Displaying Port Scanning and Traceroute with Scapy

Sean Wilkins
Network Engineer & Author

swilkins@infodispersion.com www.infodispersion.com

Introduction

Multiple ways to send and receive packets

Methods can be used to do port scanning and traceroute

Ability to create and send packets is a vital tool

Overview

- Covering the Basics of Send and Receiving Packets
- Concepts Demonstration - Sending and Receiving Packets
- Discussing How to Perform Port Scanning
- Concepts Demonstration - Port Scanning
- Reviewing the Available Traceroute Options
- Concepts Demonstration - Traceroute

Basics

What steps are needed to perform port scans and traceroutes?

Need to cover how to send and receive traffic.

Start with covering basics of sending and receiving packets

Why do you need to know this?

Will need a variety of tools to perform the job

There are many tools available, some are built in as well as add ons

Commands

Covered information will be used in further modules

Common Scapy commands

send, sendp, sr, sr1, srp, and srp1

send Command

```
send(pkt, [inter=0], [loop=0], [count=1], [return_packets=False],
[iface=conf.iface], [filter=None])

Parameters (Incomplete)

        pkt – the packets
        inter – time (in s) between two packets (default 0)
        loop – send packet indefinitely (default 0 or False)
        count – number of packets to send (default None=1)
        return_packets – returns a list of the sent packets
        iface – the interface to send the packets on
        filter - Filters based on BPF statement
```

sendp Command

```
sendp(pkt, [inter=0], [loop=0], [count=1], [return_packets=False],
[iface=conf.iface], [filter=None])

Parameters (Incomplete)

        pkt – the packets
        inter – time (in s) between two packets (default 0)
        loop – send packet indefinitely (default 0 or False)
        count – number of packets to send (default None=1)
        return_packets – returns a list of the sent packets
        iface – the interface to send the packets on
        filter - Filters based on BPF statement
```

sr Command

```
sr(pkt, [inter=0], [timeout=∞], [retry=1], [iface=conf.iface], [filter=None])

Parameters (Incomplete)

  pkt – the packets

  timeout – how much time to wait after the last packet has been sent (Defaults
  to infinity)

  inter – time (in s) between two packets (default 0)

  retry – if positive, how many times to resend unanswered packets if negative,
  how many times to retry when no more packets are answered (Default is 1)

  iface – the interface to send the packets on

  multi – whether to accept multiple answers for the same stimulus

  filter - Filters based on BPF statement
```

sr1 Command

```
sr1(pkt, [inter=0], [timeout=∞], [retry=1], [iface=conf.iface], [filter=None])
```

Parameters (Incomplete)

- pkt – the packets
- timeout – how much time to wait after the last packet has been sent (Defaults to infinity)
- inter – time (in s) between two packets (default 0)
- retry – if positive, how many times to resend unanswered packets if negative, how many times to retry when no more packets are answered (Default is 1)
- iface – the interface to send the packets on
- multi – whether to accept multiple answers for the same stimulus
- filter - Filters based on BPF statement

srp Command

```
srp(pkt, [inter=0], [timeout=∞], [retry=1], [iface=conf.iface], [filter=None])
```

Parameters (Incomplete)

pkt – the packets timeout – how much time to wait after the last packet has been sent (Defaults to infinity)

inter – time (in s) between two packets (default 0)

retry – if positive, how many times to resend unanswered packets if negative, how many times to retry when no more packets are answered (Default is 1)

iface – the interface to send the packets on

multi – whether to accept mu6ltiple answers for the same stimulus

filter - Filters based on BPF statement

srp1 Command

```
srp1(pkt, [inter=0], [timeout=∞], [retry=1], [iface=conf.iface], [filter=None])
```

Parameters (Incomplete)

pkt – the packetsvtimeout – how much time to wait after the last packet has been sent (Defaults to infinity)

inter – time (in s) between two packets (default 0)

retry – if positive, how many times to resend unanswered packets if negative, how many times to retry when no more packets are answered (Default is 1)

iface – the interface to send the packets on

multi – whether to accept multiple answers for the same stimulus

filter - Filters based on BPF statement

Final Points

Send and receive packet commands “see” their responses

Scapy can miss return packets

Use optional parameters to avoid missed returns send and sendp commands sent out traffic quickly

