University of Glasgow

Connection Establishment in a Fragmented Network

Networked Systems (H) – Lecture 2

2019-2020

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow | This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/ or send a letter to Creative Commons, PO Box 1866,

Lecture Outline

- Review of TCP
- Client-server connection establishment
- Peer-to-peer connections and NAT traversal

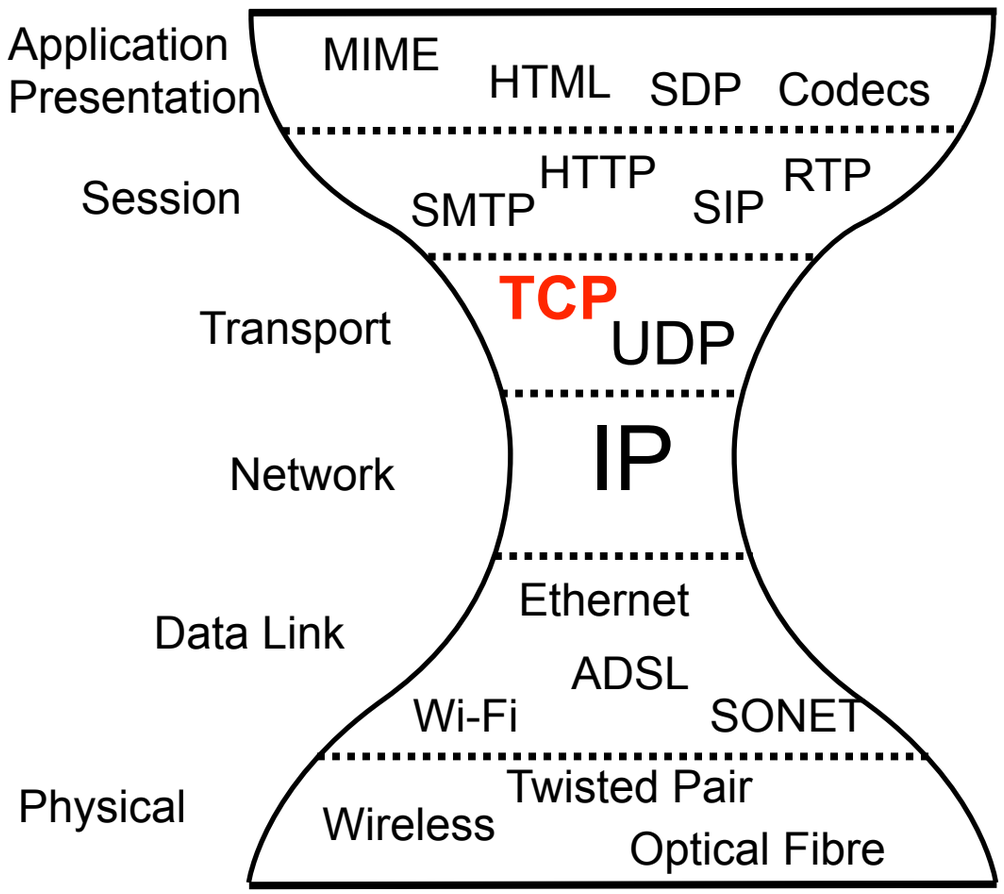
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Review of TCP

- TCP Segment Format
- TCP Service Model
- TCP Sockets Programming

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

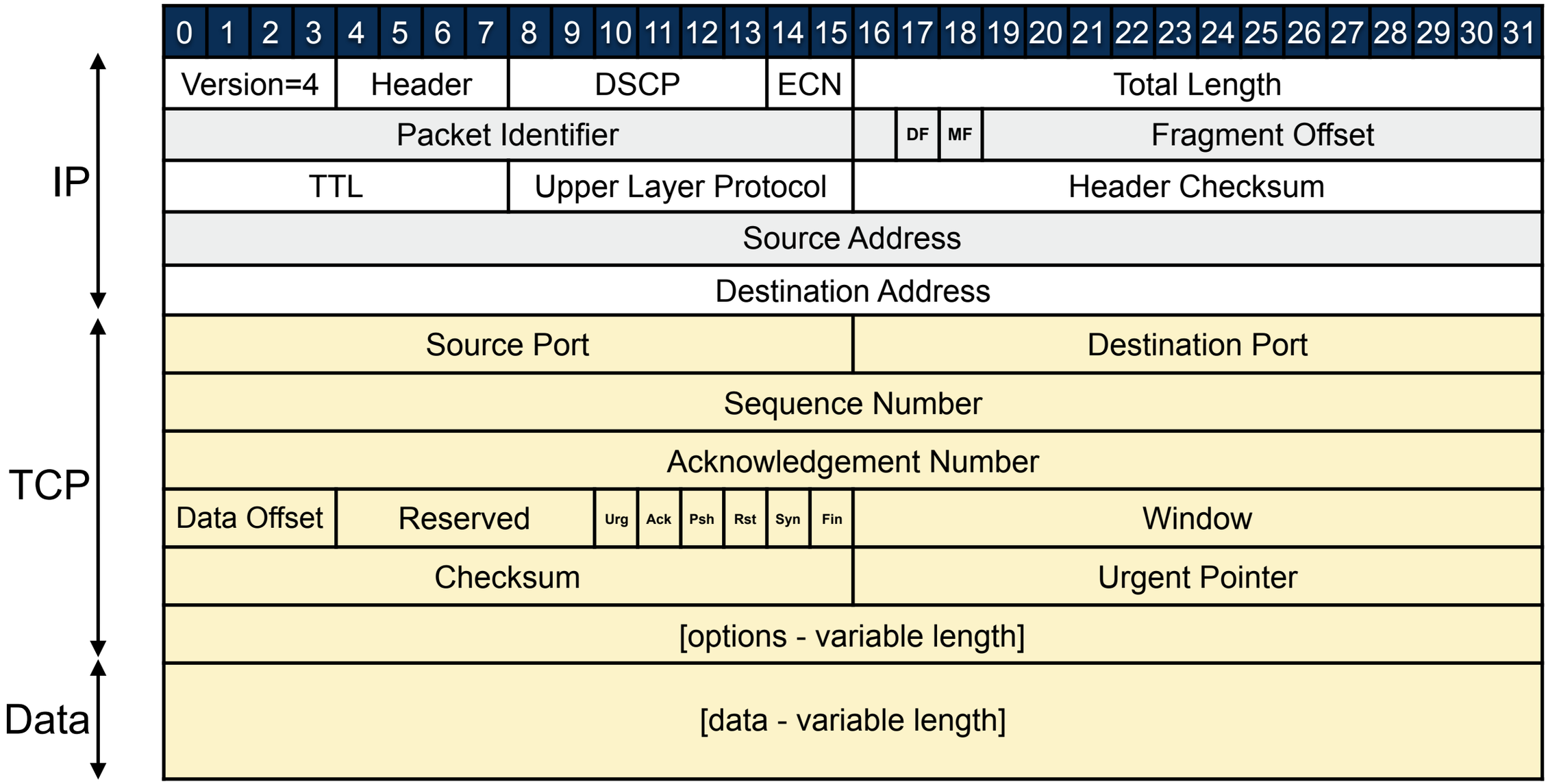
TCP: Reliable Data Transport for the Internet



- TCP transport layer
  - Provides a reliable, ordered, byte stream service over the unreliable IP network
  - Provides congestion control, to adapt to network capacity
  - Most widely used Internet transport protocol
- Delivered within IP
  - TCP *segments* carried as data in IP packets
  - IP *packets* carried as data in link layer frames
  - Link layer *frames* delivered over physical layer

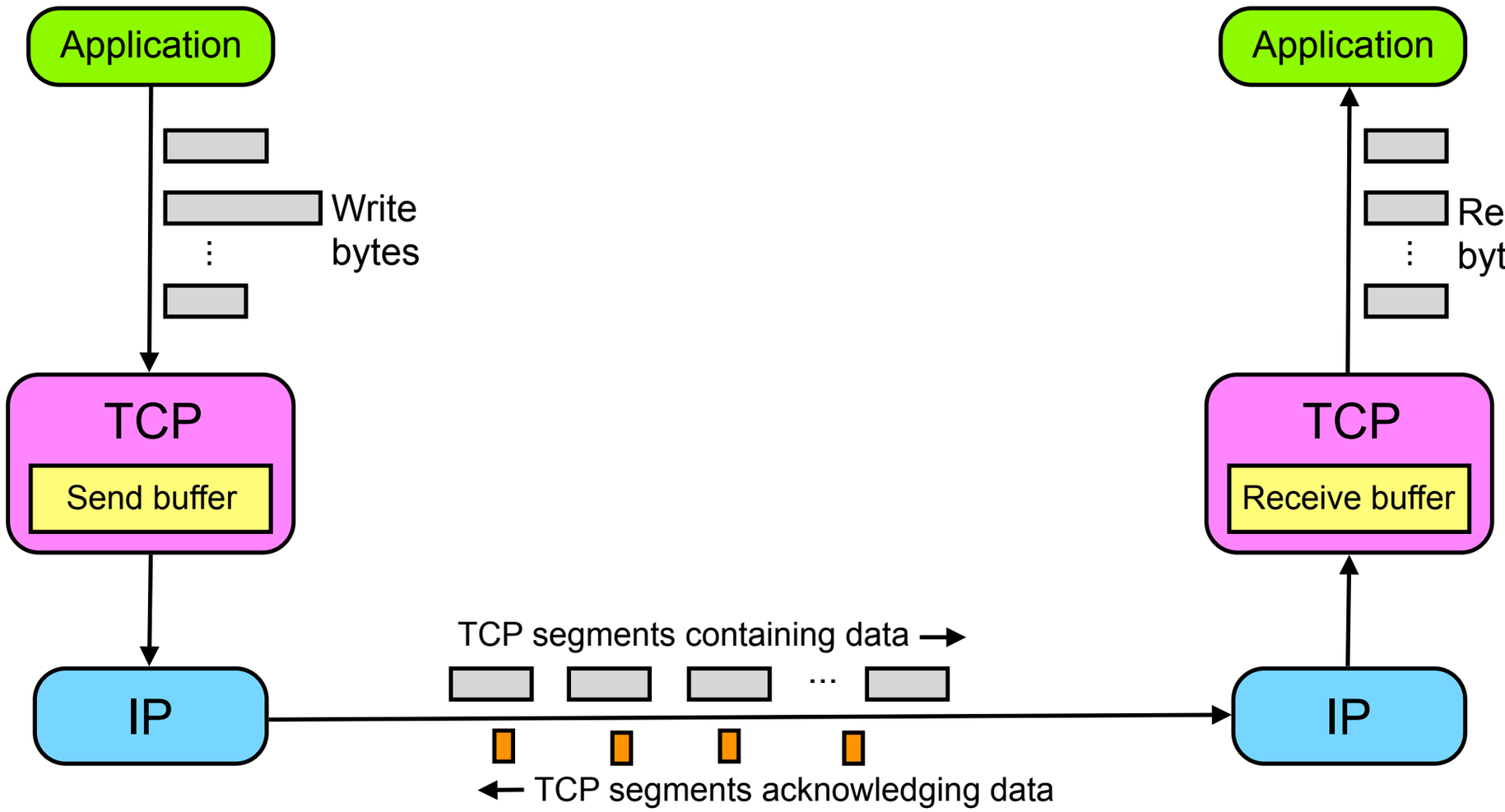
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Segment Format



