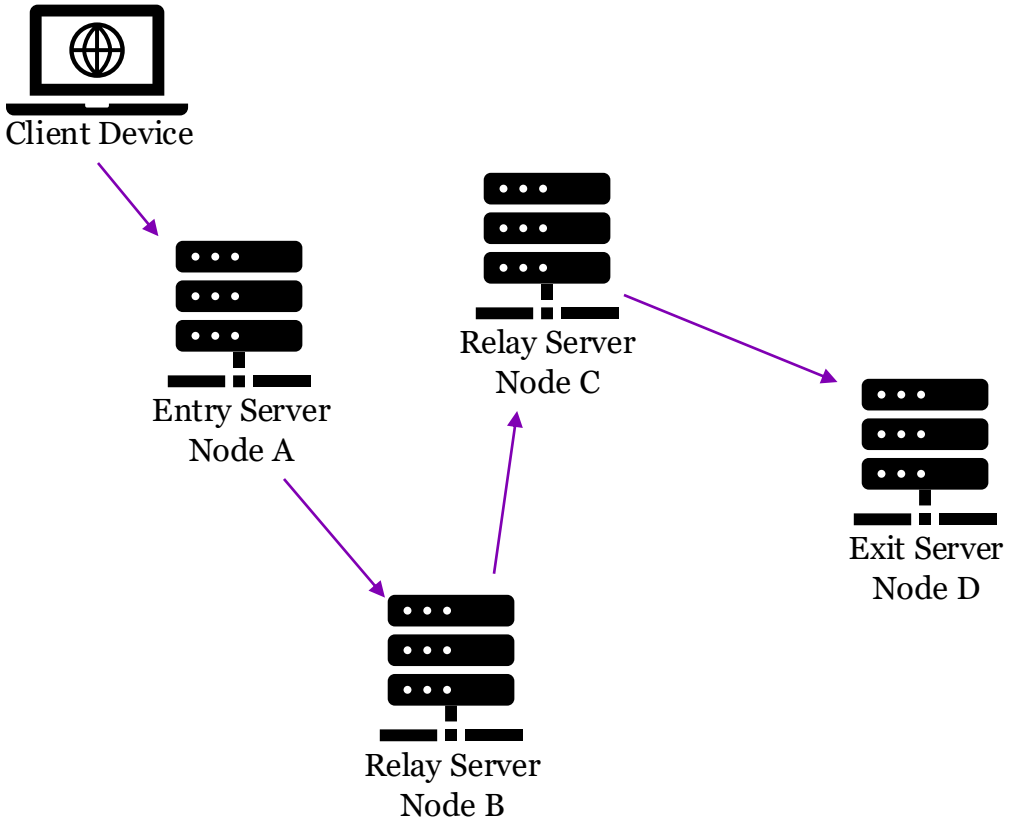


- M = Start onion route using $C_{session}$
- M_1 = PublicKeyE(M, C_{pub})
- M_2 = SymmetricE(M_1 + C_{addr}, $B_{session}$)
- M_3 = SymmetricE(M_2 + B_{addr}, $A_{session}$)
- Client sends M_3 to Node A
- Node A decrypts using $A_{session}$ and forwards M_2 to B_{addr}
- Node B decrypts using $B_{session}$ and forwards M_1 to C_{addr}
- Node C decrypts using C_{priv} and now has $C_{session}$

Onion Routing: Client->Destination

- Client then extends the path to the next planned node D

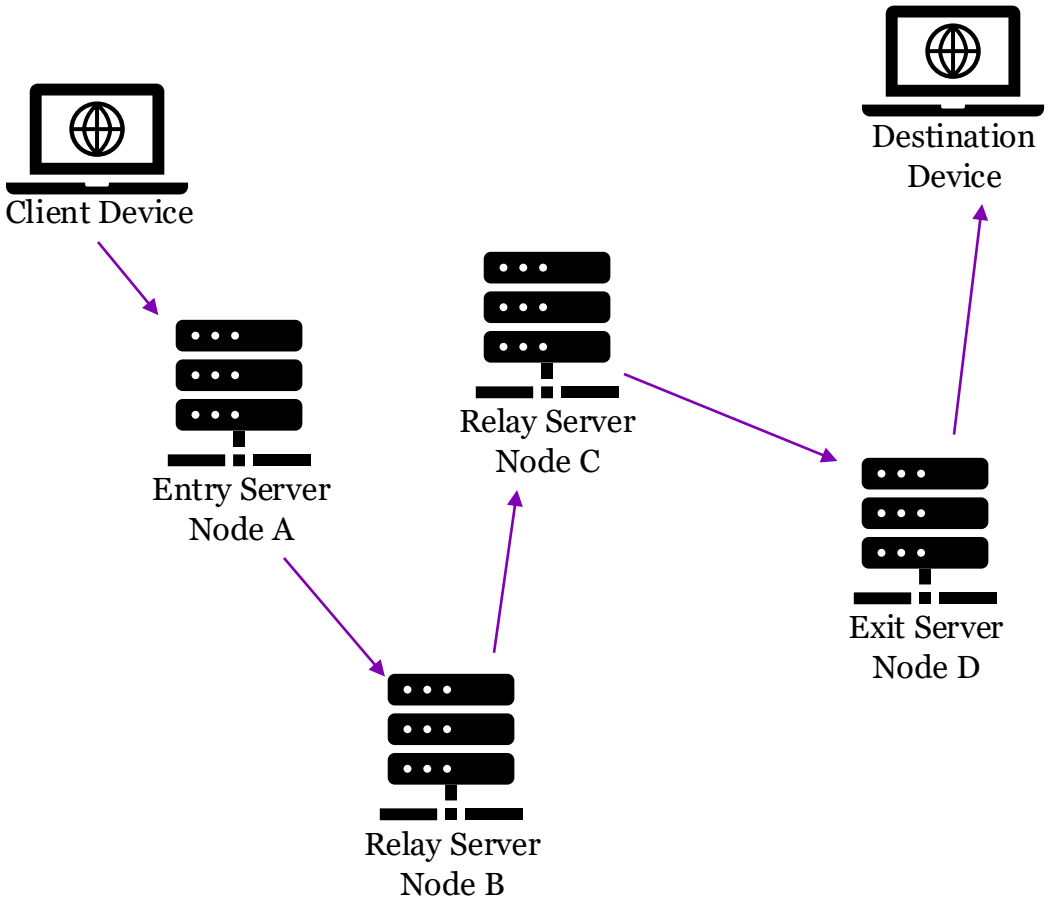
- M = Start onion route using $D_{session}$
- M_1 = PublicKeyE(M, D_{pub})
- M_2 = SymmetricE(M_1 + D_{addr}, $C_{session}$)
- M_3 = SymmetricE(M_2 + C_{addr}, $B_{session}$)
- M_4 = SymmetricE(M_3 + B_{addr}, $A_{session}$)
- Client sends M_4 to Node A
- Node A decrypts using $A_{session}$ and forwards M_3 to B_{addr}
- Node B decrypts using $B_{session}$ and forwards M_2 to C_{addr}
- Node C decrypts using $C_{session}$ and forwards M_1 to D_{addr}
- Node D decrypts using D_{priv} and now has $D_{session}$



Onion Routing: Client->Destination

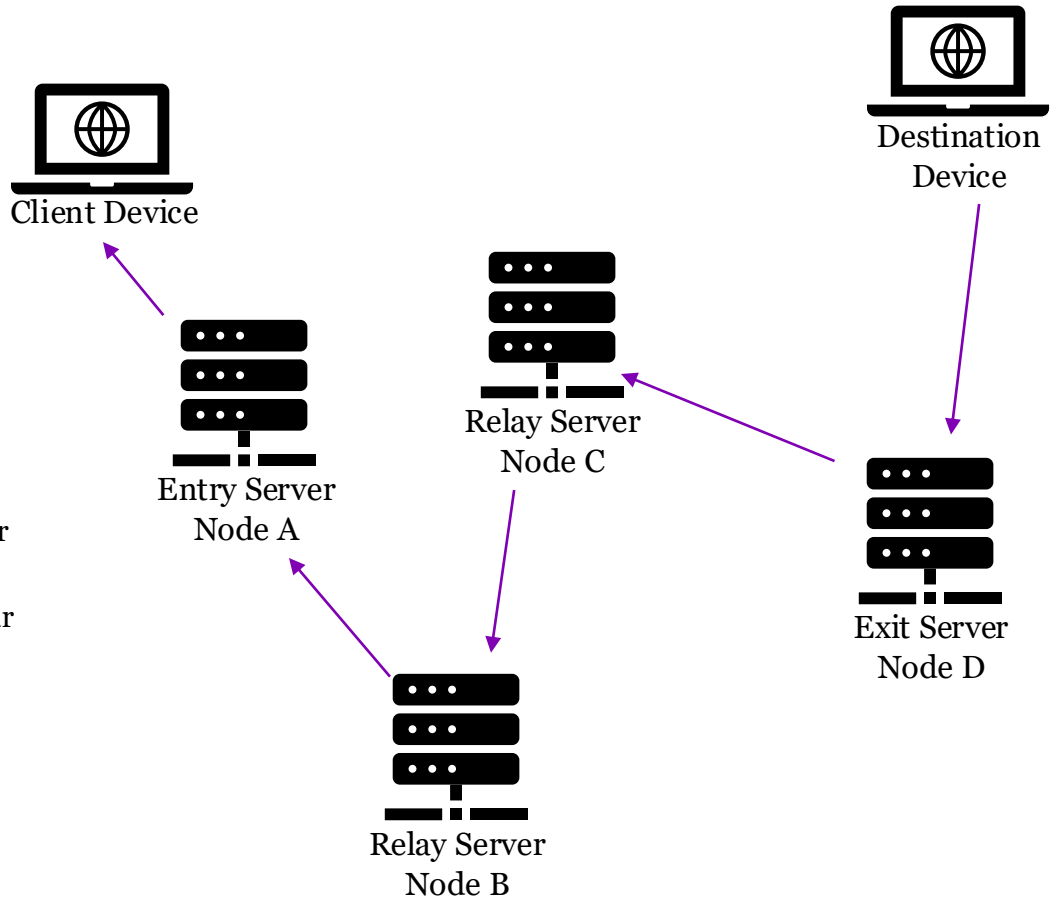
- Client then uses the established path to communicate with destination server