sr, sr1, srp, and srp1 are limited by the timeout

Scapy will wait ~ 2 secs for a response

May need a separate process that responds first

arping command is built into Scapy performs quicker

Demo

Scapy Sending and Receiving Packets

- send
- sendp
- sr
- sr1
- srp
- srp1

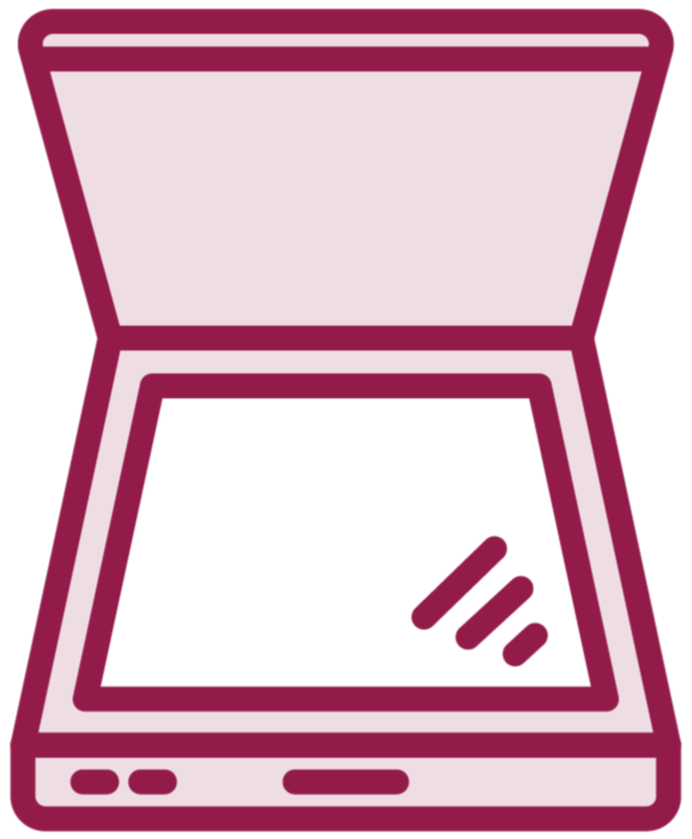
Let’s look at how these scans are performed

Port Scan

Again, why is this information important?

Ability to create, modify, send and receive traffic extremely important

The knowledge will be used to perform a port scan



Opens ports can give information on device’s status

Open port numbers can tell what service types are running

Helps determine what applications are used and the OS

Should only have ports open for services that are required

Port Scans

Common traditionally used port scans

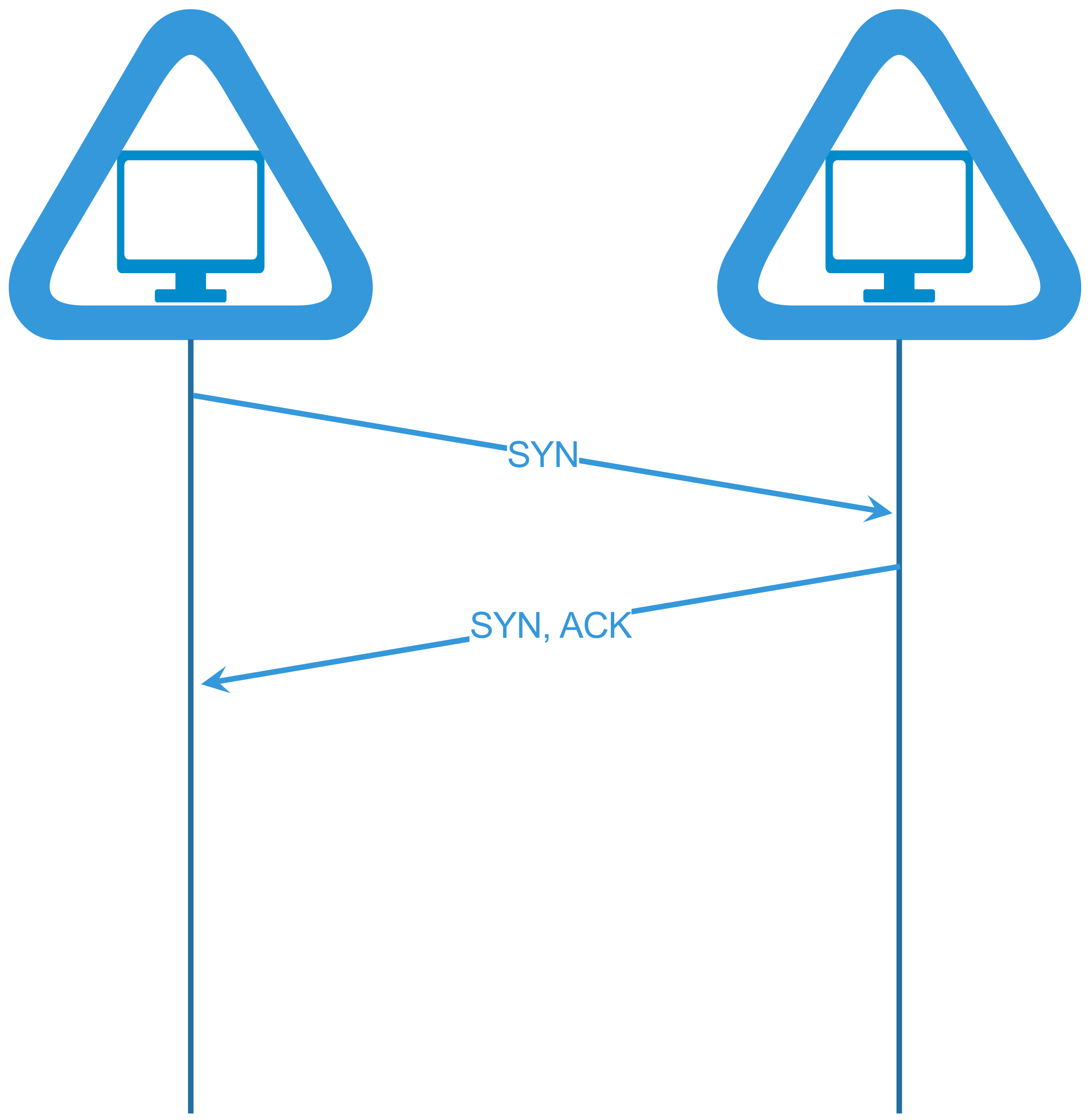
- TCP Scan (half open scan)
- FIN Scan
- NULL Scan
- XMAS Scan
- UDP Scan