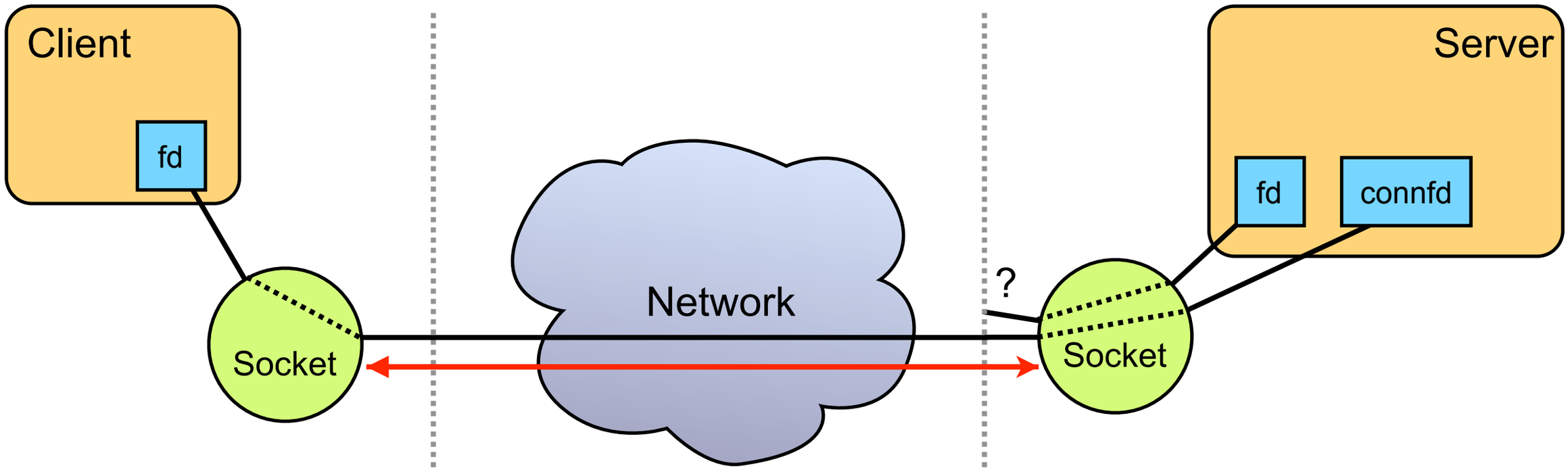
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Service Model



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Programming Model




```
int fd = socket(...)

connect(fd, ..., ...)

send(fd, data, datalen, flags)

recv(fd, buffer, buflen, flags)

close(fd)
```

```
int fd = socket(...)
bind(fd, ..., ...)
listen(fd, ...)
connfd = accept(fd, ...)

recv(connfd, buffer, buflen, flags)

send(connfd, data, datalen, flags)

close(connfd)
```

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

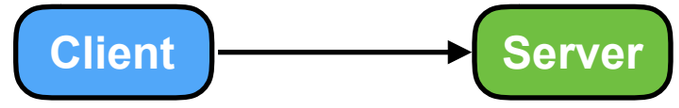
Client-server Connections

- Connection Establishment and the Impact of the RTT on performance
- Performance of secure connections
- Performance of dual stack hosts
- Discussion

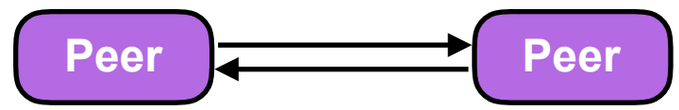
CC BY ND

TCP Connections

- Typical TCP-based application uses client-server model
  - Server listens for connections on a port
  - Clients connects to the server
  - Client sends requests; server responds



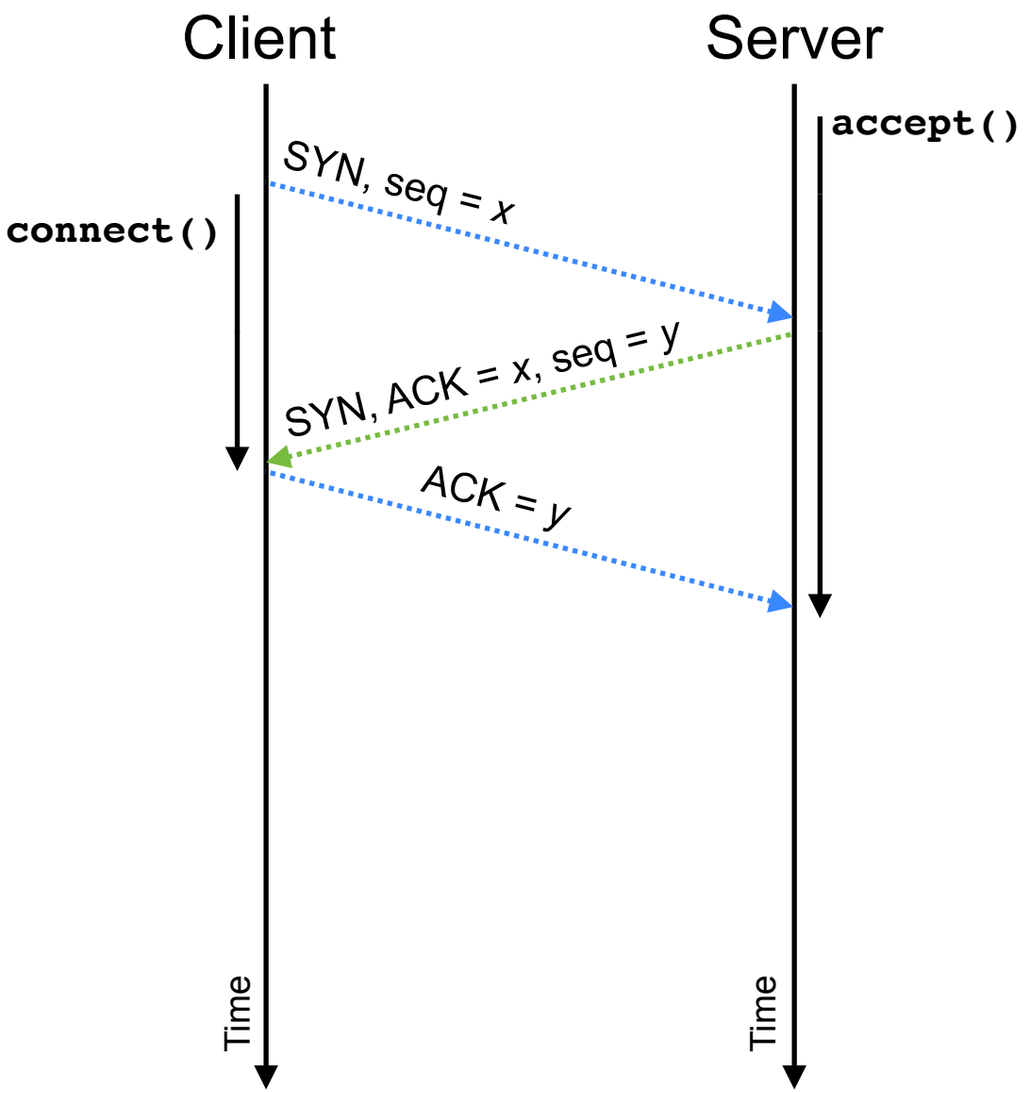
- Can be used in a peer-to-peer manner
  - The two peers simultaneously **`connect()`** to each other, then send and receiver data
  - Uncommon, but possible – requires both peers to have known and accessible IP addresses, and to **`bind()`** to known ports



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Client-Server Connection Establishment

- The **`connect()`** call triggers the three-way handshake:
  - **Client → Server:**
    - SYN (“synchronise”) bit set in TCP header
    - Initial sequence number chosen at random
  - **Server → Client:**
    - SYN
    - ACK for client’s initial sequence number
    - Initial sequence number chosen at random
  - **Client → Server:**
    - ACK for server’s initial sequence number
- When handshake completes, connection is established

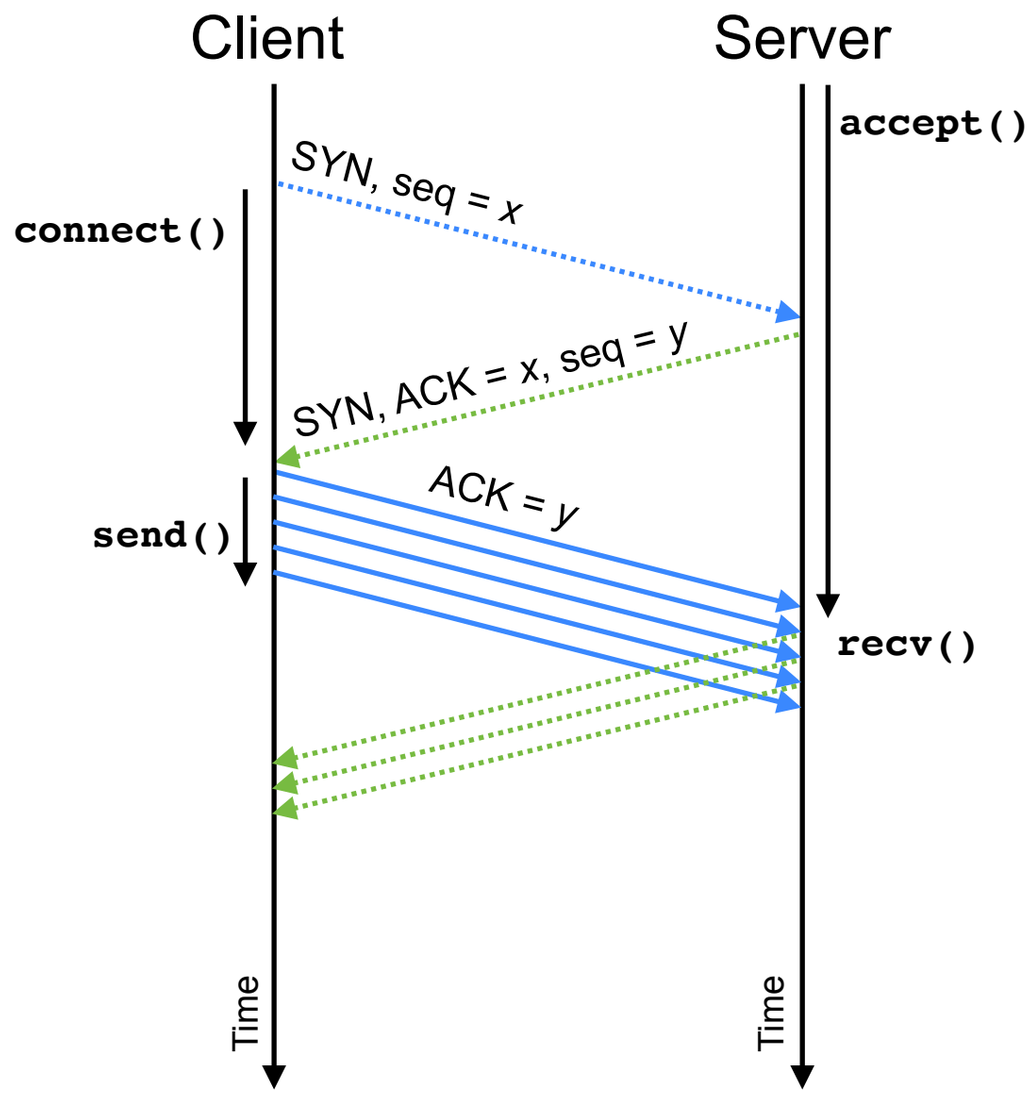


Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Response Time

- Calls **`send()`**/**`recv()`** transmit data

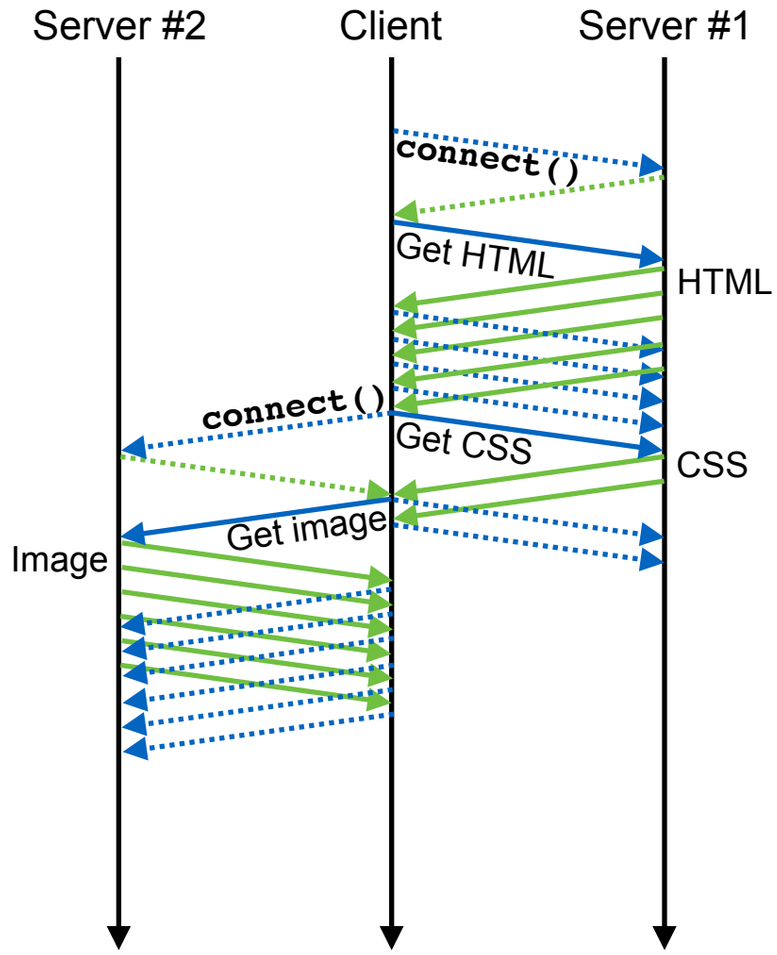
- For short flows, initial three-way handshake can take a significant fraction of connection lifetime