- M = Message
- M_1 = SymmetricE(M + $Destination_{addr}$, $D_{session}$)
- M_2 = SymmetricE(M_1 + D_{addr}, $C_{session}$)
- M_3 = SymmetricE(M_2 + C_{addr}, $B_{session}$)
- M_4 = SymmetricE(M_3 + B_{addr}, $A_{session}$)
- Client sends M_4 to Node A
- Node A decrypts using $A_{session}$ and forwards M_3 to B_{addr}
- Node B decrypts using $B_{session}$ and forwards M_2 to C_{addr}
- Node C decrypts using $C_{session}$ and forwards M_1 to D_{addr}
- Node D decrypts using $D_{session}$ and forwards M to the Destination device



Think-pair-share

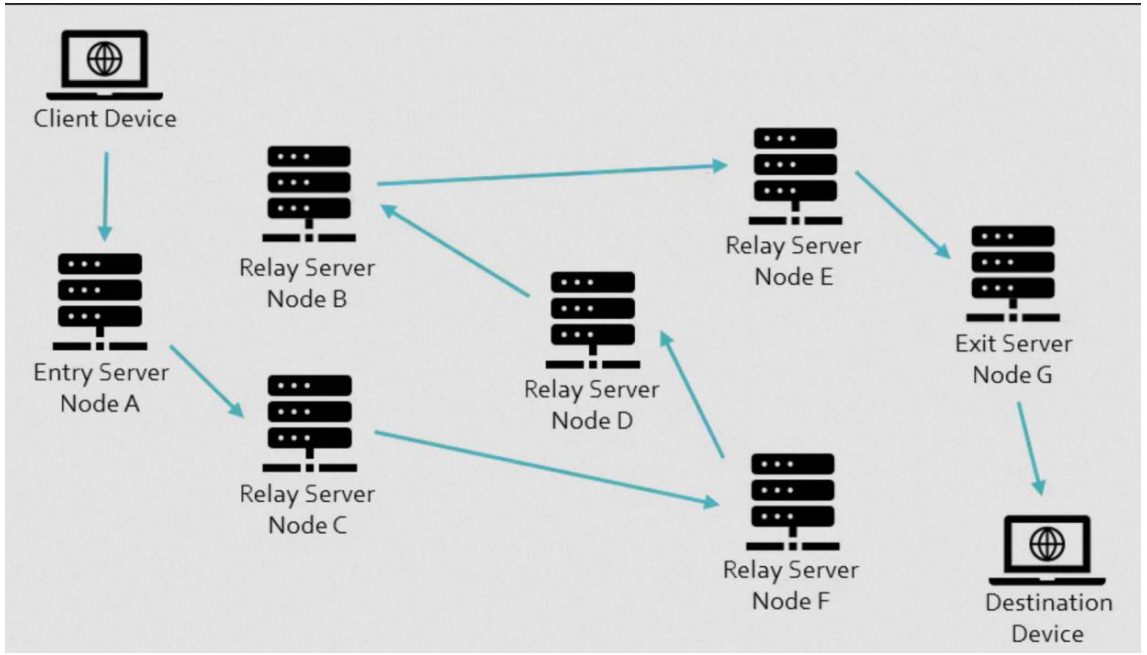
- What would the encryption look like for Destination -> Client



- M = Message is sent to Node D
- Node D: M_1 = SymmetricE(M, $D_{session}$) forwards M_1 to C_{addr}
- Node C: M_2 = SymmetricE(M_1, $C_{session}$) forwards M_2 to B_{addr}
- Node B: M_3 = SymmetricE(M_2, $B_{session}$) forwards M_3 to A_{addr}
- Node A: M_4 = SymmetricE(M_3, $A_{session}$) forwards M_4 to $Client_{addr}$
- Client decrypts M_4 using $A_{session}$
- Client decrypts M_3 using $B_{session}$
- Client decrypts M_2 using $C_{session}$
- Client decrypts M_1 using $D_{session}$
- Resulting in M

Onion Routing

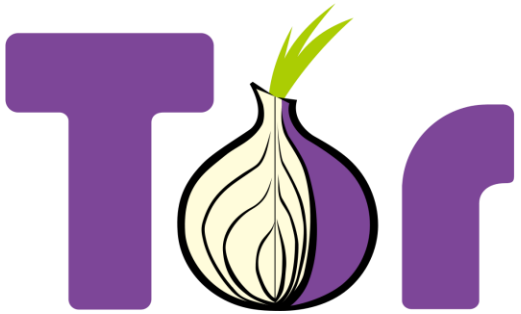
- Each Node only knows the address of where it got the packet and the address of where the packet is going
- They can only decrypt one layer of the packet
- Called onion routing because it is done in layers, with layers constructed by the client and then stripped off by the nodes



https://privacyhq.com/documentation/onion-routing-explained/

Tor

- Tor is popular software that uses onion routing
- There are many ways a user can still show who they are even if using Tor
  - Example: log into Facebook
- Routing everything across Tor could be bad because all exiting traffic could be connected together, so if one bit of traffic leaks your identity, they identity known for all traffic
- Tor is often bundled with a carefully setup browser



TLS

SSL – Old version of TLS

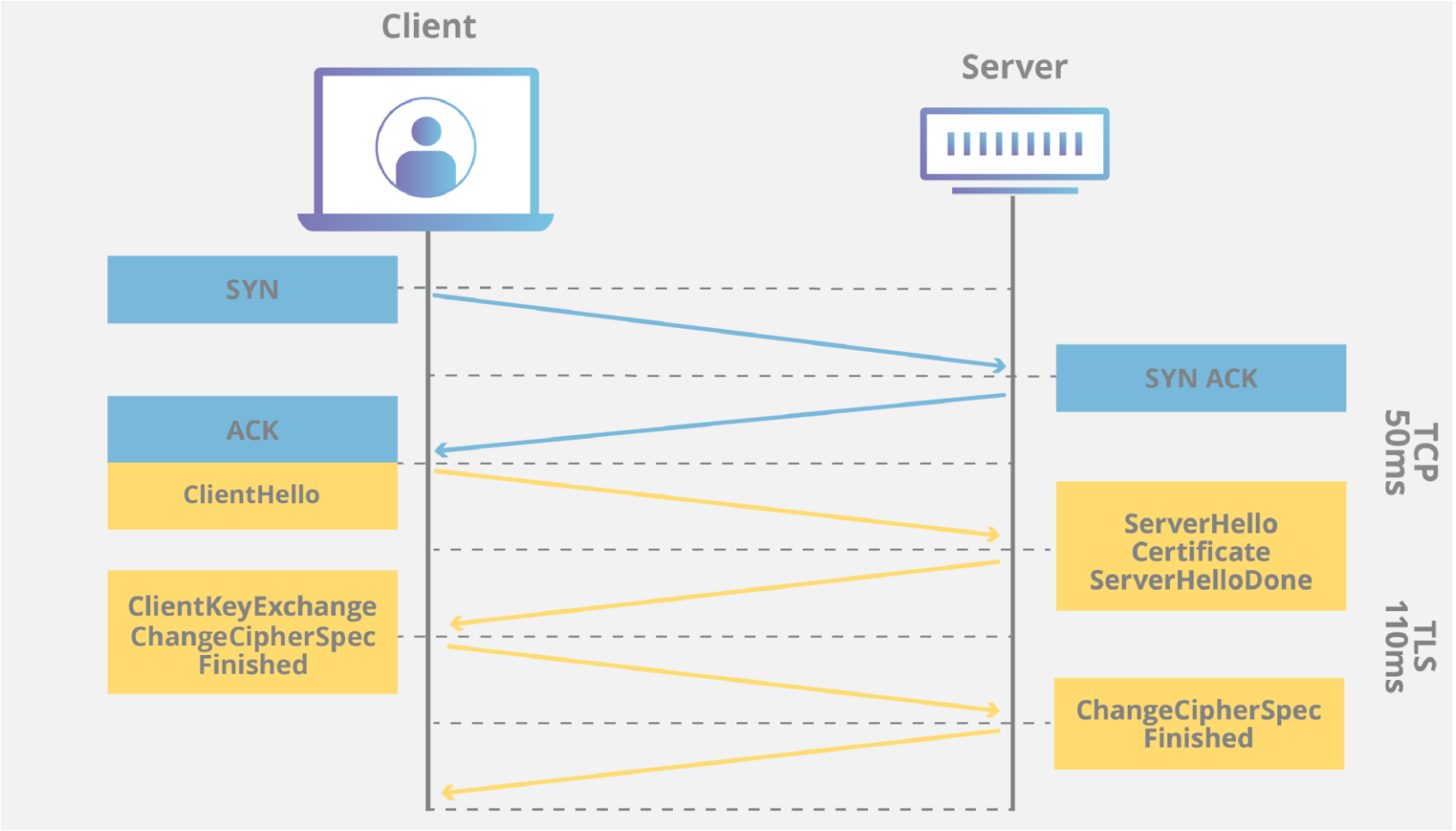
- SSL - Secure Socket Layer
  - Old version
  - Rarely supported any more
  - Name still seems to appear **everywhere**
- TLS – Transport Layer Security
  - New version
  - More secure

Think-pair-share (early view)

- Why does TCP need to happen first?
- What parts of the exchange are public?
- Why are three random byte sets used?