Designed to find available ports

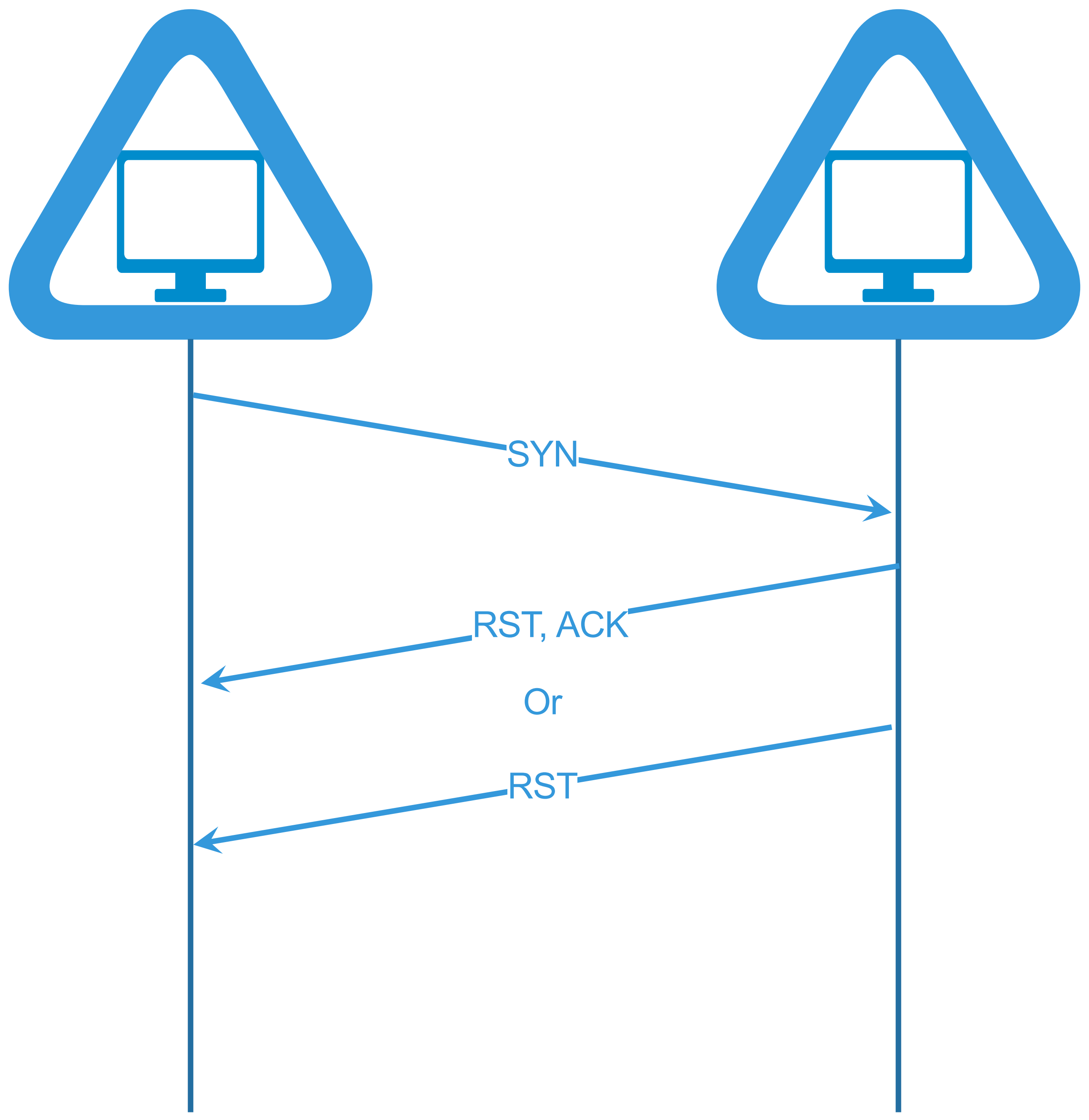
May be the same technique used to open a port