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of Latency on Performance (1/3)

- Assume a very simple web page:
  - 1x HTML file, 1x CSS file, and 1x image
  - HTML and CSS files hosted on one server, image on a different server
  - Everything retrieved via HTTP over TCP/IP
- How long does it take to retrieve that page?
  - 1x RTT handshake to connect to server 1
  - 1x RTT + serialisation time to fetch HTML file
  - Longest of:
    - 1x RTT + serialisation time to fetch CSS file
  - and
    - 1x RTT handshake to connect to server 2
    - 1x RTT + serialisation time to fetch image



Serialisation time is time taken to transmit data:
- 1Mbyte file = 1,048,576 bytes = 8,388,608 bits
- 2Mbps link = 2,097,152 bits per second
- 8,388,608 / 2,097,152 = 4 seconds to transmit

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of Latency on Performance (2/3)

- Performance depends on RTT and available bandwidth
- What is a typical RTT?
  - Depends on distance
  - Depends on network congestion

- What is typical available bandwidth?
  - ADSL2+ → 25Mbps
  - VDSL → 50Mbps
  - 4G wireless → 15-30Mbps

    https://www.opensignal.com/reports/2019/04/uk/mobile-network-experience

    (assuming uncongested links)

Destination	Min RTT (ms)	Avg RTT (ms)	Max RTT (ms)
WiFi Router	1.1	32.6	431.7
BBC	33.4	74.6	432.3
Google	33.9	72.5	264.3
Columbia University, New York	103.8	153.3	344.1
University of California, Los Angeles	165.1	221.2	529.8
University of Technology Sydney, Australia	307.0	381.2	685.9

RTT measurements (ping times) from a residential site in Glasgow to various locations around the world

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of Latency on Performance (3/3)

- Latency hurts performance
  - Connection establishment is slower
  - Retrieving data takes longer
- As RTT increases, benefits of increasing bandwidth reduce
  - For this example, with 300ms RTT, increasing bandwidth from 15Mbps to 1Gbps gives only 22% reduction in page download time
  - Connection setup time – the 3-way handshake of TCP – dominates in many scenarios

RTT \ BW	1 Mbps	15 Mbps	25 Mbps	50 Mbps	100 Mbps	1 Gbps
1ms	43.9	2.9	1.7	0.9	0.4	0.04
50ms	44.1	3.1	1.9	1.1	0.6	0.24
100ms	44.3	3.3	2.1	1.3	0.8	0.44
150ms	44.5	3.5	2.4	1.5	1.0	0.64
300ms	45.1	4.1	2.9	2.1	1.6	1.24

→10x

→1.8x

→1.3x

Download times, in seconds, for the simple web page described earlier, for a range of RTTs and bandwidths

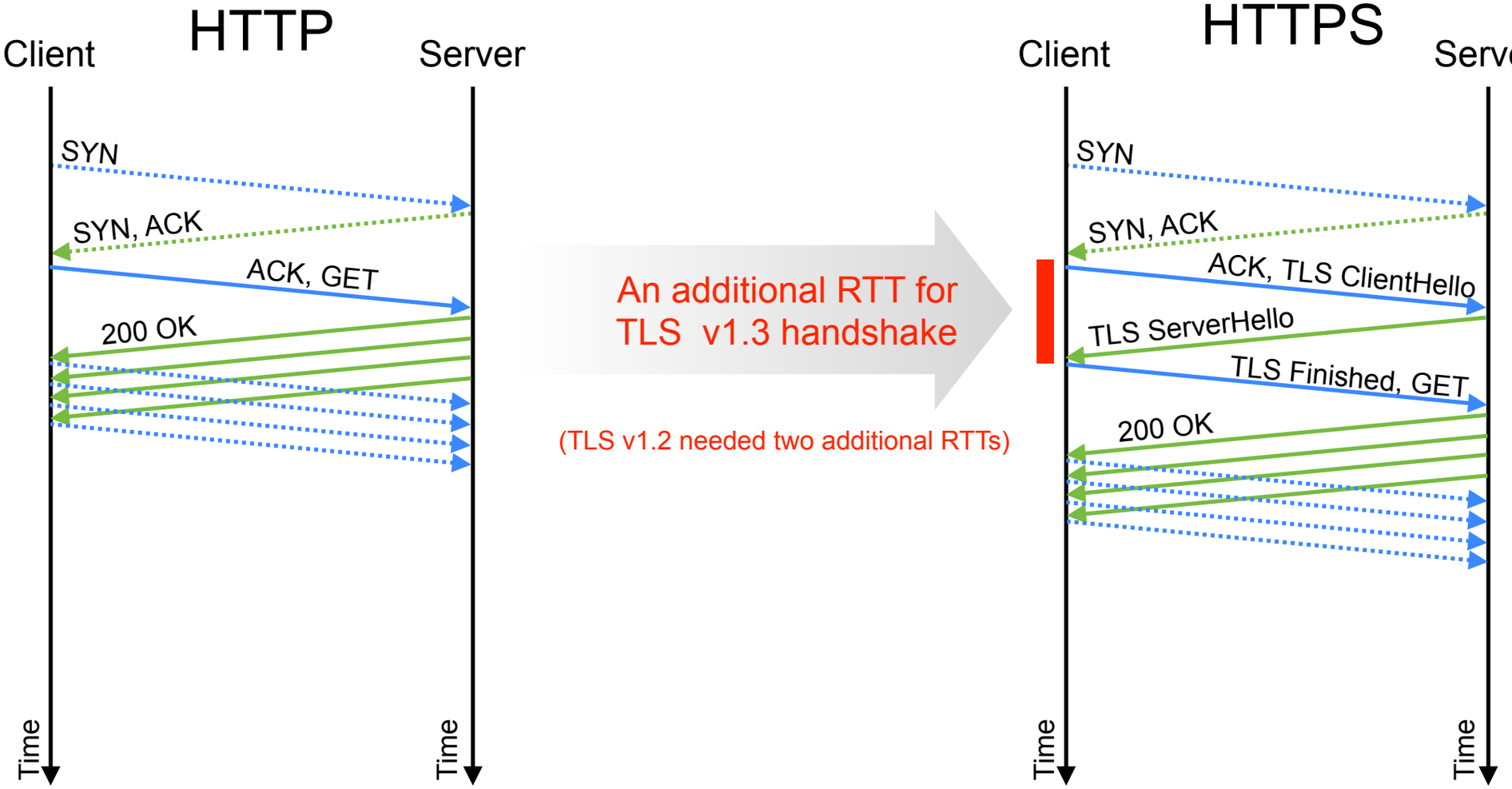
HTML = 500 kbytes, CSS = 100 kbytes, IMG = 5 Mbytes
Assuming no impact due to congestion control

- *Is it still worth paying extra for a faster Internet connection?*

CC BY ND

Impact of Transport Layer Security (1/2)

- The protocol running over TCP can also impact performance
  - HTTP sends and retrieves data immediately the TCP connection is open
  - HTTPS opens a TCP connection, negotiates security parameters using TLS within that TCP connection, then starts to send and receive encrypted data – what impact does this have?



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of Transport Layer Security (2/2)

- TLS v1.3 introduces two RTTs extra latency in this example
- Further causes impact of RTT and connection establishment to dominate performance

	1 Mbps	15 Mbps	25 Mbps	50 Mbps	100 Mbps	1 Gbps
1ms	43.9	2.9	1.7	0.9	0.4	0.04
50ms	44.2	3.2	2.1	1.2	0.7	0.34
100ms	44.5	3.5	2.4	1.5	1.0	0.64
150ms	44.8	3.8	2.7	1.8	1.3	0.94
300ms	45.7	4.7	3.6	2.7	2.2	1.84

→10x

→1.5x

→1.2x

Download times, in seconds, for the simple web page described earlier, for a range of RTTs and bandwidths

HTML = 500 kbytes, CSS = 100 kbytes, IMG = 5 Mbytes
Assuming no impact due to congestion control
With additional RTT to model TLS 1.3 handshake

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of Latency on TCP Performance

- For client-server applications, each request takes 1x RTT plus data serialisation time
  - Unless data is very large, RTT is often most significant performance factor
  - Each TCP connection has 1x RTT connection setup overhead
    - If TLS v1.3 is used inside TCP, an additional 1x RTT overhead
    - If TLS v1.2 is used inside TCP, an additional 2x RTT

- To improve performance in your application:
  - Reduce the number of TCP connections used
  - Limit the number of request-response exchanges between client and server

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Example: HTTP/1.1

```
C: GET /index.html HTTP/1.1
C: Accept-Language: en-gb
C: Accept-Encoding: gzip, deflate
C: Accept: text/html, text/plain
C: User-Agent: Safari/523.12.2
C: Cache-Control: max-age=0
C: Connection: keep-alive
C: Host: www.dcs.gla.ac.uk
C:
```

Retrieving web page using HTTP/1.1

Once secure connection is open, each object requires only a single request-response exchange

Connections are reused for different objects, when possible

```
S: HTTP/1.1 200 OK
S: Date: Wed, 27 Feb 2008 22:44:25 GMT
S: Server: Apache/2.0.46 (Red Hat)
S: Last-Modified: Mon, 17 Nov 2003 08:06:50 GMT
S: Accept-Ranges: bytes
S: Content-Length: 3646
S: Connection: close
S: Content-Type: text/html; charset=UTF-8
S:
S: <HTML>
S: <HEAD>
S: <TITLE>Computing Science, University of Glasgow</TITLE>
S: ...remainder of page elided...
```