TLS Protocol (RSA)

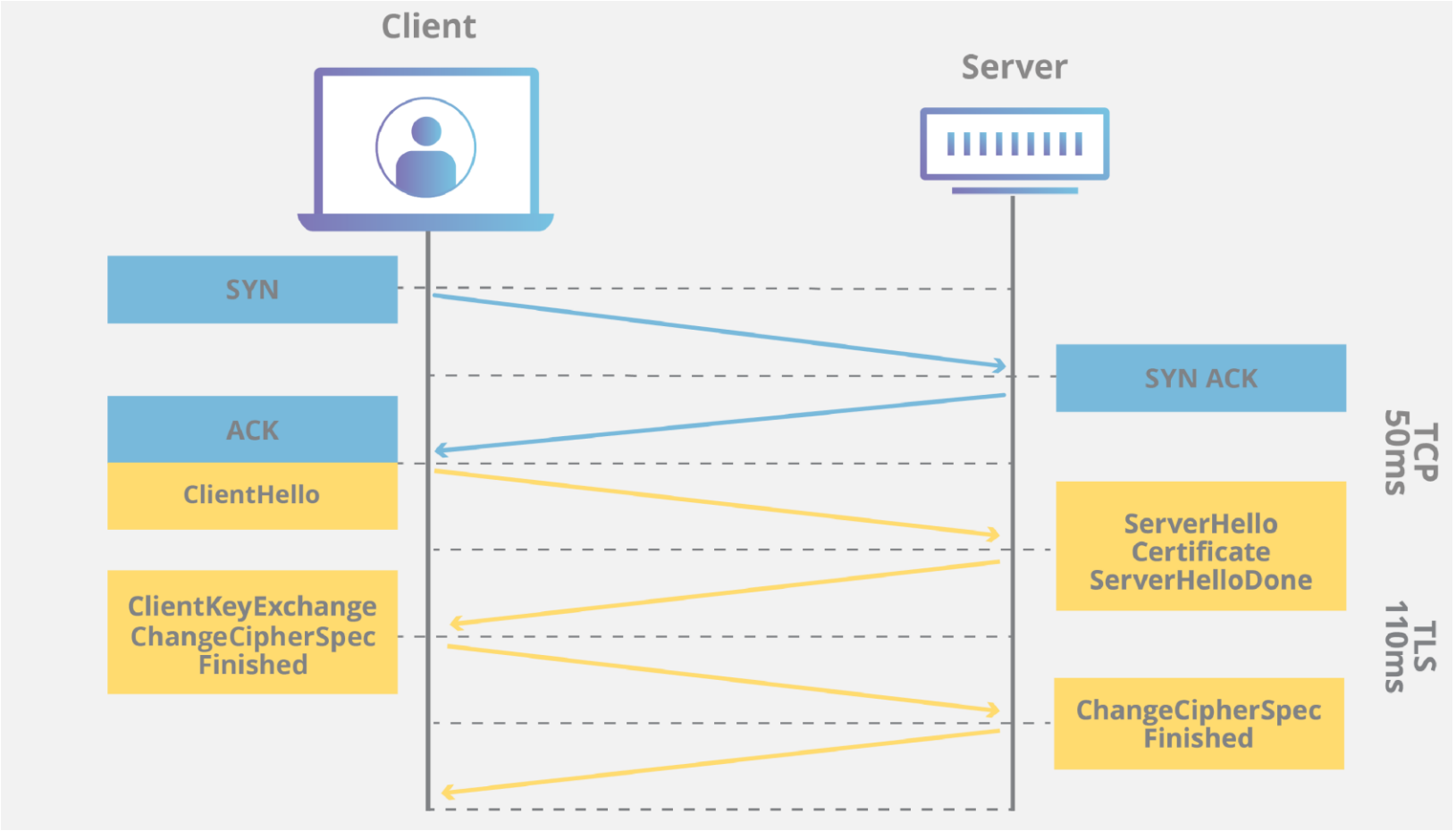
1. TCP handshake
2. Client Hello
   - What protocols the client can support
   - A large random number
3. Server Hello
   - SSL Certificate
   - Chosen cipher suite
   - A different large random number



Cloudflare, "What happens in a TLS handshake?"

TLS Protocol (RSA)

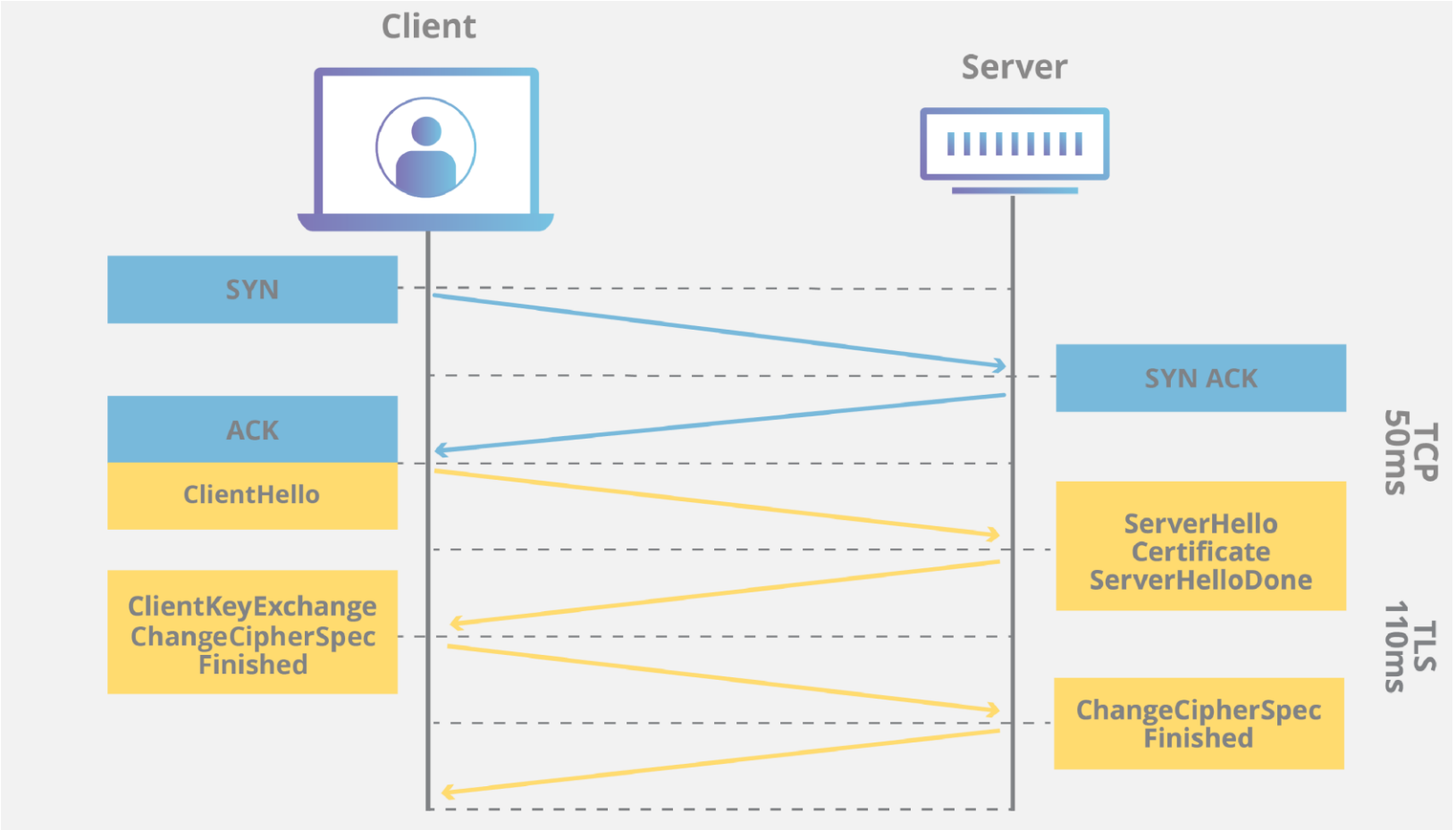
4. Authentication
   - Client verifies SSL certificate
5. The premaster secret
   - Client sends a new random string of bytes encrypted with server's public key (from SSL certificate)
6. Server uses private key to decrypt random bytes



Cloudflare, "What happens in a TLS handshake?"