Others try to find ports that may not show as open

TCP Port Scan

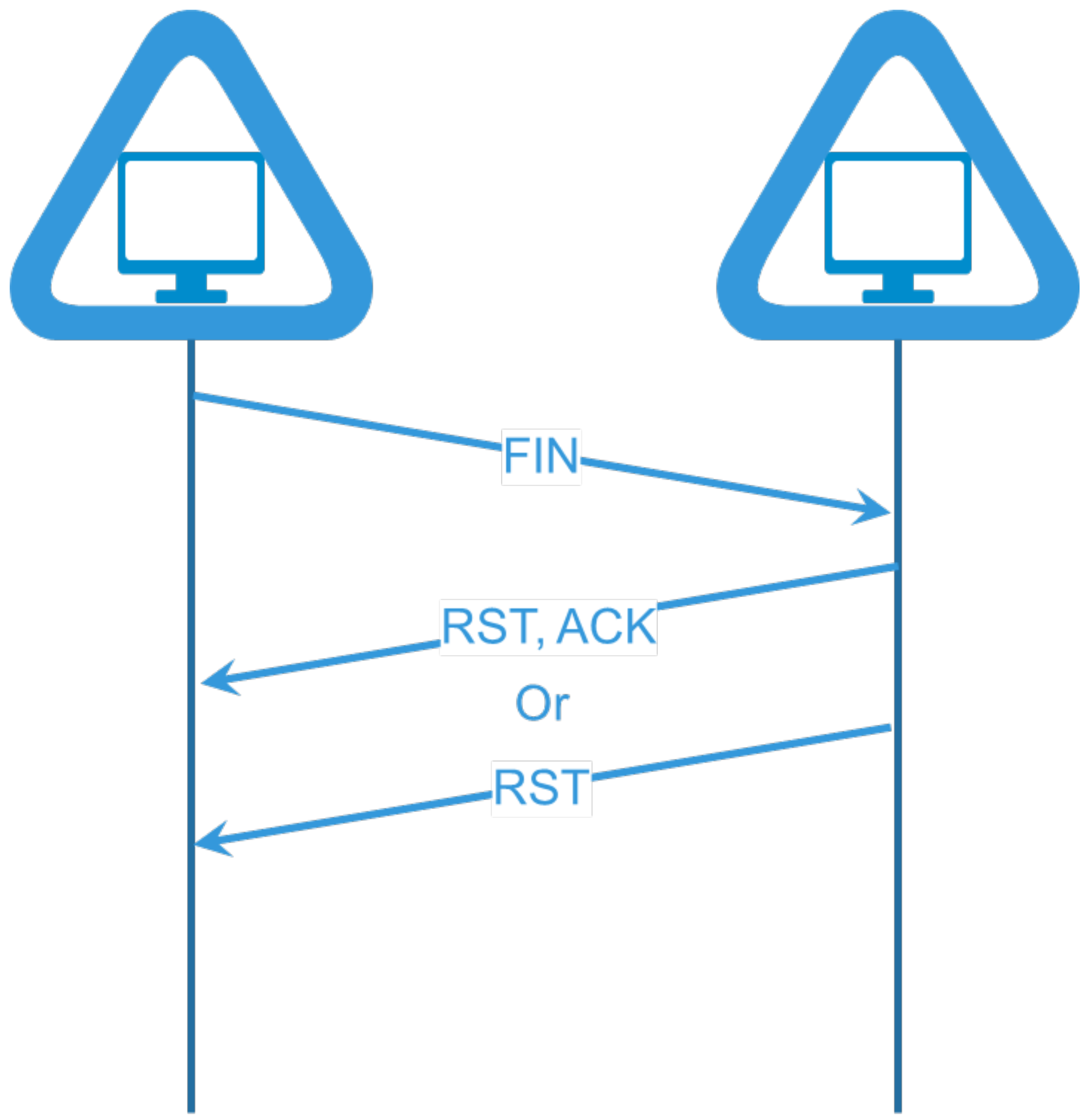


TCP Port Scan

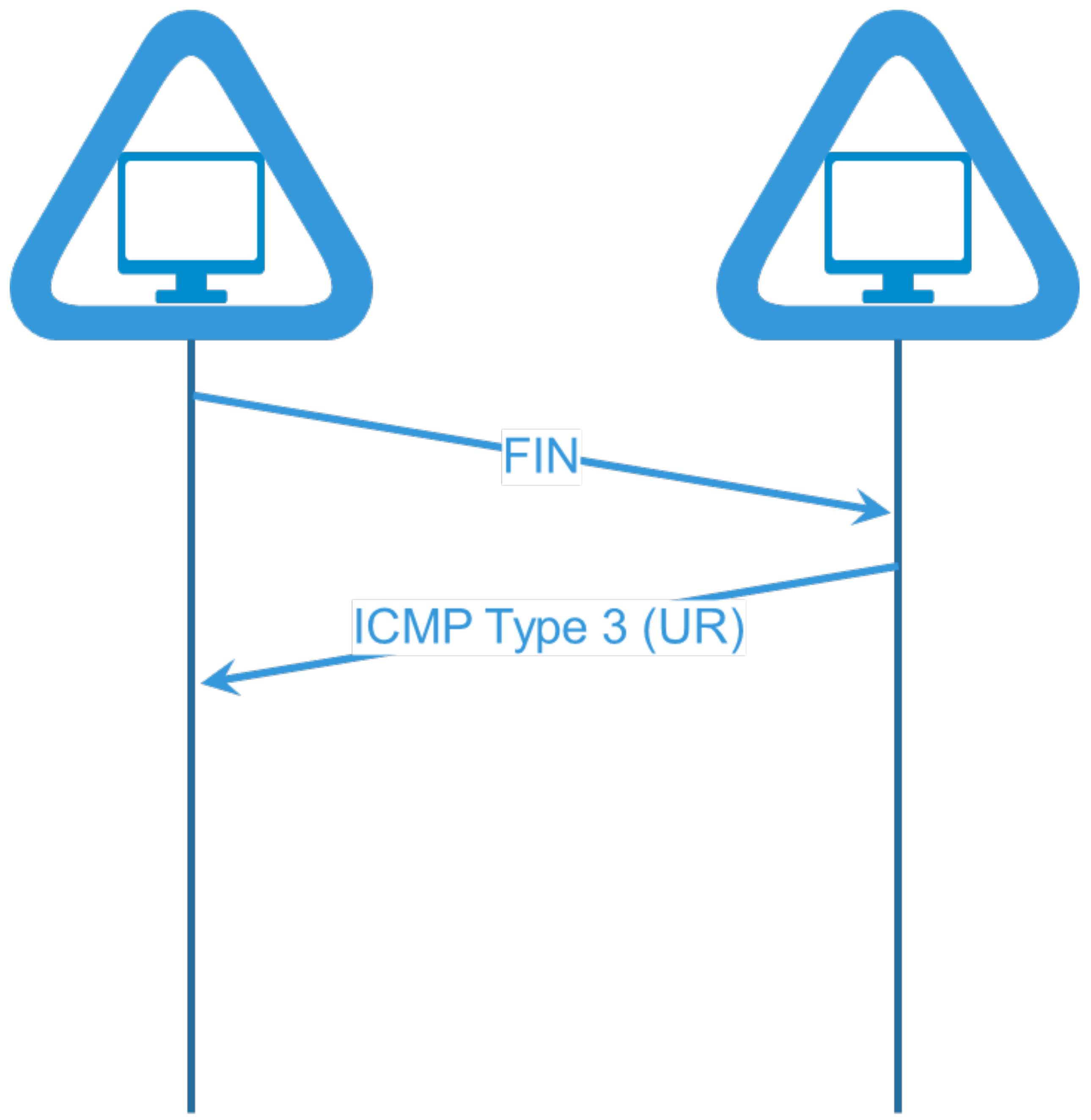


A SYN port scan is usually the most up-front version.