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Example: SMTP

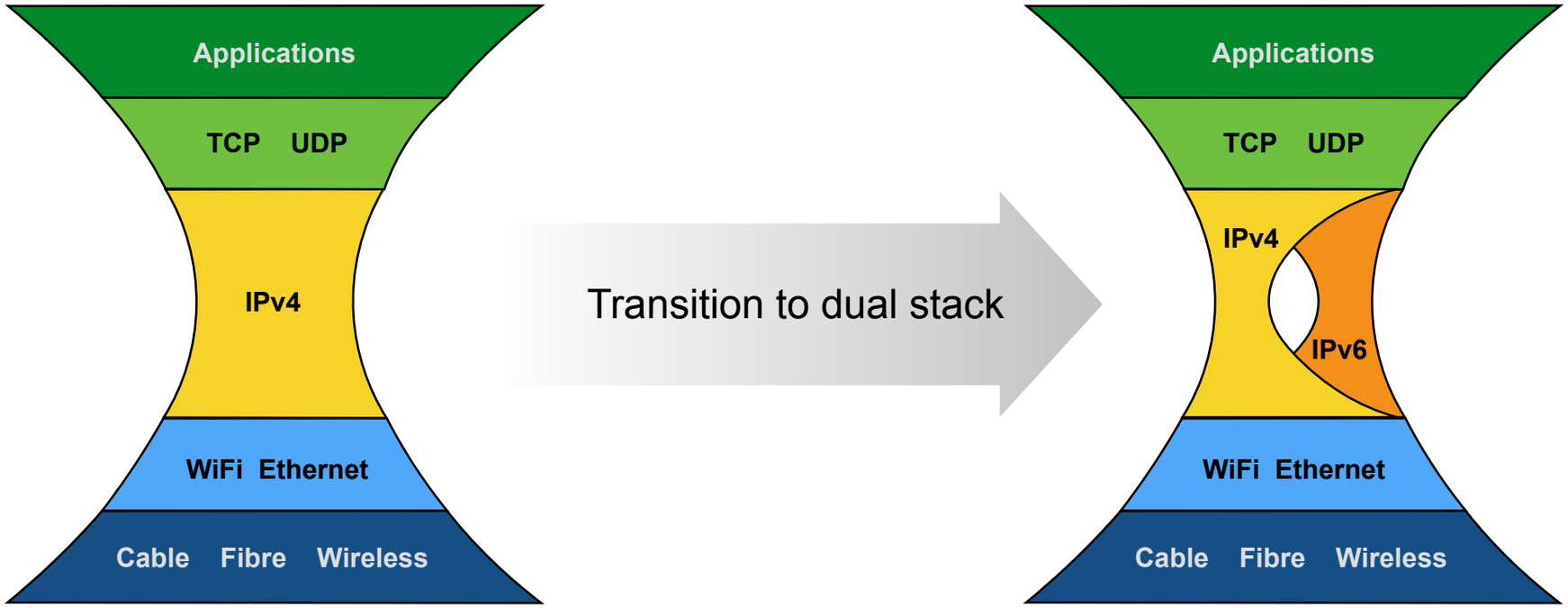
```
S: 220 mr1.dcs.gla.ac.uk ESMTP Exim 4.42 Wed, 27 Feb 2008 10:31:18 +0000
C: HELO bo720-1-01.dcs.gla.ac.uk
S: 250 mr1.dcs.gla.ac.uk Hello bo720-1-01.dcs.gla.ac.uk [130.209.250.151]
C: MAIL FROM:csp@dcs.gla.ac.uk
S: 250 OK
C: RCPT TO:csp@csperkins.org
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
C: From: Colin Perkins <csp@dcs.gla.ac.uk>
C: To:   Colin Perkins <csp@csperkins.org>
C: Date: Wed 27 Feb 2008 10:32:45
C: Subject: Test
C:
C: This is a test
C: .
S: 250 OK id=1JUJa1-00073j-22
C: QUIT
S: 221 mr1.dcs.gla.ac.uk closing connection
```

Sending email using SMTP

Line-by-line request and response:
→ many unnecessary round trips to server
→ compare to HTTP/1.1, which bundles all parameters into a single request
→ poor performance

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Impact of IPv6 and Dual Stack Deployments



- IPv4 → IPv6 transition means we have **two** Internets
  - Some hosts only connect via IPv4, some only via IPv6, some have both types of address
  - Some links carry only IPv4 traffic, some only IPv6 traffic, some both types of traffic
  - Some firewalls allow IPv4 but block IPv6, some allow IPv6 but block IPv4, some allow or block both types of traffic
- IPv6 network is **not** a subset of the IPv4 network; it's separate, but overlaps in places

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Happy Eyeballs

- How to connect to a host with more than one IP address?

Internet Engineering Task Force (IETF) D. Schinazi
Request for Comments: 8305 T. Pauly
Obsoletes: 6555 Apple Inc.
Category: Standards Track December 2017
ISSN: 2070-1721

Happy Eyeballs Version 2: Better Connectivity Using Concurrency

Abstract

Many communication protocols operating over the modern Internet use hostnames. These often resolve to multiple IP addresses, each of which may have different performance and connectivity characteristics. Since specific addresses or address families (IPv4 or IPv6) may be blocked, broken, or sub-optimal on a network, clients that attempt multiple connections in parallel have a chance of establishing a connection more quickly. This document specifies requirements for algorithms that reduce this user-visible delay and provides an example algorithm, referred to as "Happy Eyeballs". This document obsoletes the original algorithm description in RFC 6555.

Status of This Memo

This is an Internet Standards Track document.

https://datatracker.ietf.org/doc/rfc8305/

```
[stlinux02] > ./dnslookup netflix.com
netflix.com IPv6 2a01:578:3::364c:3c27
netflix.com IPv6 2a01:578:3::3411:1b81
netflix.com IPv6 2a01:578:3::22fb:b596
netflix.com IPv6 2a01:578:3::36c2:57d0
netflix.com IPv6 2a01:578:3::36e5:444d
netflix.com IPv6 2a01:578:3::369a:5167
netflix.com IPv6 2a01:578:3::364d:1824
netflix.com IPv6 2a01:578:3::34d1:e0a1
netflix.com IPv4 54.171.116.69
netflix.com IPv4 52.211.94.145
netflix.com IPv4 52.31.182.100
netflix.com IPv4 34.250.41.147
netflix.com IPv4 52.211.42.108
netflix.com IPv4 54.194.155.146
netflix.com IPv4 52.19.40.147
netflix.com IPv4 52.210.7.69
[stlinux02] >
```

- Perform DNS lookups for IPv4 and IPv6 in parallel; start with whichever completes first
- Call **`connect()`** for that address; if not connected within ~100ms, start **`connect()`** to next address on the list in parallel, alternating between IPv4 and IPv6
- Use first **`connect()`** that succeeds, drop other successful connections
- Balances speed to connect vs. network overload by trying all at once in parallel

cc BY ND

Discussion: Connection Latency

- Two components to network performance
  - Bandwidth
  - RTT
- In many cases, RTT can dominate

- How to improve performance?
  - Overlap **`connect()`** calls for hosts with multiple addresses
  - Overlap TCP and TLS handshakes → QUIC transport protocol, Lecture 4
  - Reduce number of TCP connections made
  - Reduce number of requests per connection
  - Reduce RTT: increase speed of light or reduce queuing delay → lectures 5, 6

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

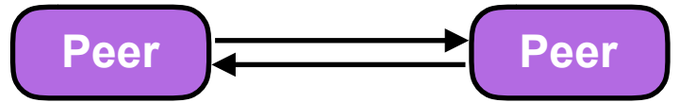
Peer-to-peer Connections

- Peer-to-peer connections and NAT traversal
- Binding discovery and connection probing

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Peer-to-peer Connection Establishment

- TCP *should* be usable in a peer-to-peer manner
  - The two peers simultaneously **`connect()`** to each other, then send and receiver data
  - Uncommon, but possible – requires both peers to have known and accessible IP addresses, and to **`bind()`** to known ports
- UDP *should* be even easier – just send to peer IP address and port

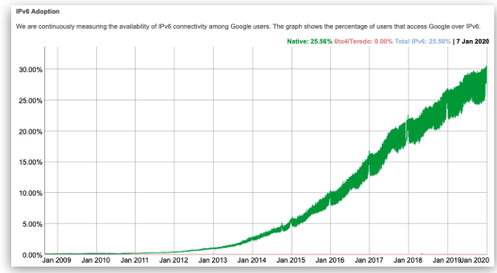


- In practise peer-to-peer connection establishment is difficult, due to **network address translation** (NAT)

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Network Address Translation (NAT)

- IPv4 address space is exhausted
- IPv6 is the long-term solution
- The transition is taking many years

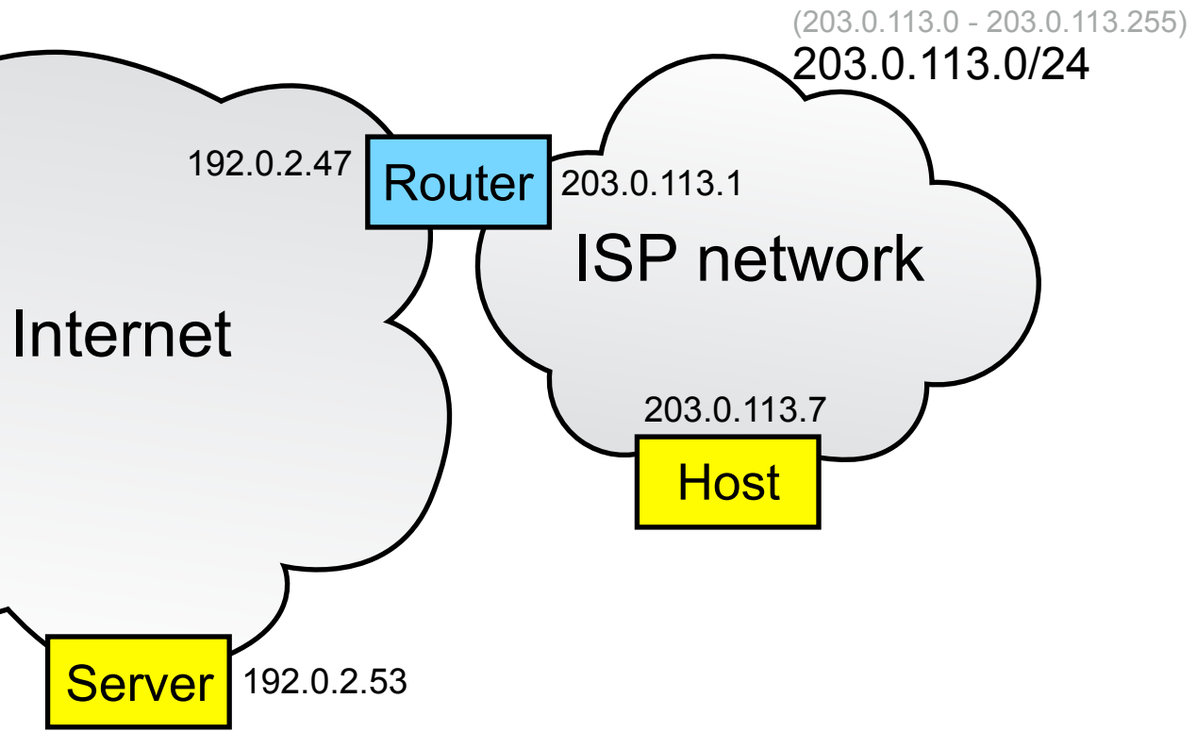


- There is a widely deployed work-around to IPv4 address shortage: NAT (network address translation)

- However, this has serious consequences for the transport layer

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

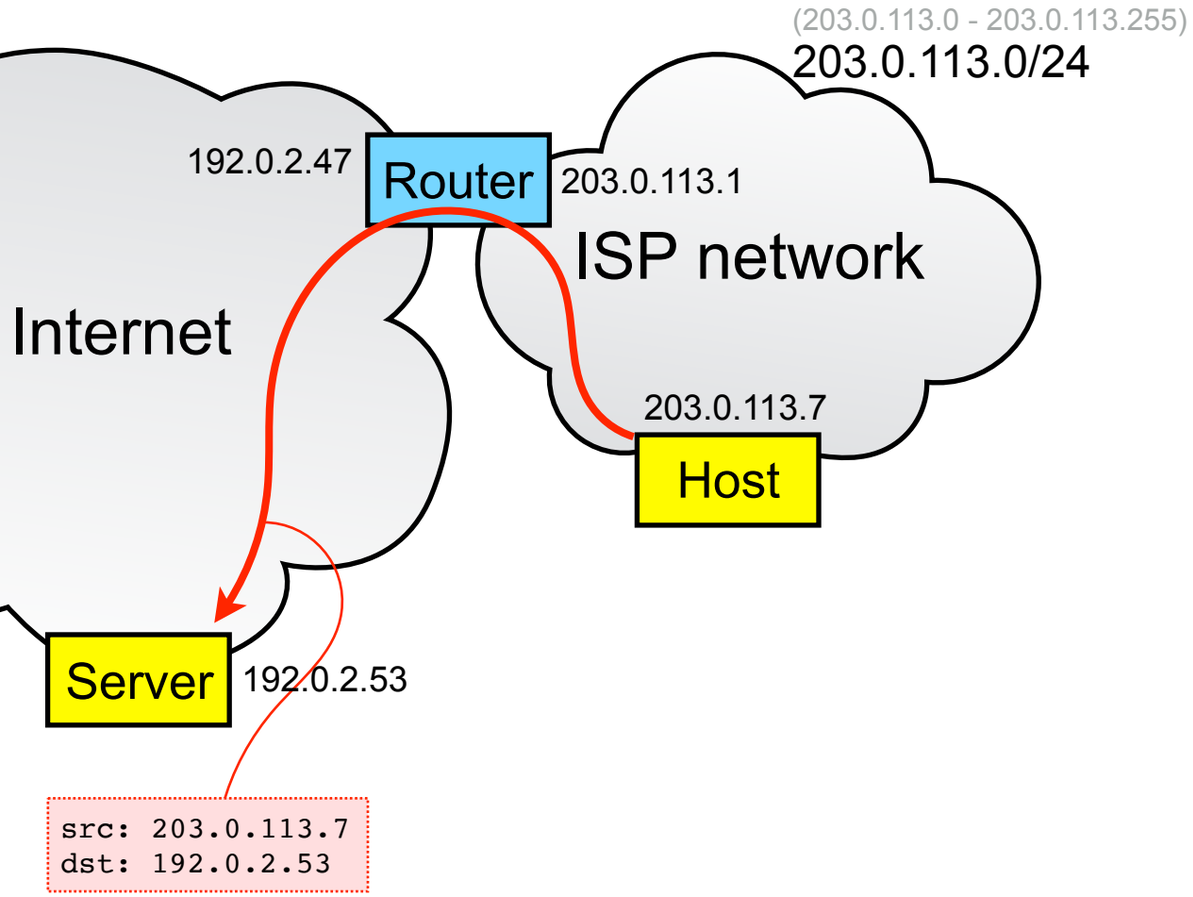
Connecting a Single Host



- An Internet service provider owns an IP address prefix
- They assign a customer a single address for a single host