TLS Protocol (RSA)

7. Session keys created
   - Both computers generate a session key using all three sets of random bytes
8. Client is ready
   - Client sends "finished" encrypted with the session key
9. Server is ready
   - Server sends "finished" encrypted with the session key.



Cloudflare, "What happens in a TLS handshake?"

Think-pair-share

- Why does TCP need to happen first?
- What parts of the exchange are public?
- Why are three random byte sets used?

TLS 1.3

- The new version of TLS:
  - Has a very short list of ciphers it supports
  - Cuts the number of steps down
- Client Hello
- Server generates master secret
- Server Hello and “finished”
- Client verification
- Client “finished”

TLS Protocol

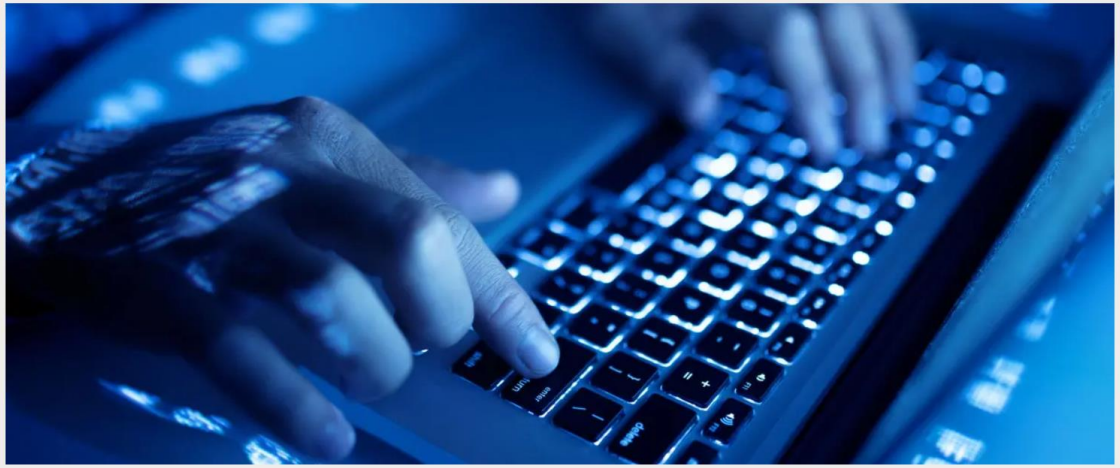
- Certificates are vital to TLS
- But keeping certificates up and running is a common security issue in companies

SONICWALL

Products Solutions Partners Support Company Contact Us

SSL, TLS CERTIFICATES EXPIRING ON US GOVERNMENT SITES DURING FEDERAL SHUTDOWN

by SonicWall Staff



The short- and long-term impacts of the U.S. Federal Government shutdown have been well documented during the last month. Government employees aren't receiving paychecks. National parks are being vandalized. Air travel could soon been at risk.

But as the shutdown carries on, the trust and security of the government's online presence is eroding, too. Many government sites aren't being updated, support and communication is unavailable, and some are completely inaccessible to visitors. And across a handful of government sites (e.g., *.gov* domains), SSL and TLS digital certificates are beginning to expire.

DENIAL OF SERVICE

Denial of Service (DoS): An attack that prevents valid users from accessing a service.

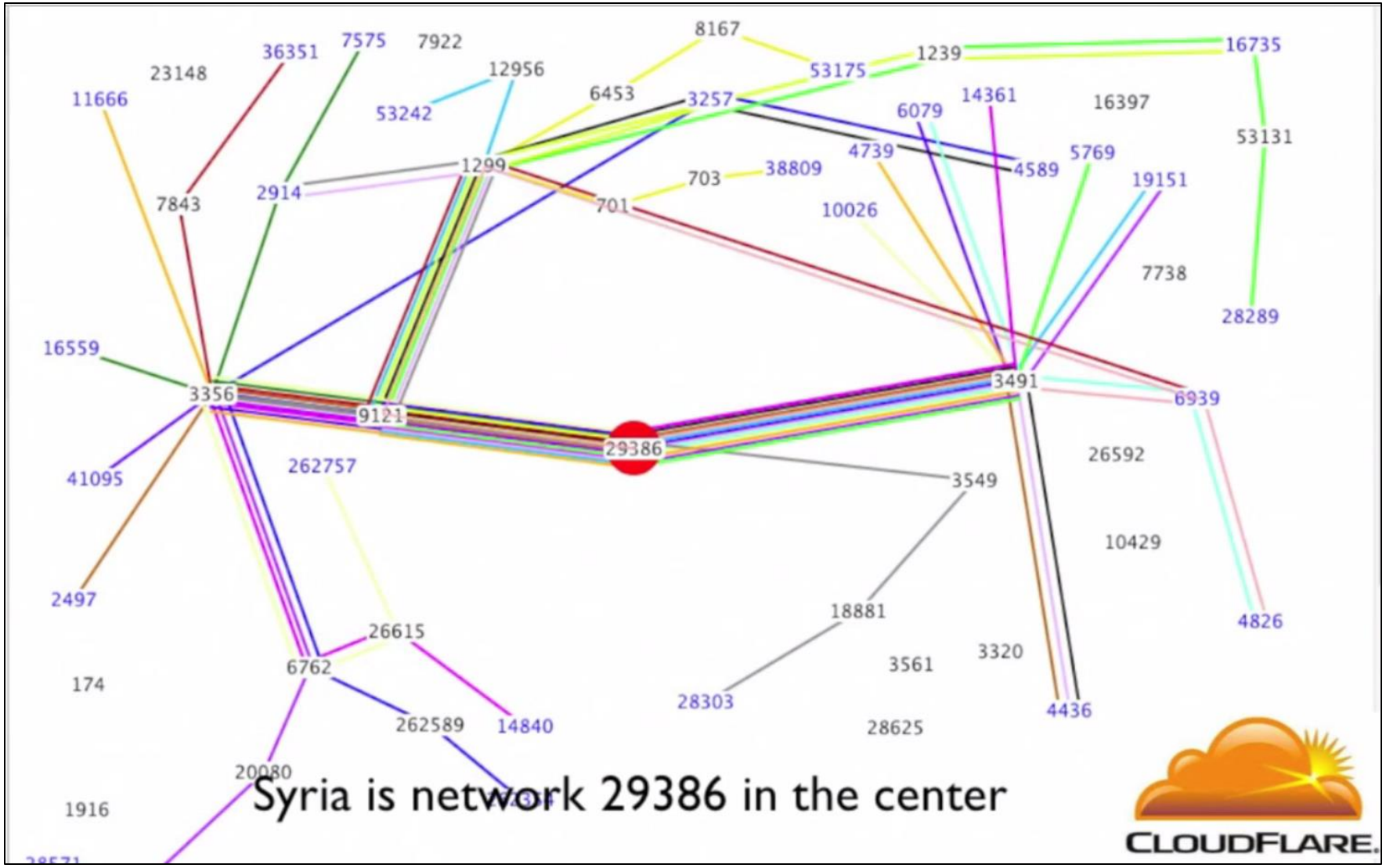
Common examples:

- Cutting power, cables, etc.
- Overloading a server with invalid traffic
- Removing a user account
- Changing the DNS to point to the wrong page