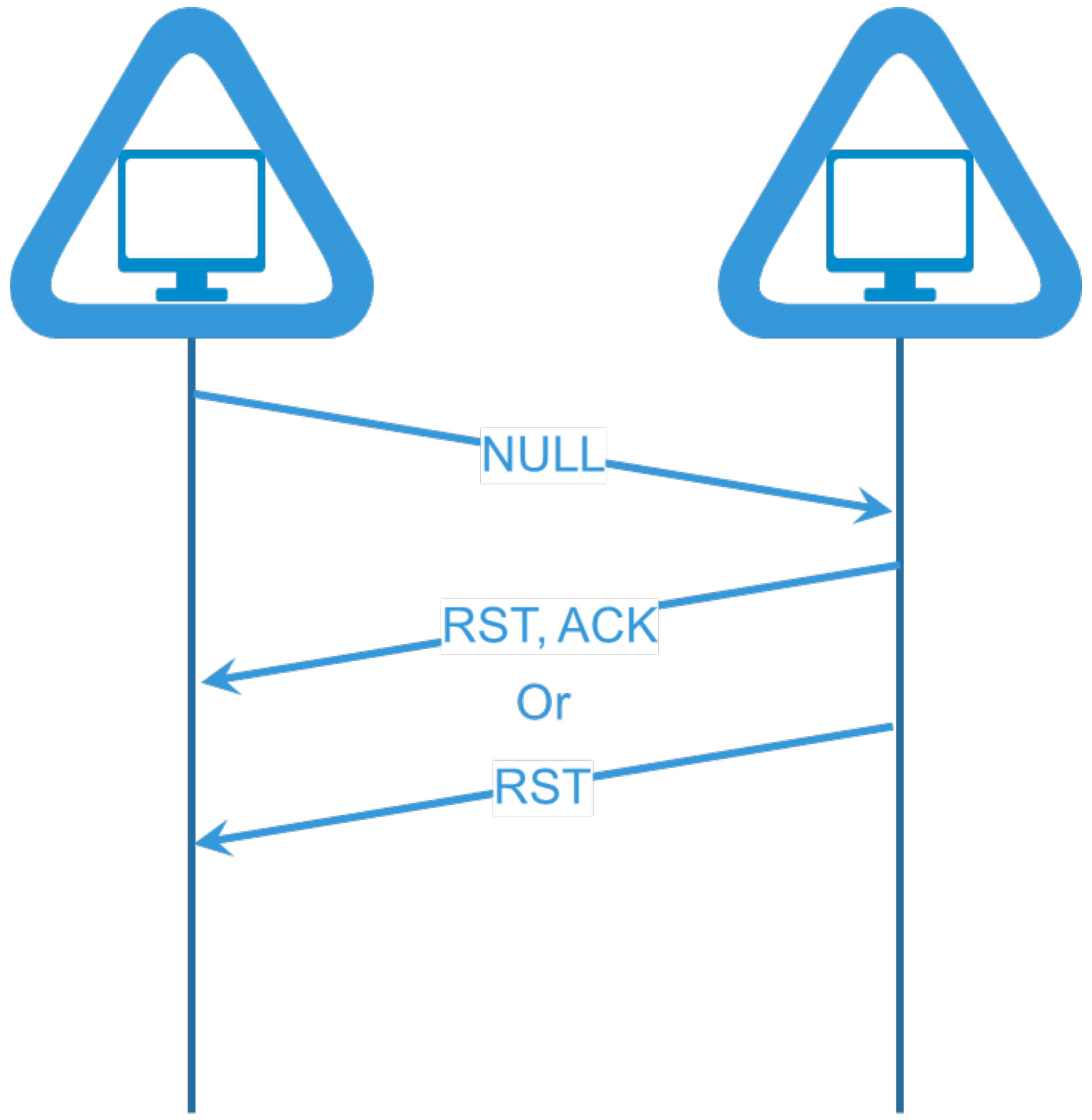
FIN Scan



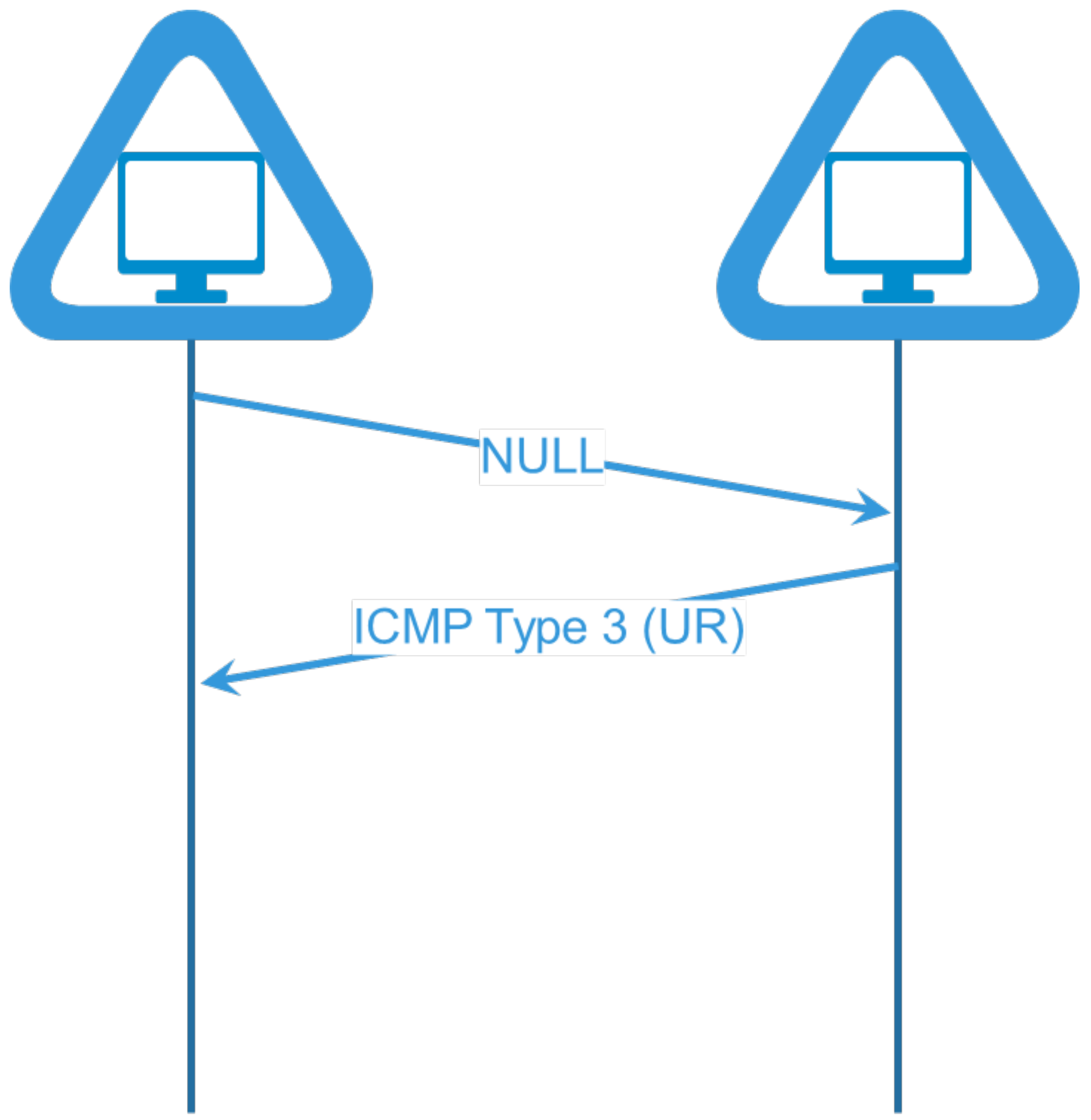
FIN Scan



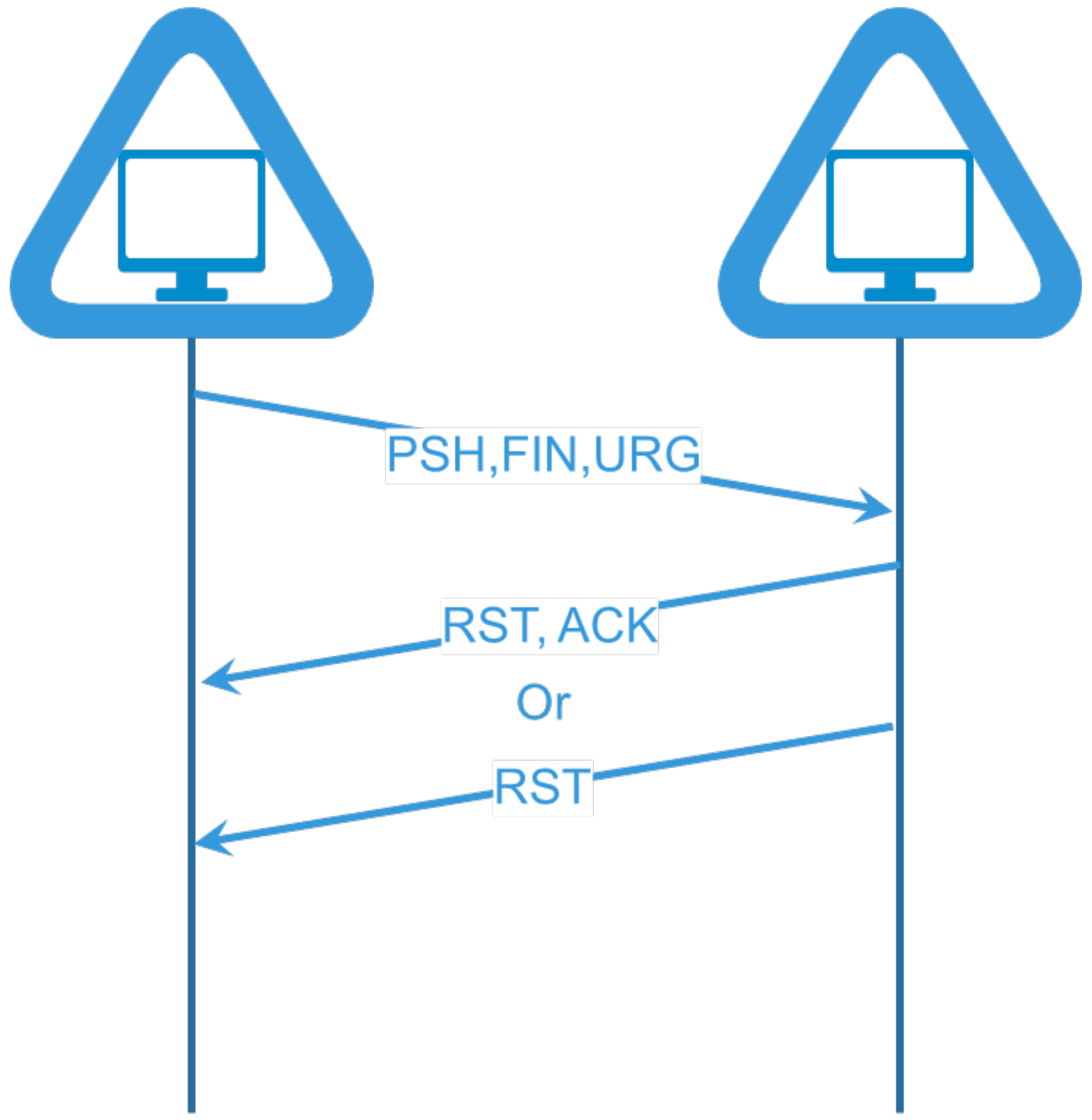
NULL Scan



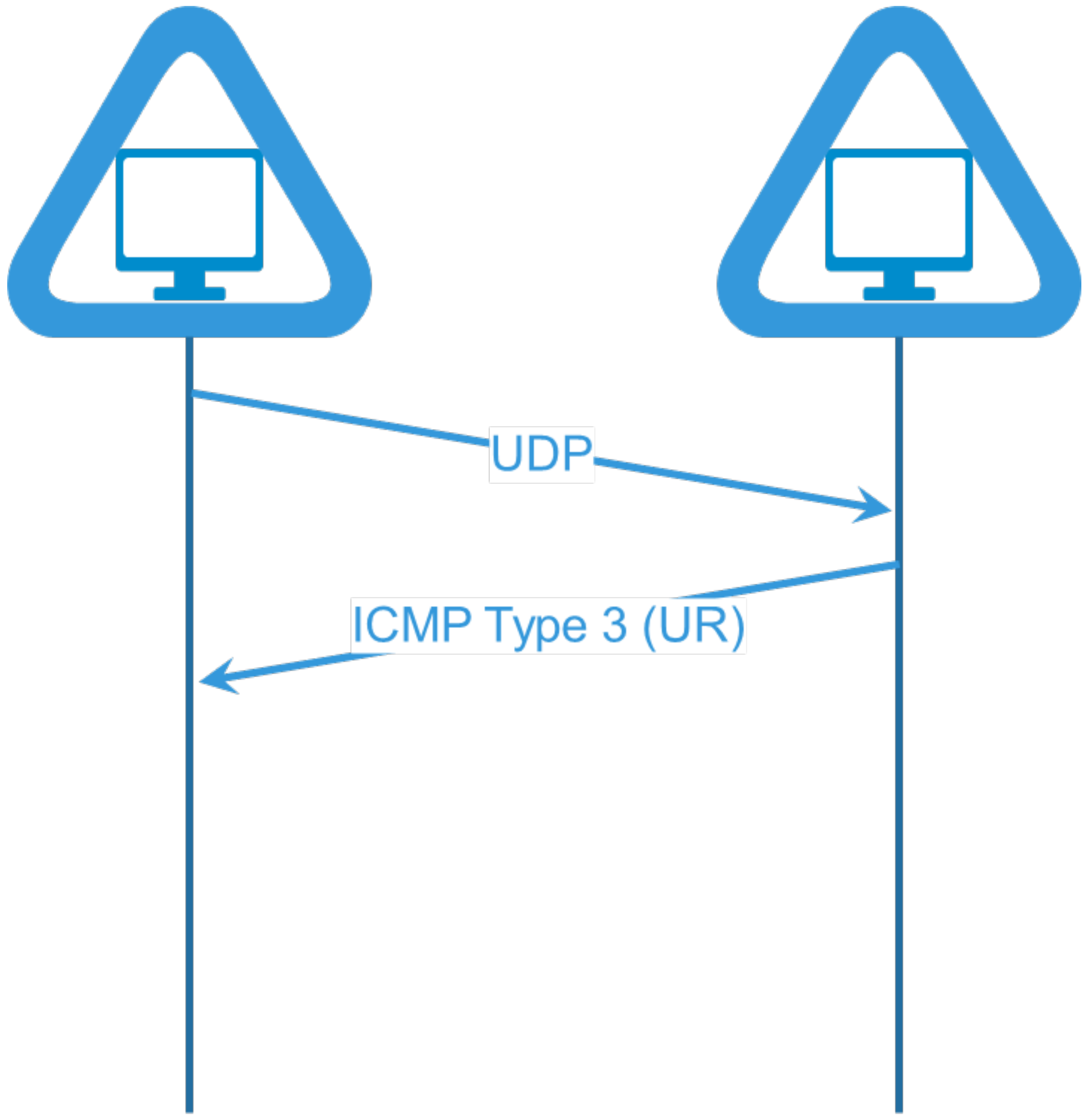
NULL Scan



xMAS Scan



UDP Scan



Demo

Port Scanning Techniques

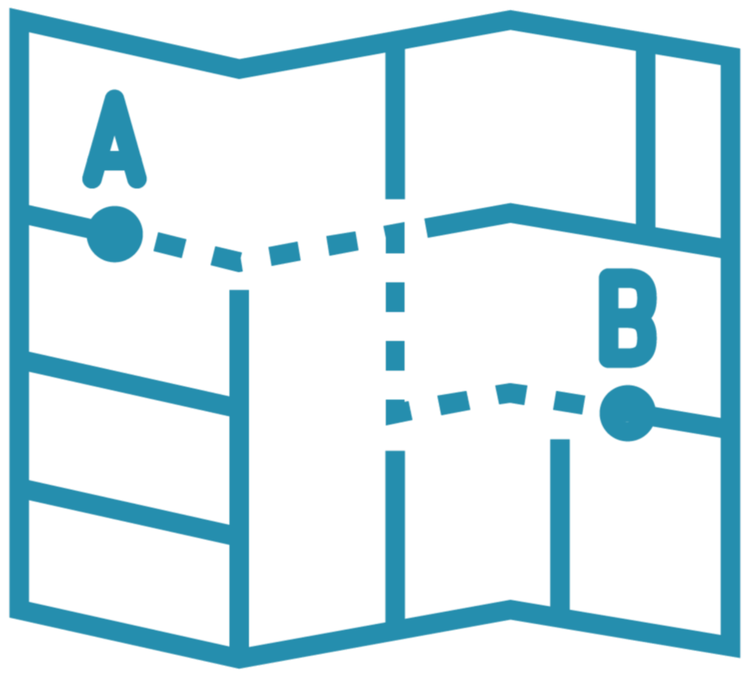
Traceroute

Scapy can do traceroutes

Essential tool for IT workers

Used to map out a network's structure

Traceroute



Two main ways to use

- ICMP alone
- Combination of UDP and ICMP

Basics are the same no matter the method used

- Series of packets are sent towards a destination
- TTL fields sequenced 1-30
- TTL field limits the number of hops

Traceroute

Initial messages will use:

ICMP as echo request

UDP packet

If TTL is exceeded

An ICMP response should be sent back

With both responses combined: the path can be determined

TCP

Can be used as initial packet

Can be slower

Helps with troubleshooting

Demo

Traceroute Techniques

Summary

- Covering the Basics of Send and Receiving Packets
- Concepts Demonstration - Sending and Receiving Packets
- Discussing How to Perform Port Scanning
- Concepts Demonstration - Port Scanning
- Reviewing the Available Traceroute Options
- Concepts Demonstration - Traceroute