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Connecting a Single Host

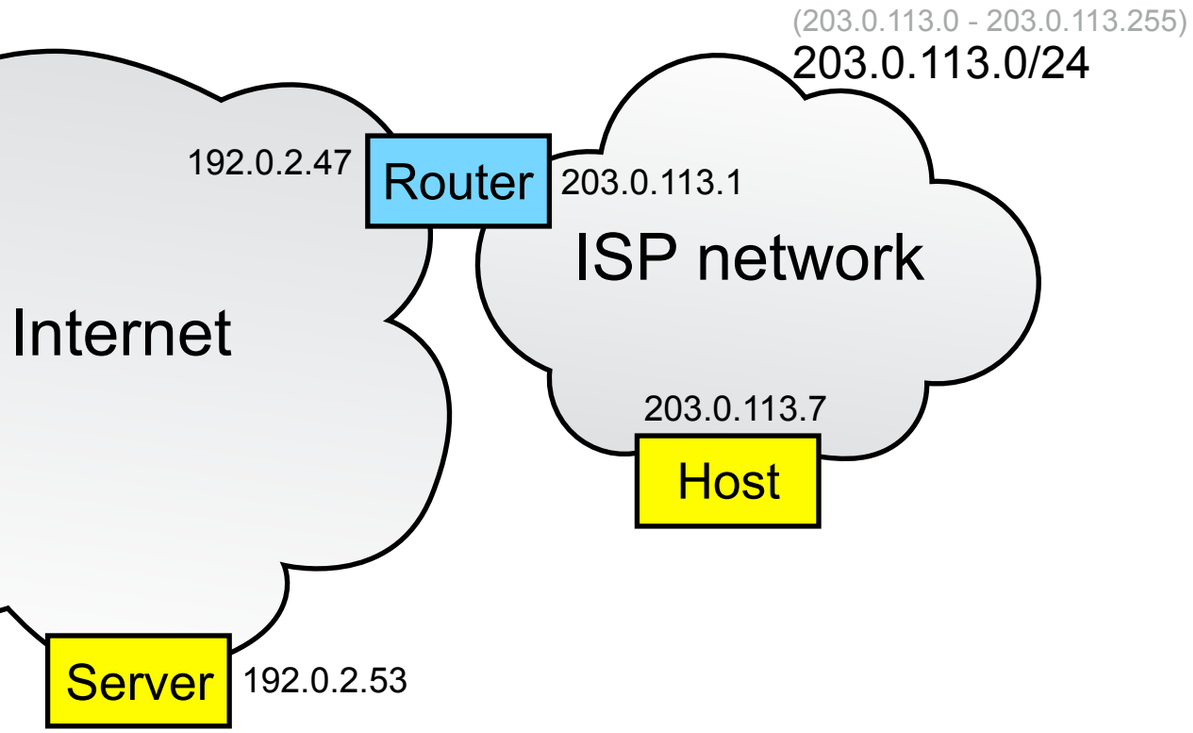


- An Internet service provider owns an IP address prefix
- They assign a customer a single address for a single host

- No address translation

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

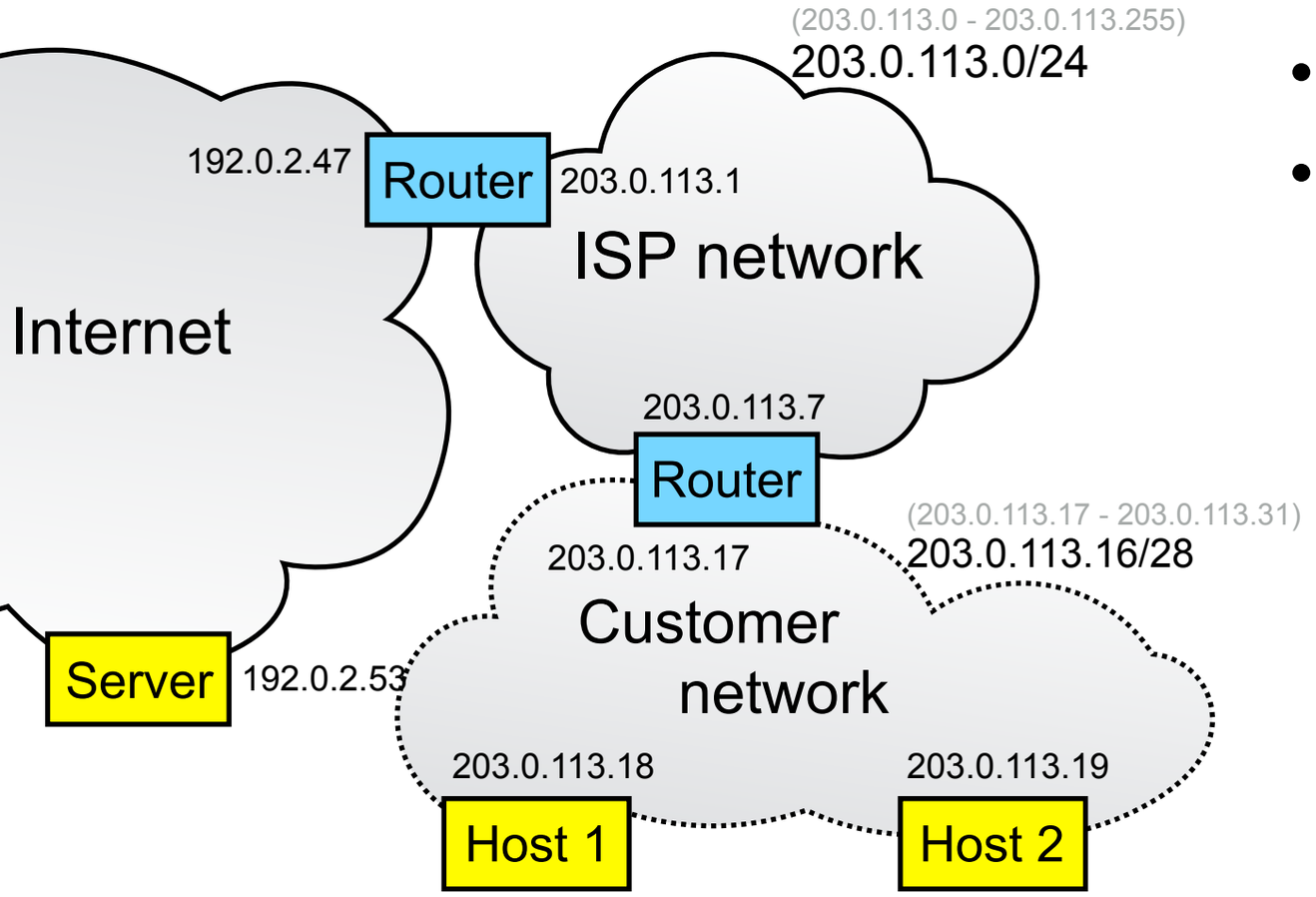
Connecting Multiple Hosts



- The customer buys another host
- How does it connect?

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

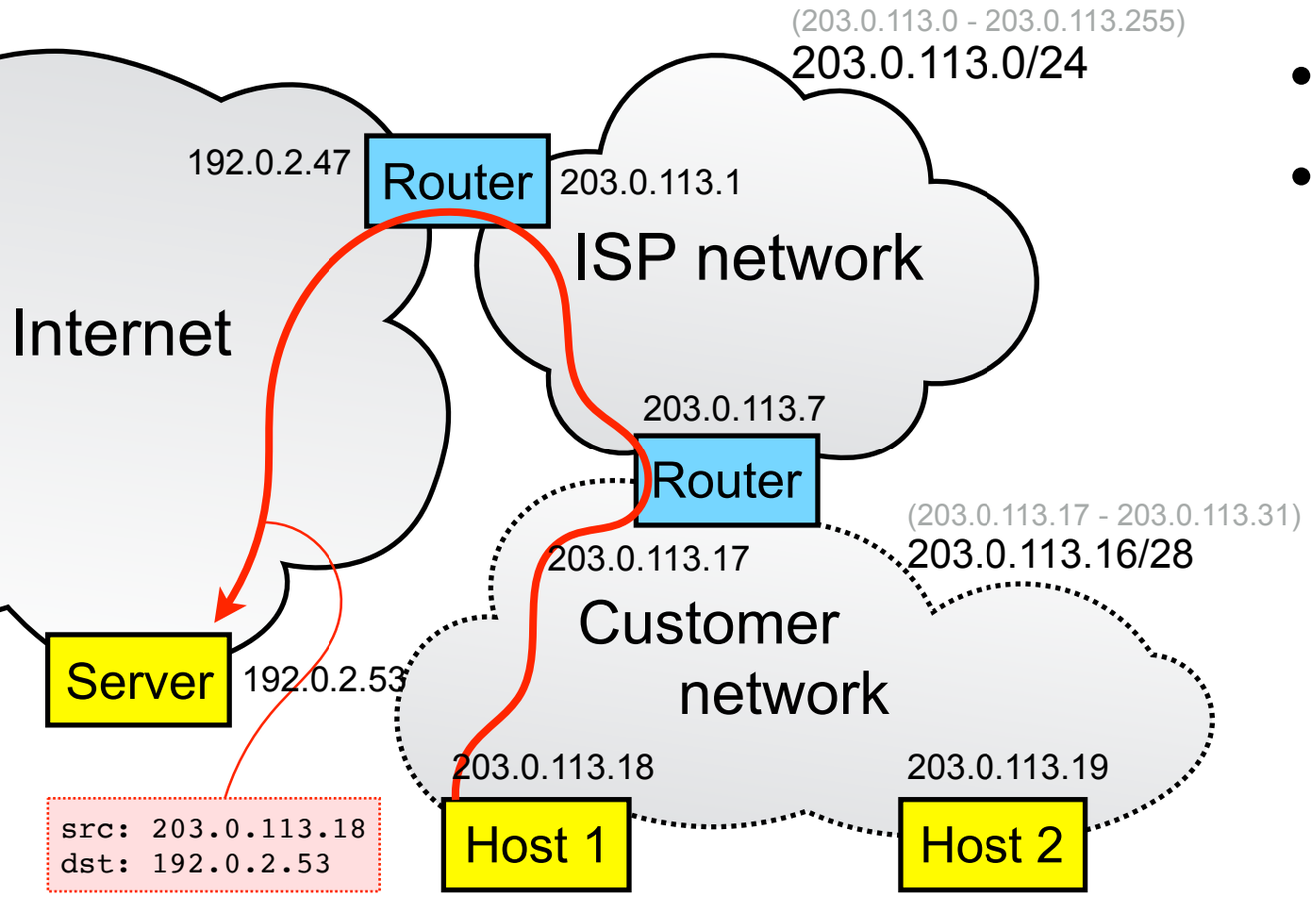
Connecting Multiple Hosts



- The customer buys another host
- How does it connect?
- What's supposed to occur:
  - Customer acquires a router, which gets the customer's previous IP address
  - ISP assigns new range of IP addresses to customer (from the ISP's prefix)
  - Customer gives each host an address from that new range