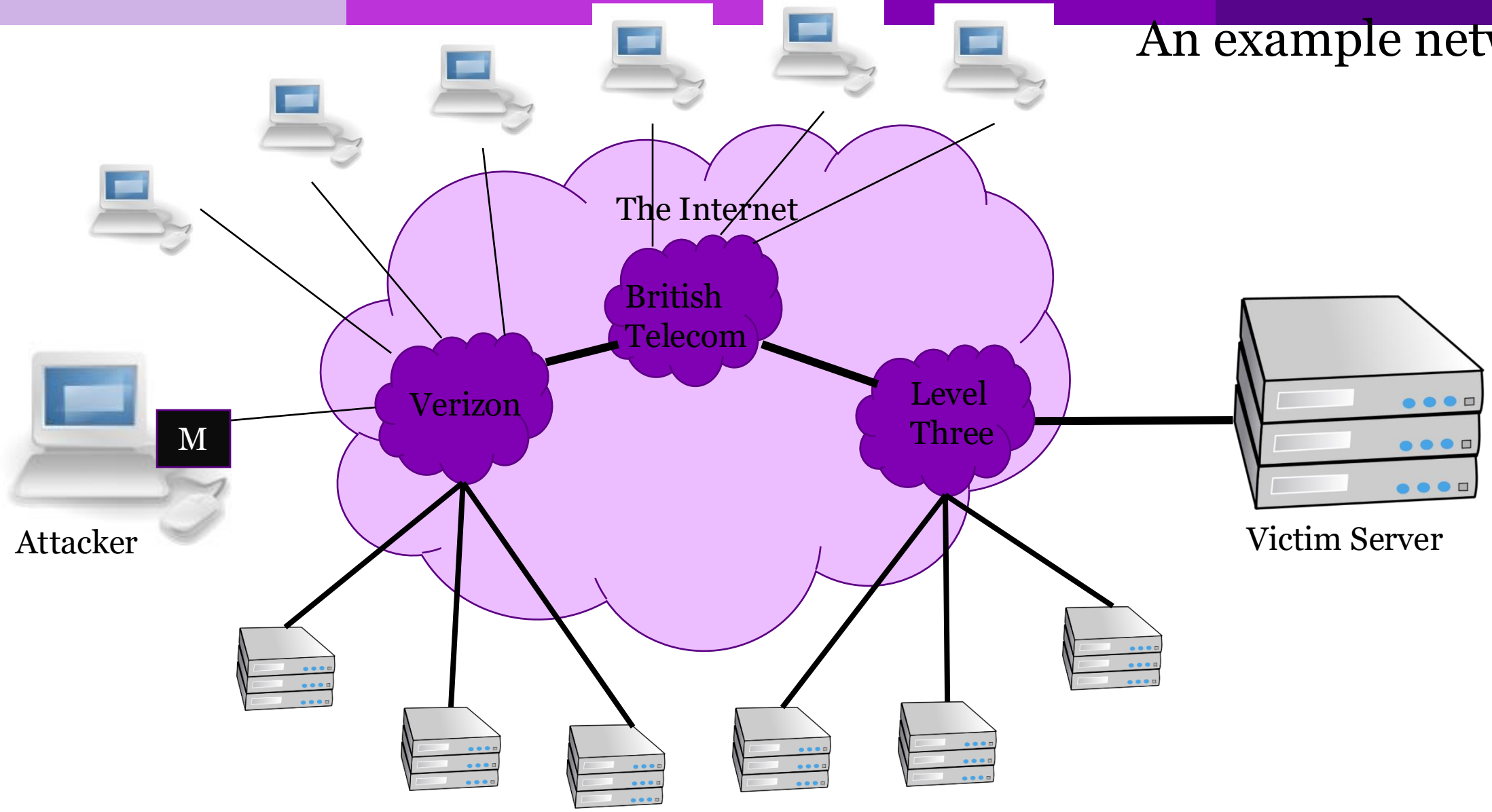
Attacks:

- SYN flooding
- Spoofing
- Smurfing

Syria's network shutdown is a DoS



An example network



SYN Flooding

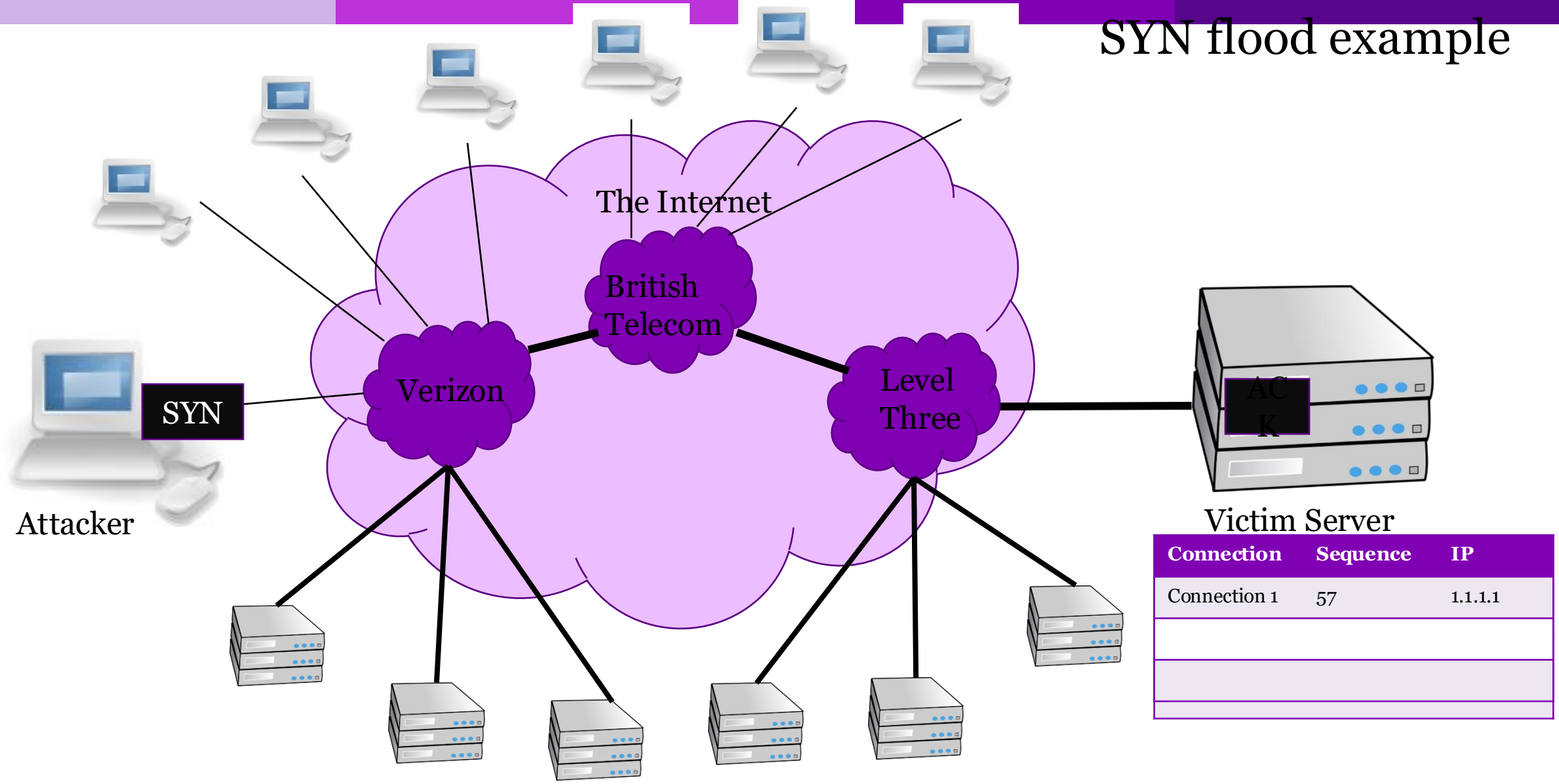
Send tons of requests at the victim and overload them.

- Basic three-part handshake used by Alice to initiate a TCP connection with Bob.

$$
\begin{aligned}
A \rightarrow B: &\quad \texttt{SYN}, X \\
B \rightarrow A: &\quad \texttt{ACK}, X+1; \texttt{SYN}, Y \\
A \rightarrow B: &\quad \texttt{ACK}, Y+1
\end{aligned}
$$

- Alice sends many SYN packets, without acknowledging any replies. Bob accumulates more SYN packets than he can handle.

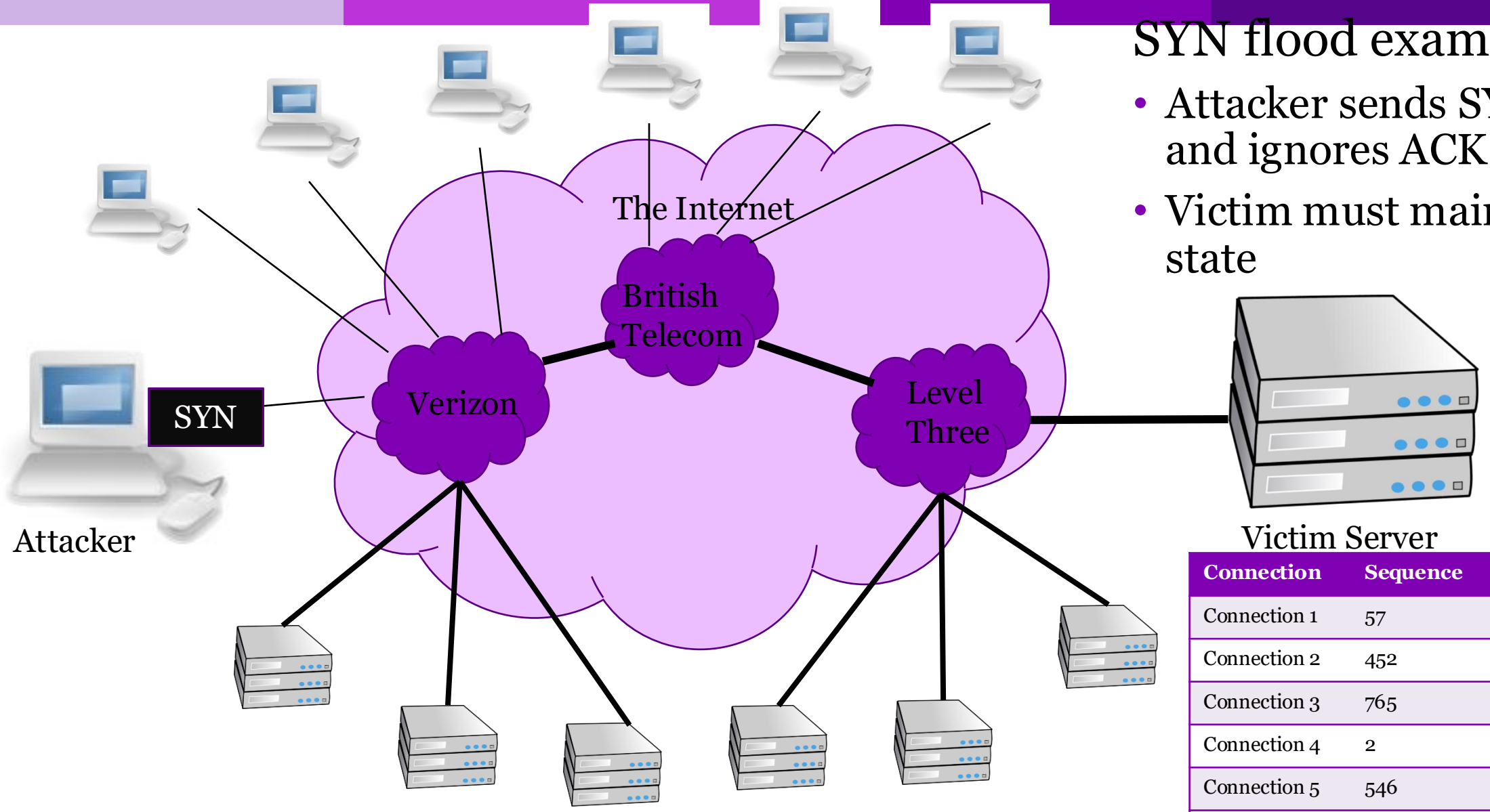
SYN flood example



Connection	Sequence	IP
Connection 1	57	1.1.1.1

SYN flood example

- Attacker sends SYN and ignores ACK
- Victim must maintain state



Connection	Sequence	IP
Connection 1	57	1.1.1.1
Connection 2	452	1.1.1.1
Connection 3	765	1.1.1.1
Connection 4	2	1.1.1.1
Connection 5	546	1.1.1.1
Connection 6	97	1.1.1.1
Connection 7	56	1.1.1.1
Connection 8	15	1.1.1.1

SYN Flooding

- Problems
  - Attribution – attacker users their own IP which could be traced
  - Bandwidth – attacker users their own bandwidth which is likely smaller than a server's
- Effective against a small target
  - Someone running a game server in their home
- Not effective against a large target
  - Company website

Spoofing: forged TCP packets

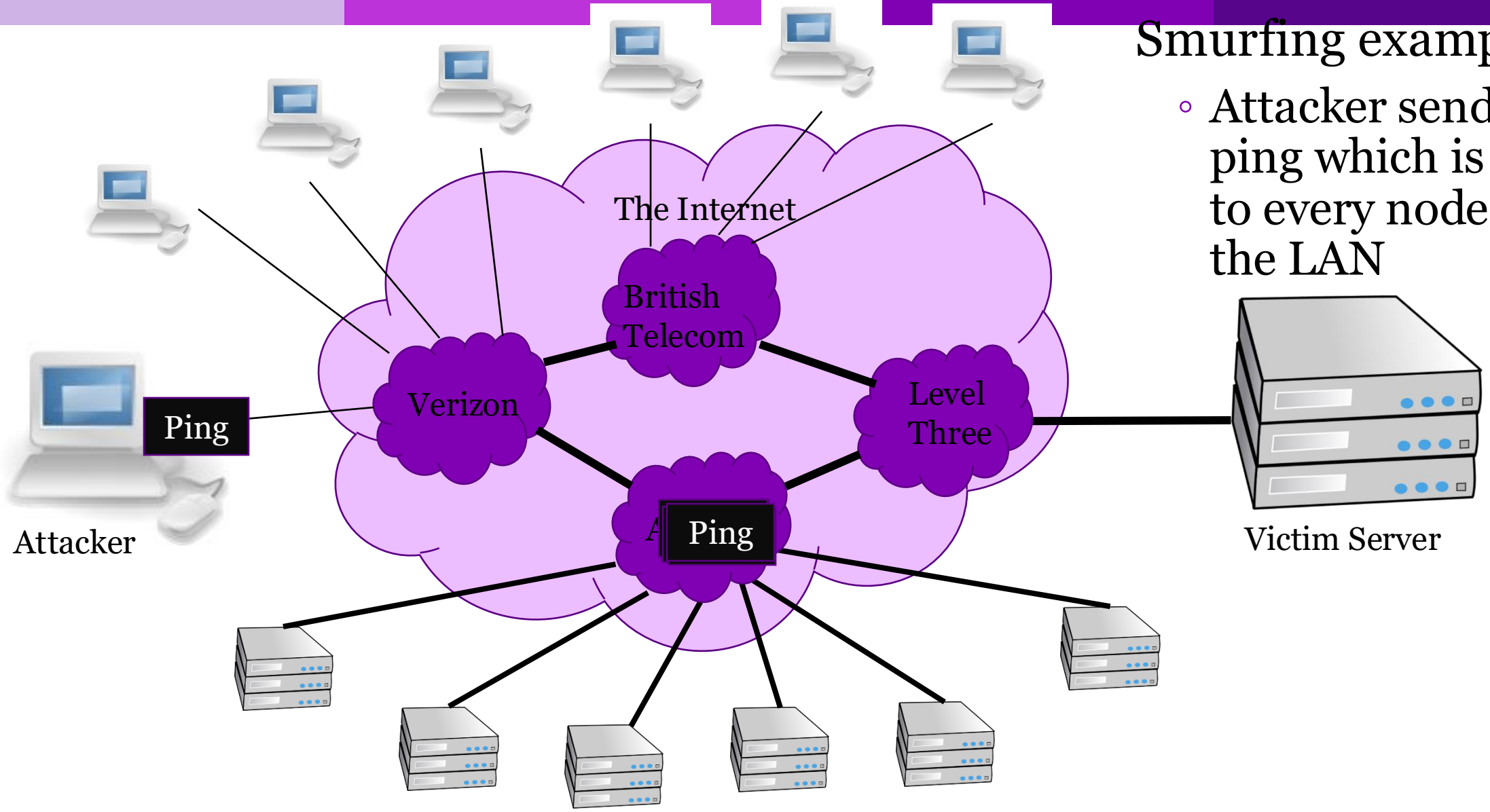
- Same as SYN flooding, but forge the source of the TCP packet
- Advantages:
  - Harder to trace
  - ACKs are sent to a second computer, less attacker bandwidth used
- Problems:
  - Ingress filtering is commonly used to drop packets with source addresses outside their origin network fragment.

Smurfing (directed broadcast)

- The smurfing attack exploits the ICMP (Internet Control Message Protocol) whereby remote hosts respond to echo packets to say they are alive (ping).
- Some implementations respond to pings to broadcast addresses.
- Idea: Ping a LAN to find hosts, which then all respond to the ping.
- Attack: make a packet with a forged source address containing the victim's IP number. Send it to a smurf amplifier, who swamp the target with replies.