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

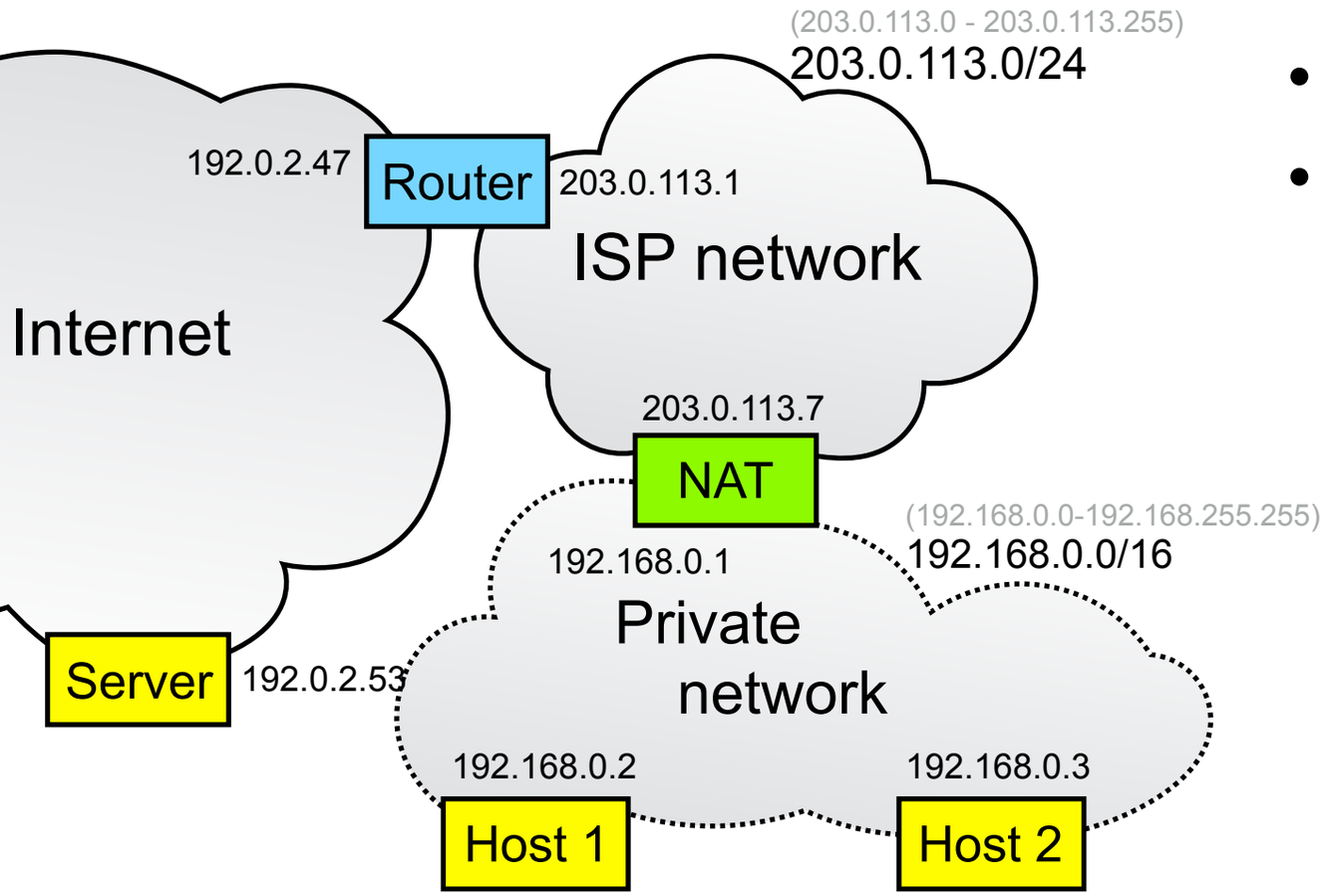
Connecting Multiple Hosts



- The customer buys another host
- How does it connect?
- What's supposed to occur:
  - Customer acquires a router, which gets the customer's previous IP address
  - ISP assigns new range of IP addresses to customer's network (from the ISP's prefix)
  - Customer gives each host an address from that new range
  - No address translation

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

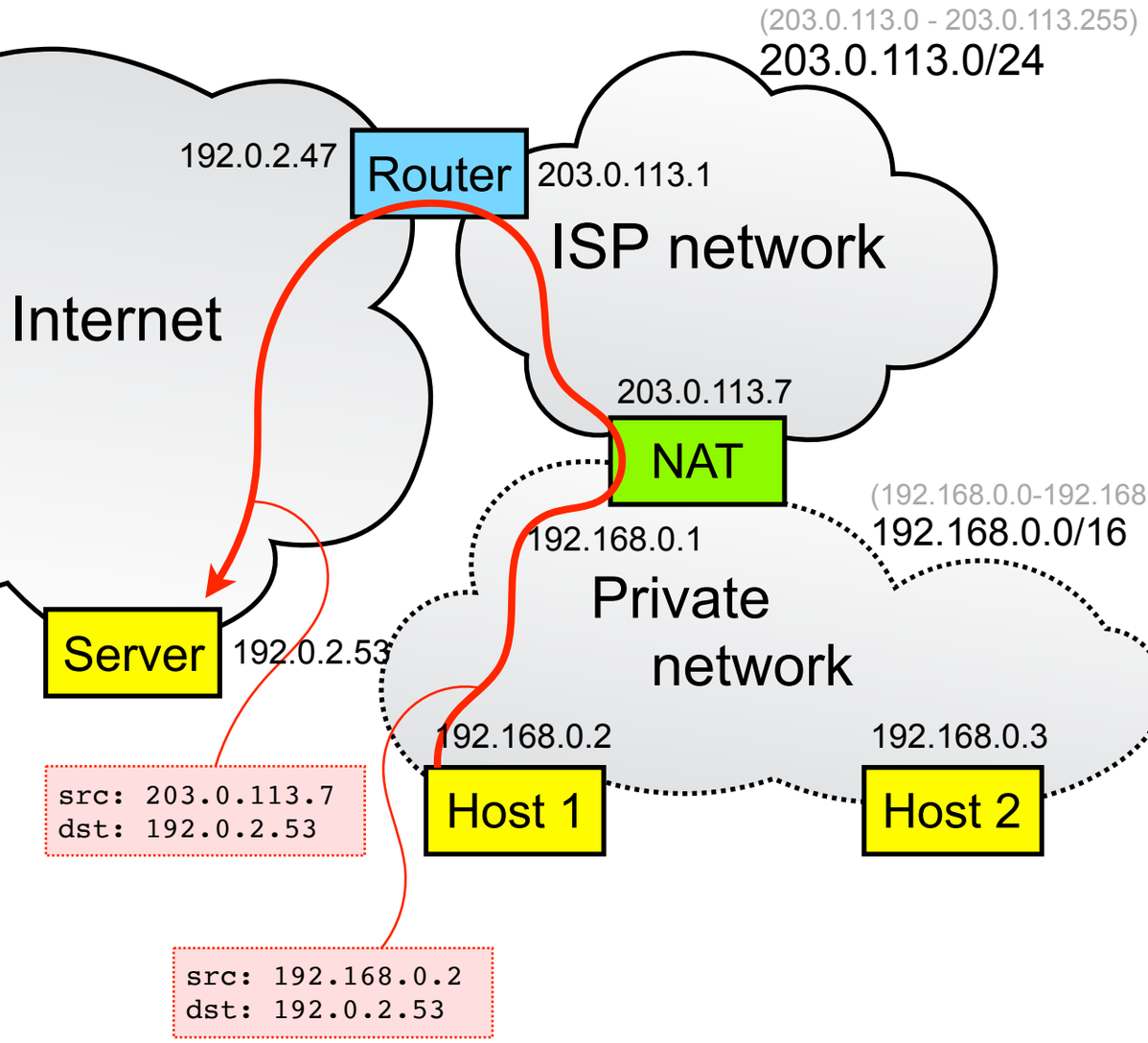
Network Address Translation



- The customer buys another host
- How does it connect?
- What actually happens:
  - Customer acquires a NAT router, which gets the customer's previous IP address
  - Customer gives each host on their network a private address

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

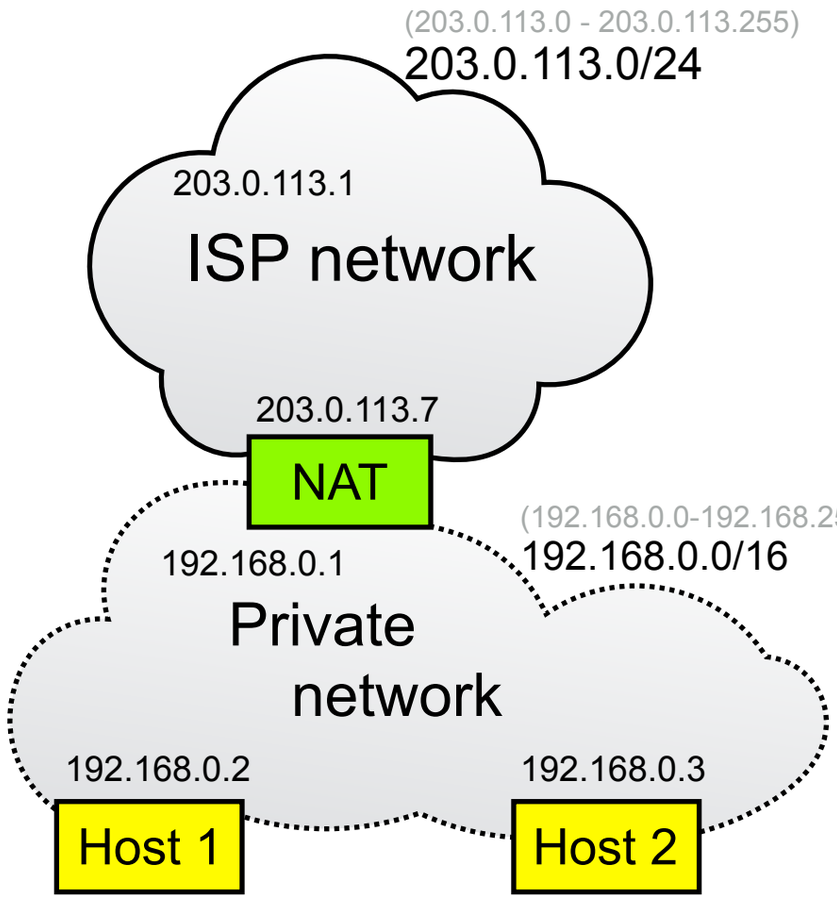
Network Address Translation



- The customer buys another host
- How does it connect?
- What actually happens:
  - Customer acquires a NAT router, which gets the customer's previous IP address
  - Customer gives each host on their network a private address
- NAT performs address translation on packets traversing it:
  - Change source IP address in packet header to match external address of NAT
  - Change source TCP/UDP port in packet header to some unused value
  - Records the mapping, so the reverse changes can be made to any incoming replies as they traverse the NAT in the reverse direction