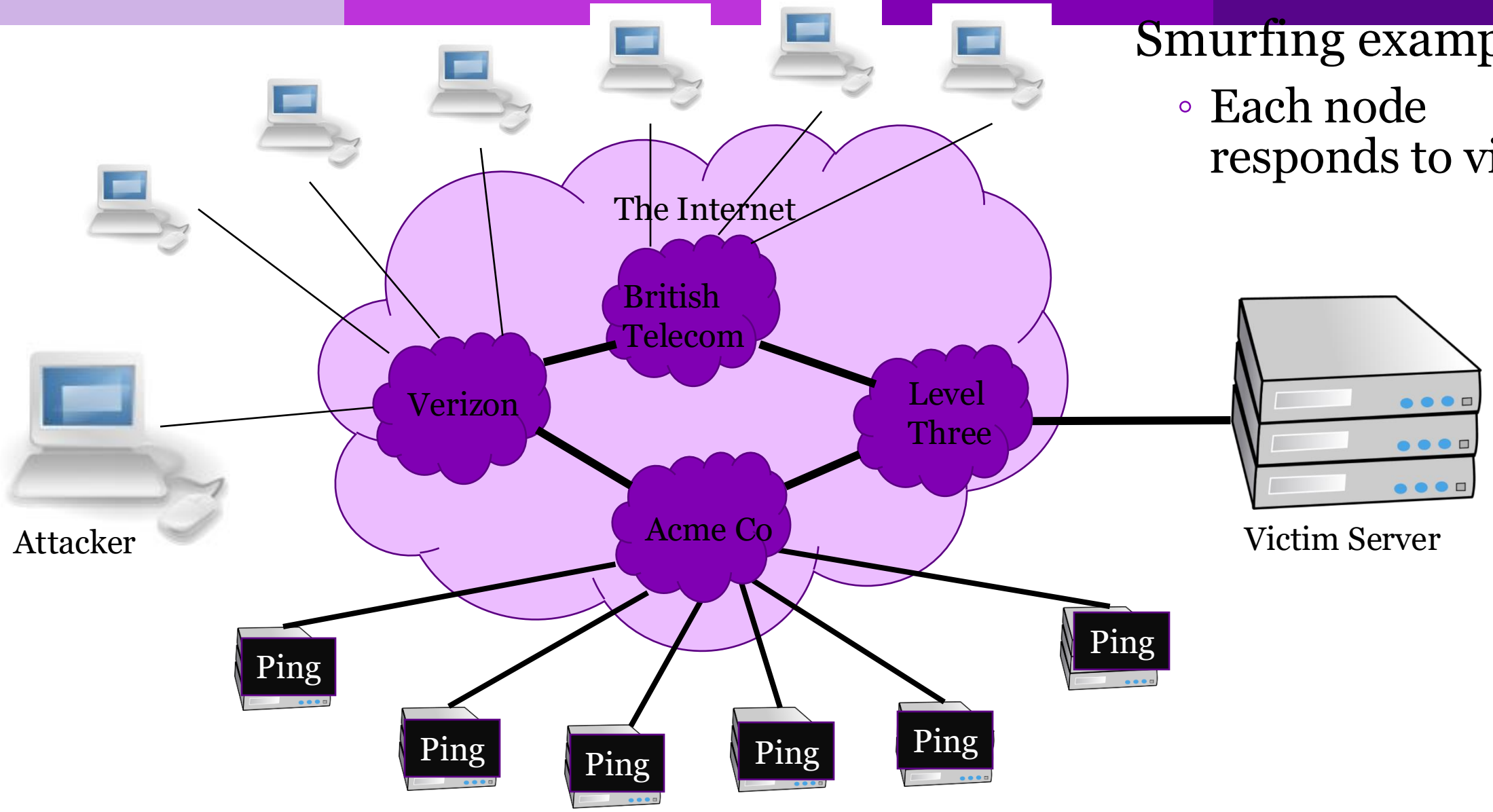
Smurfing example

- Attacker sends 1 ping which is sent to every node on the LAN



Smurfing example

- Each node responds to victim



LANs that allow Smurf attacks are badly configured. One approach is to ban these LANs.

powertech

Smurf Amplifier Registry (SAR)
http://www.powertech.no/smurf/

Current top ten smurf amplifiers (updated every 5 minutes)
(last update: 2016-01-17 23:31:02 CET)

Network	#Dups	#Incidents	Registered at	Home AS
212.1.130.0/24	38	0	1999-02-20 09:41	AS9105
204.158.83.0/24	27	0	1999-02-20 10:09	AS3354
209.241.162.0/24	27	0	1999-02-20 08:51	AS701
159.14.24.0/24	20	0	1999-02-20 09:39	AS2914
192.220.134.0/24	19	0	1999-02-20 09:38	AS685
204.193.121.0/24	19	0	1999-02-20 08:54	AS701
198.253.187.0/24	16	0	1999-02-20 09:34	AS22
164.106.163.0/24	14	0	1999-02-20 10:11	AS7066
12.17.161.0/24	13	0	2000-11-29 19:05	not-analyzed
199.98.24.0/24	13	0	1999-02-18 11:09	AS6199

2457713 networks have been probed with the SAR
56 of them are currently broken
193885 have been fixed after being listed here

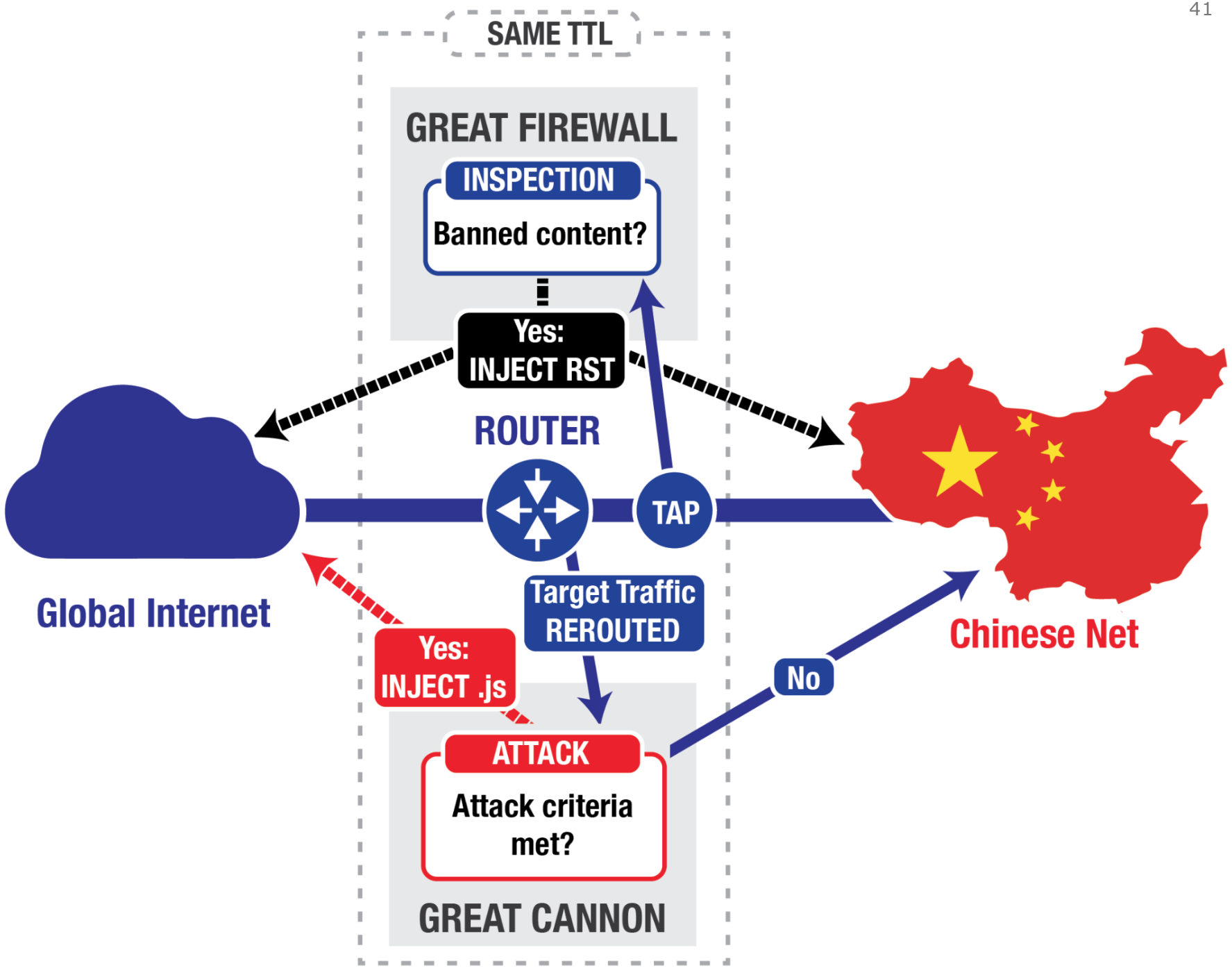
Distributed Denial of Service (DDoS)

A large number of machines work together to perform an attack that prevents valid users from accessing a service.

Common examples:

- Slashdot effect – a large number of valid users all try and access at once.
- Botnets
- Amazon web services