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

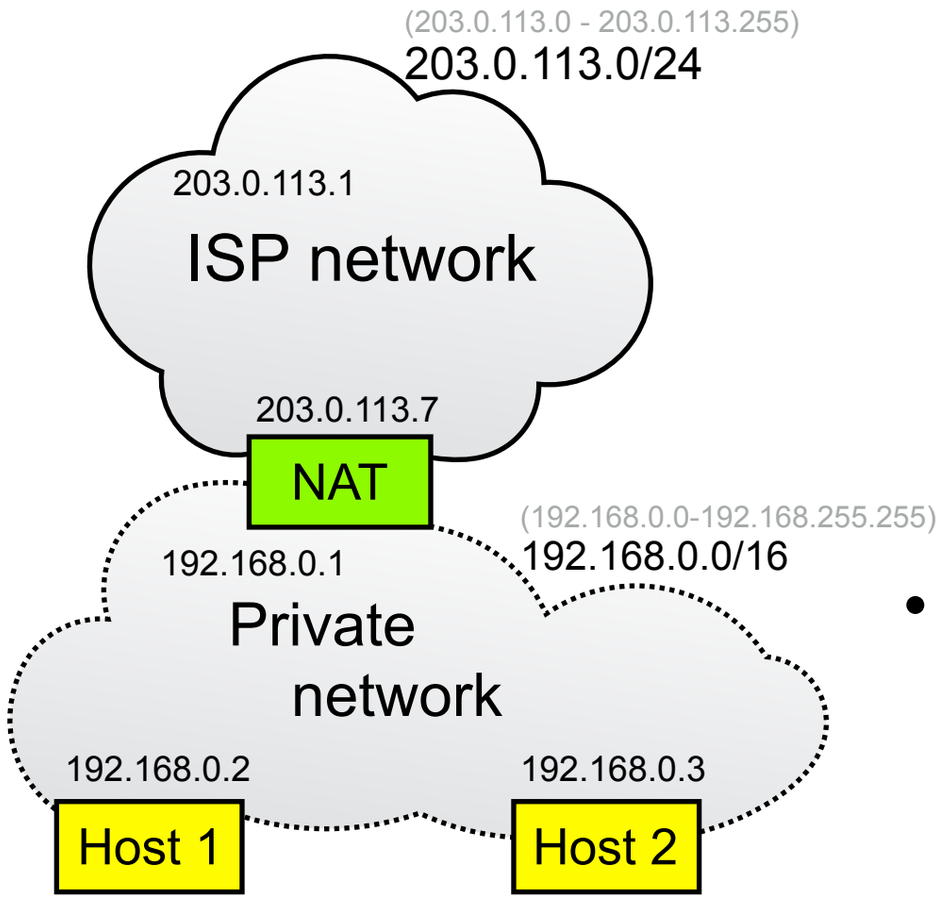
NAT and Private Address Ranges



- The NAT hides a private network behind a single public IP address
- Private IP network addresses:
  - 10.0.0.0/8
  - 176.16.0.0/12
  - 192.168.0.0/16
- Tries to give the illusion of more address space by reusing addresses in different parts of the network
  - e.g., most home networks use 192.168.0.0/16

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Problems due to NAT



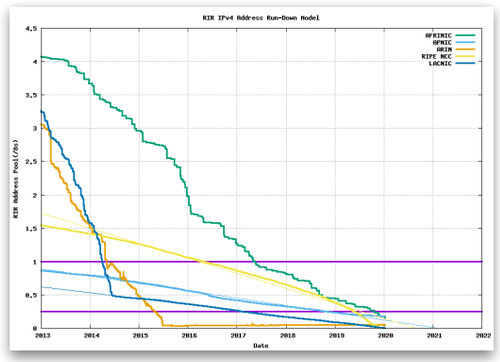
- Many applications fail with NAT:
  - Client-server applications with client behind NAT work without changes – web and email
  - Client-server applications with server behind NAT fail – need explicit port forwarding
  - Peer-to-peer applications fail – complex NAT traversal algorithm needed to connect

- NAT provides no security benefit:
  - NAT devices often include a firewall to help protect the private network
  - NAT function itself provides little security or privacy benefit

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

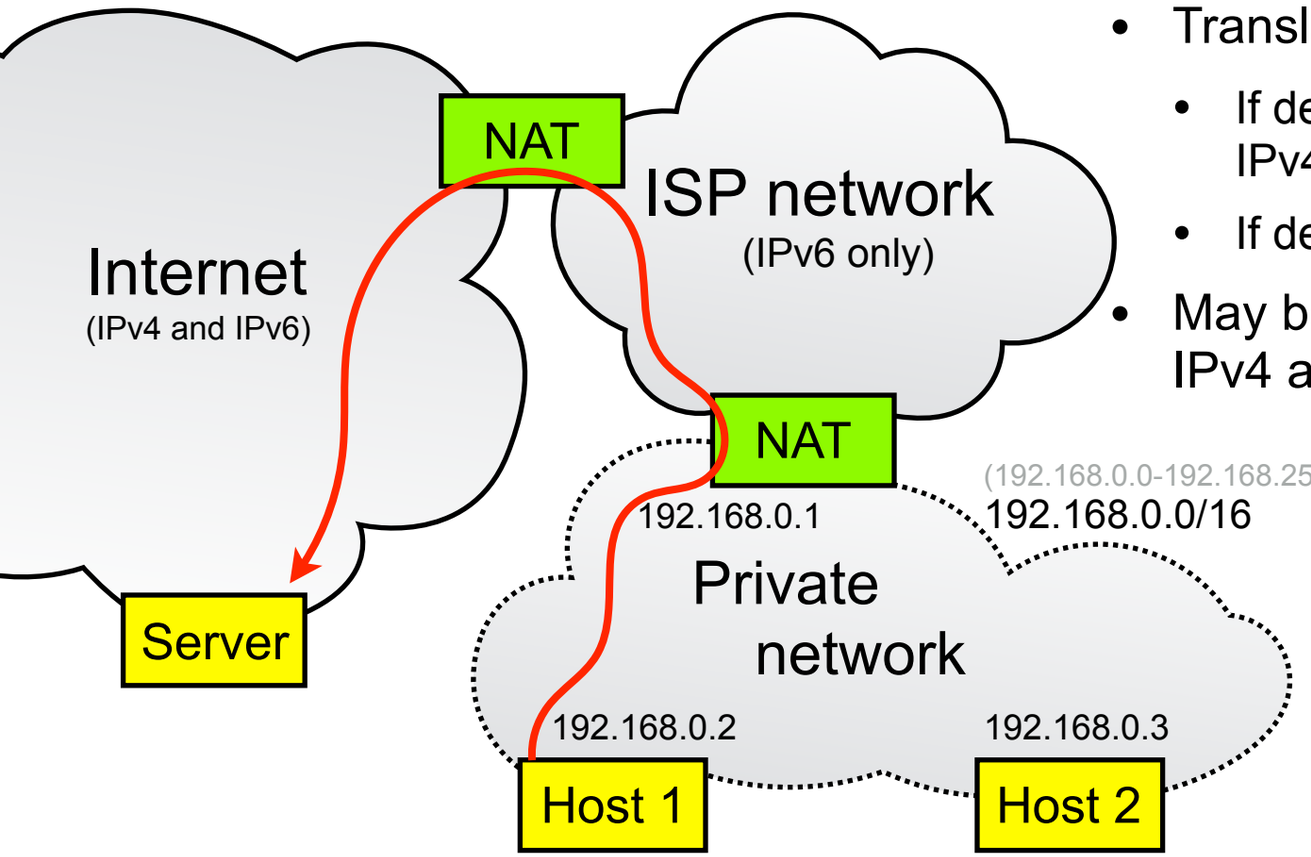
NAT Breaks Applications – Why Use It? (1/3)

- To work around lack of IPv4 address space:
  - Many ISPs have insufficient IPv4 addresses to give their customers a large enough prefix – each customer given one IPv4 address and a NAT
  - Many customers don't want to pay their ISP for more IPv4 addresses – addresses are scarce, so expensive
  - IPv6 is designed to make addresses cheap and plentiful, to avoid these problems



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

NAT Breaks Applications – Why Use It? (2/3)



- To translate between IPv4 and IPv6 addresses
  - ISP network runs IPv6 only; customers given public IPv6 addresses
  - Translate IPv4-to-IPv6 as packets leave private network
    - If destined for IPv4 host on the Internet, translate back to IPv4 on leaving ISP network
    - If destined for IPv6 host on the Internet, forward directly
- May be useful if ISP has more customers than it has IPv4 addresses

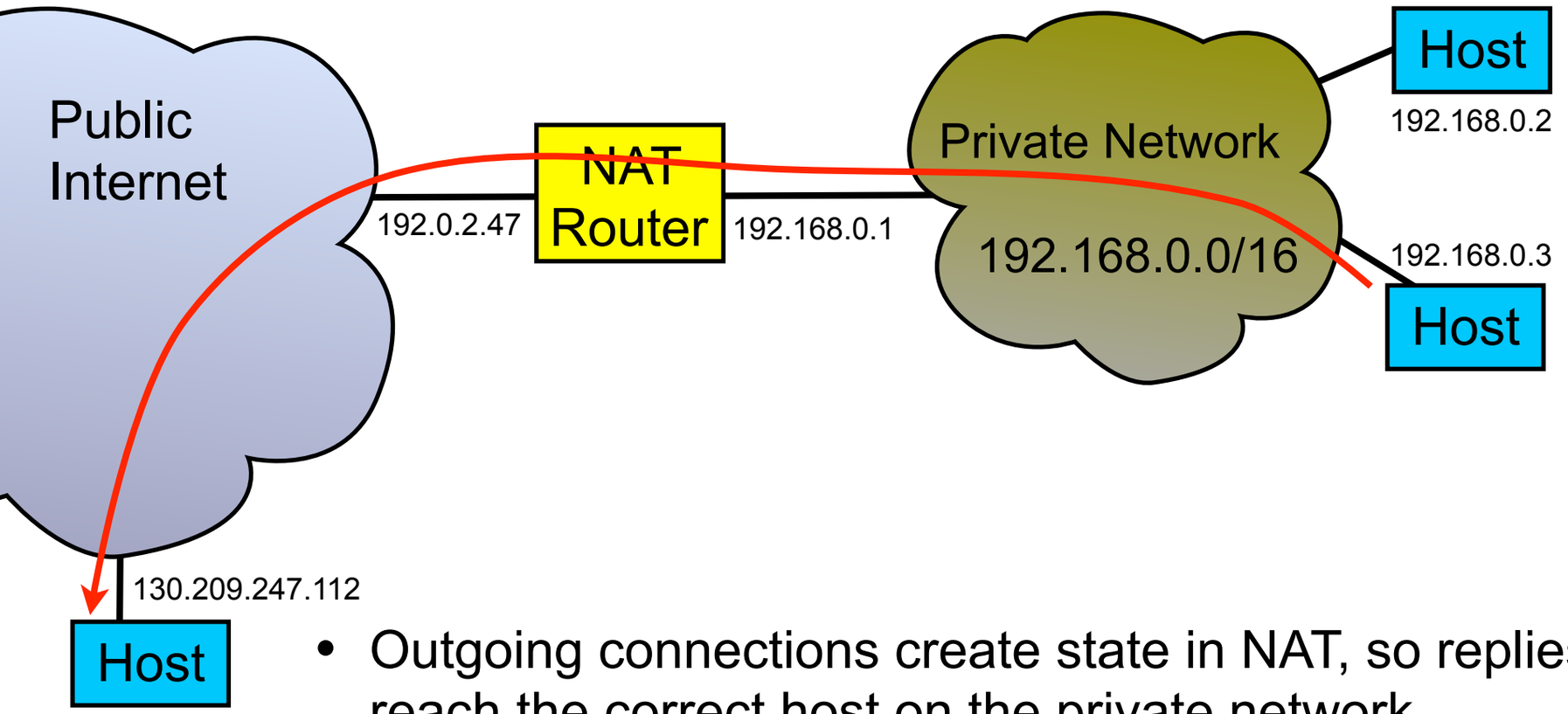
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

NAT Breaks Applications – Why Use It? (3/3)

- To avoid re-numbering a network when changing to a new ISP
  - Hard-coding IP addresses, rather than DNS names, in configuration files and application is a bad idea
  - Many people do it anyway – makes changing IP addresses difficult
  - IPv6 tries to make renumbering networks easier, by providing better auto-configuration
    - Insufficient experience to know how well this works in practice
  - Some vendors also offer IPv6-to-IPv6 NAT [RFC 6296]