Great Cannon of China is a DDoS attack caused by a MITM attack

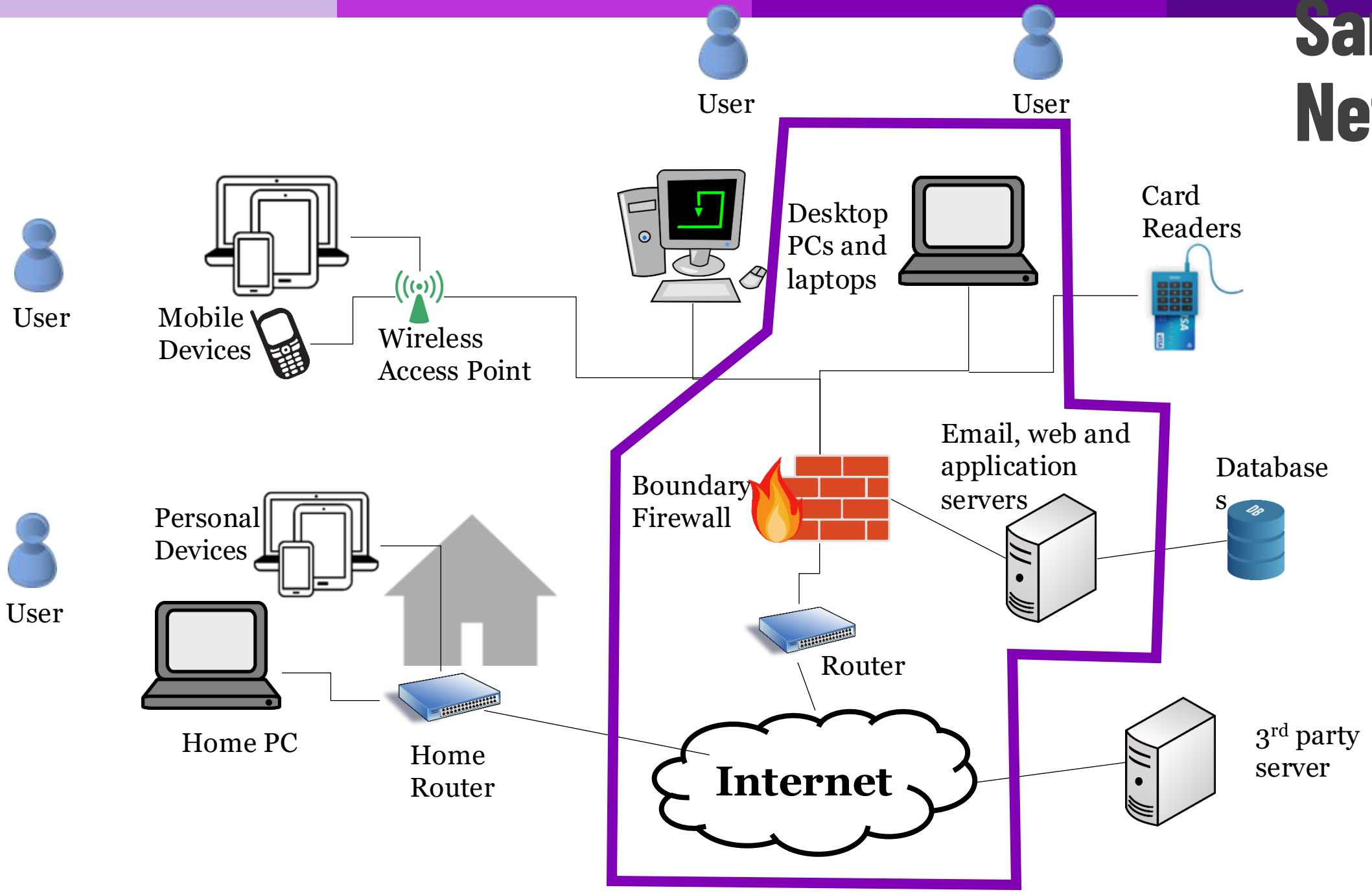


FIREWALLS

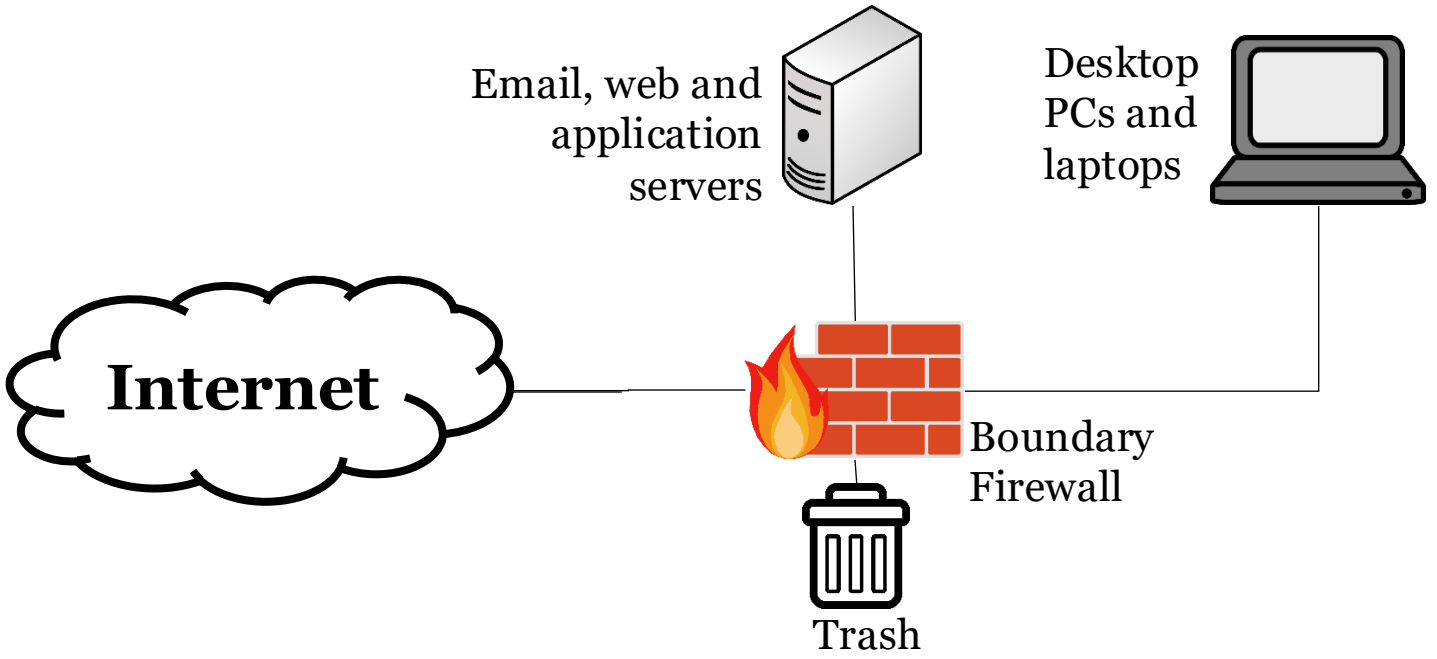
Firewalls

- Firewalls divide the untrusted outside of a network from the more trusted interior of a network
- Often they run on dedicated devices
  - Less possibilities for compromise – no compilers, linkers, loaders, debuggers, programming libraries, or other tools an attacker might use to escalate their attack
  - Easier to maintain few accounts
  - Physically divide the inside from outside of a network

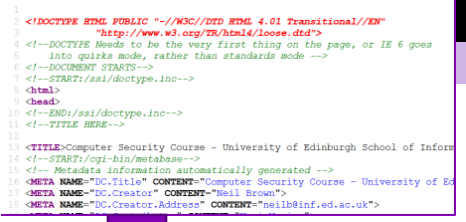
Sample Network



- Questionable things come from the internet AND from the local network
- Firewall applies a set of rules
- Based on rules, it allows or denies the traffic
- Firewalls can also act a routers deciding where to send traffic



Rule	Type	Source Address	Destination Address	Destination Port	Action
1	TCP	*	192.168.1.*	22	Permit
2	UDP	*	192.1681.*	69	Permit
3	TCP	192.168.1.*	*	80	Permit
4	TCP	*	192.168.1.18	80	Permit
5	UDP	*	192.168.1.*	*	Deny



Sender: Apache

7	**Application** — Network process to application
6	**Presentation** — Data representation and encryption
5	**Session** — Interhost communication
4	**Transport** — End-to-end connection and reliability
3	**Network** — Path determination and IP (Logical Addressing)
2	**Data Link** — MAC and LLC (Physical Addressing)
1	**Physical** — Media, signal, and binary transmission

Recipient: Firefox user

7	**Application** — Network process to application
6	**Presentation** — Data representation and encryption
5	**Session** — Interhost communication
4	**Transport** — End-to-end connection and reliability
3	**Network** — Path determination and IP (Logical Addressing)
2	**Data Link** — MAC and LLC (Physical Addressing)
1	**Physical** — Media, signal, and binary transmission

Sender: Apache server

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

A firewall takes in network traffic and compares it to a set of rules. It must process several OSI levels to reach the data it needs.

For example, to filter out all traffic from IP 216.34.181.45 the packet needs to be processed through level 3 where IP addresses can be read.

Firewall

3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

Recipient: Firefox user

7	**Application** Network process to application
6	**Presentation** Data representation and encryption
5	**Session** Interhost communication
4	**Transport** End-to-end connection and reliability
3	**Network** Path determination and IP (Logical Addressing)
2	**Data Link** MAC and LLC (Physical Addressing)
1	**Physical** Media, signal, and binary transmission

Firewall ruleset from a custom home router

- Taken from an arsTechnica article

root@ars-router: ~

```
##### Service rules
# OpenVPN
-A INPUT -p udp -m udp --dport 1194 -j ACCEPT

# ssh - drop any IP that tries more than 10 connections per minute
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m recent --set --name DE
FAULT --mask 255.255.255.255 --rsource
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m recent --update --seco
nds 60 --hitcount 11 --name DEFAULT --mask 255.255.255.255 --rsource -j LOGDROP
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT

# www - accept from LAN
-A INPUT -i p1p1 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -i p1p1 -p tcp -m tcp --dport 443 -j ACCEPT

# DNS - accept from LAN
-A INPUT -i p1p1 -p tcp --dport 53 -j ACCEPT
-A INPUT -i p1p1 -p udp --dport 53 -j ACCEPT

# default drop because I'm awesome
-A INPUT -j DROP

##### forwarding ruleset
```

Image: http://arstechnica.co.uk/gadgets/2016/01/numbers-dont-lie-its-time-to-build-your-own-router/

There are many types of Firewalls

Key differences include:

- How implemented
  - Software – slower, easier to deploy on personal computers
  - Hardware – faster, somewhat safer, harder to add in
- Number of OSI levels of processing required
  - Packet size (level 1)
  - MAC (level 2) and IP (level 3) filtering
  - Port filtering (level 3)
  - Deep packet (level 4+)

Today we will talk about:

- Packet filtering gateway
- Stateful inspection firewall
- Application proxy
- Personal firewalls

Application proxy

- Simulates the (proper) effects of an application at OSI level 7
- Effectively a protective Man In The Middle that screens information at an application layer (OSI 7)
- Allows an administrator to block certain application requests.
- For example:
  - Block all web traffic containing certain words
  - Remove all macros from Microsoft Word files in email
  - Prevent anything that looks like a credit card number from leaving a database

Personal firewalls

- Runs on the workstation that it protects (software)
- Provides basic protection, especially for home or mobile devices
- Malicious software can disable part or all of the firewall
- Any rootkit type software can disable the firewall

QUESTIONS