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

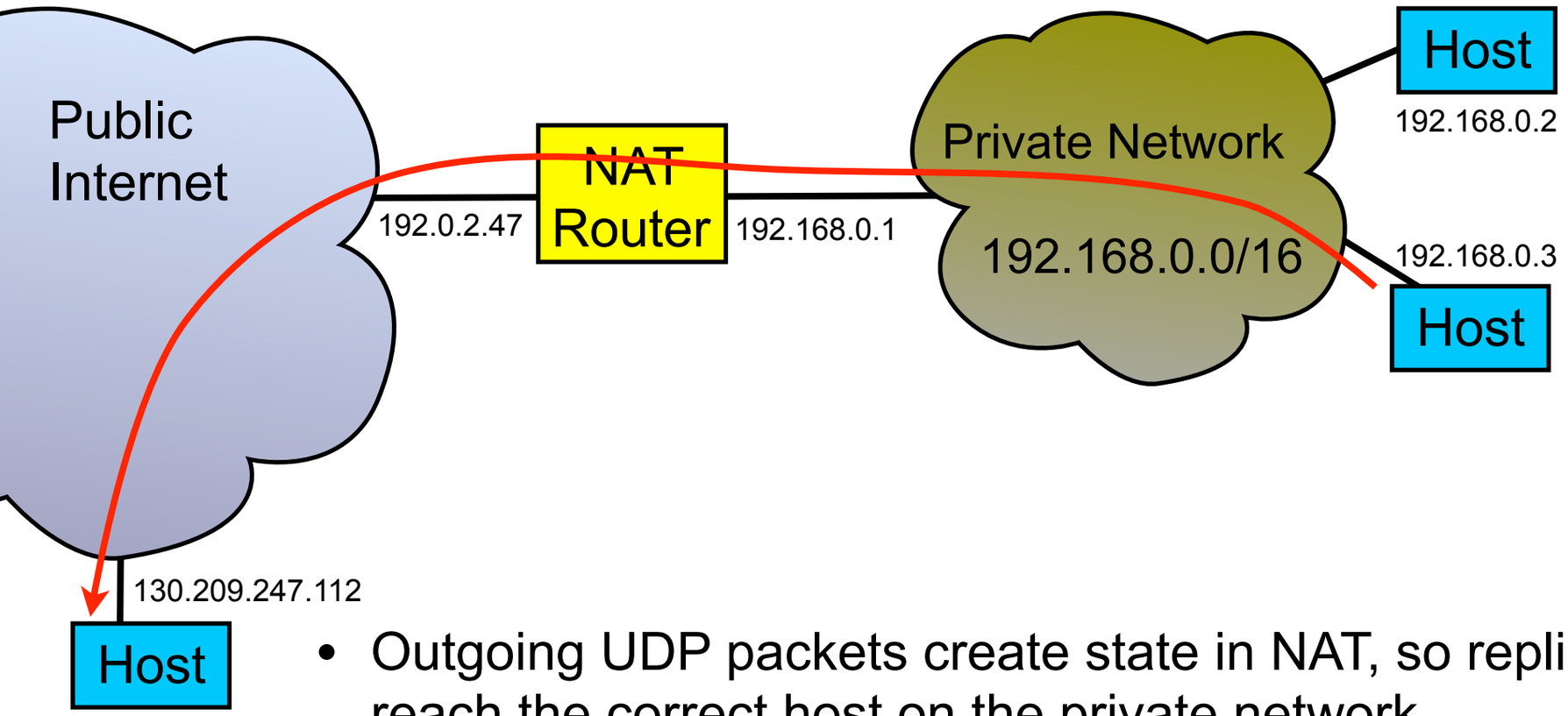
Implications of NAT for TCP Connections



- Outgoing connections create state in NAT, so replies can be translated to reach the correct host on the private network
  - Need to send data periodically, else NAT will assume the connection has failed
  - Recommended time out interval is 2 hours, many NATs use shorter [RFC5382]
- No state for incoming connections
  - NAT can’t know where to forward incoming connections, without manual configuration
  - Complicates running a server behind the NAT, or peer-to-peer applications

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Implications of NAT for UDP Flows

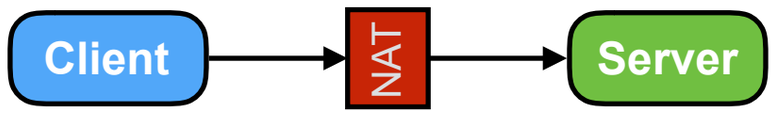


- Outgoing UDP packets create state in NAT, so replies can be translated to reach the correct host on the private network
  - UDP not connection-oriented; NAT can't detect the end of a flow, so use short timeout to cleanup state once UDP flow has stopped
    - UDP NAT traversal standards suggest sending a keep-alive every 15 seconds [RFC4787]
- No state for incoming connections
  - UDP NATs often more permissive about allowing incoming packets than TCP NATs; many allow replies from anywhere to an open port – simpler for peer-to-peer traffic

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Peer-to-peer NAT Traversal Concepts

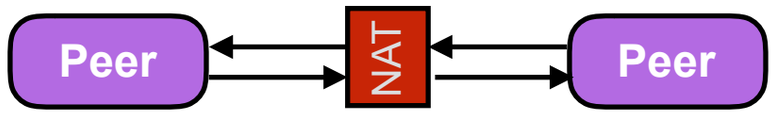
- NATs support outbound connections from client to server



- Incoming connections fail, since NAT cannot know how to translate the incoming packets

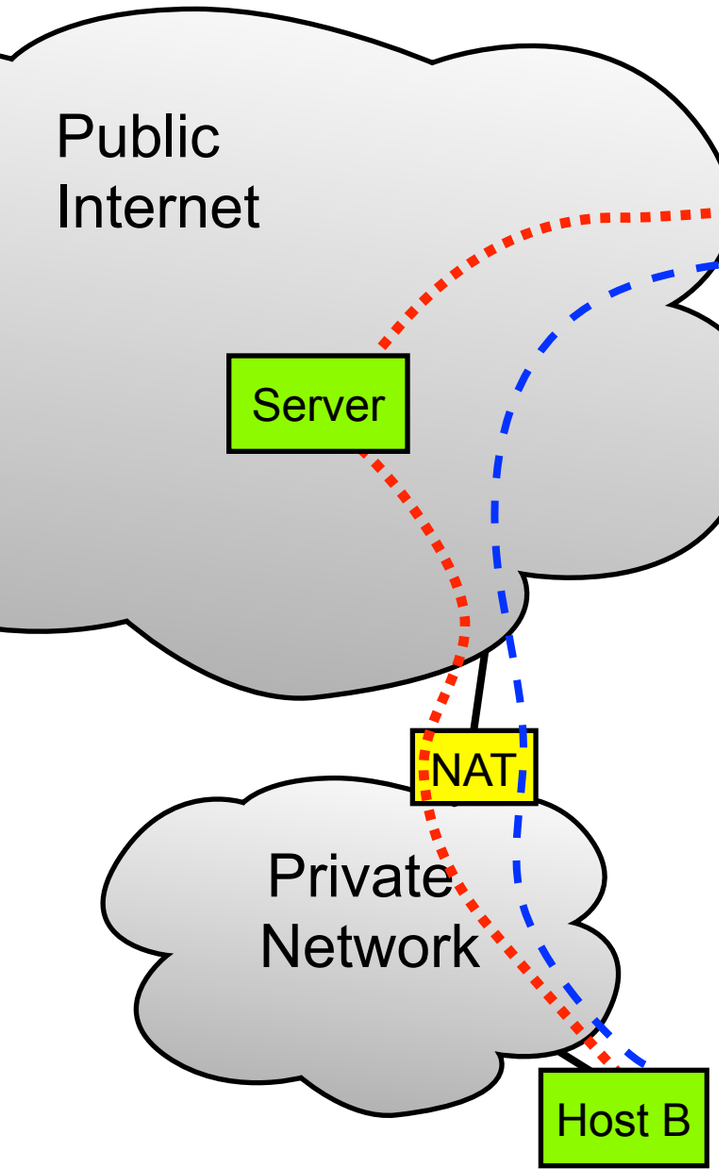


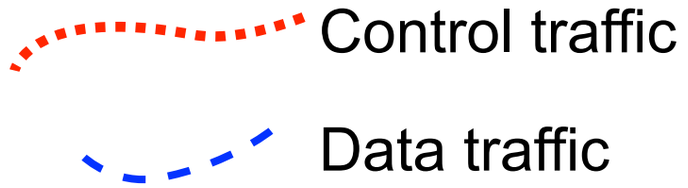
- Peer-to-peer connections can succeed if both NATs think a client server connection is being opened, and the response is coming from the server



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Peer-to-peer NAT Traversal Concepts

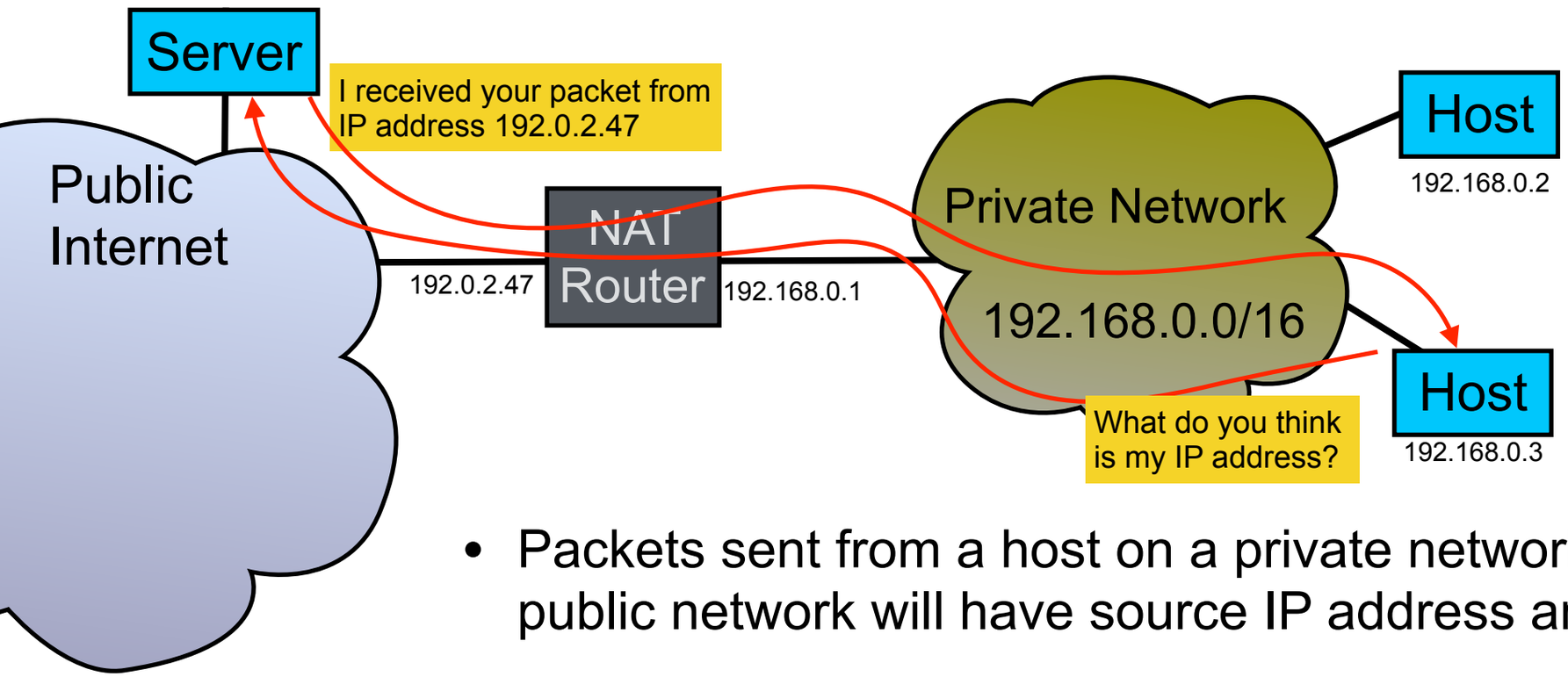




- Referral server on public network used to discover external (mapped) address/port on the NAT
- Referral server exchanges possible addresses with peer
- Peers systematically try to make connection using every possible combinations of addresses
  - Every possible network interface and protocol, mapped and local
  - Complex and generates significant traffic overhead

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

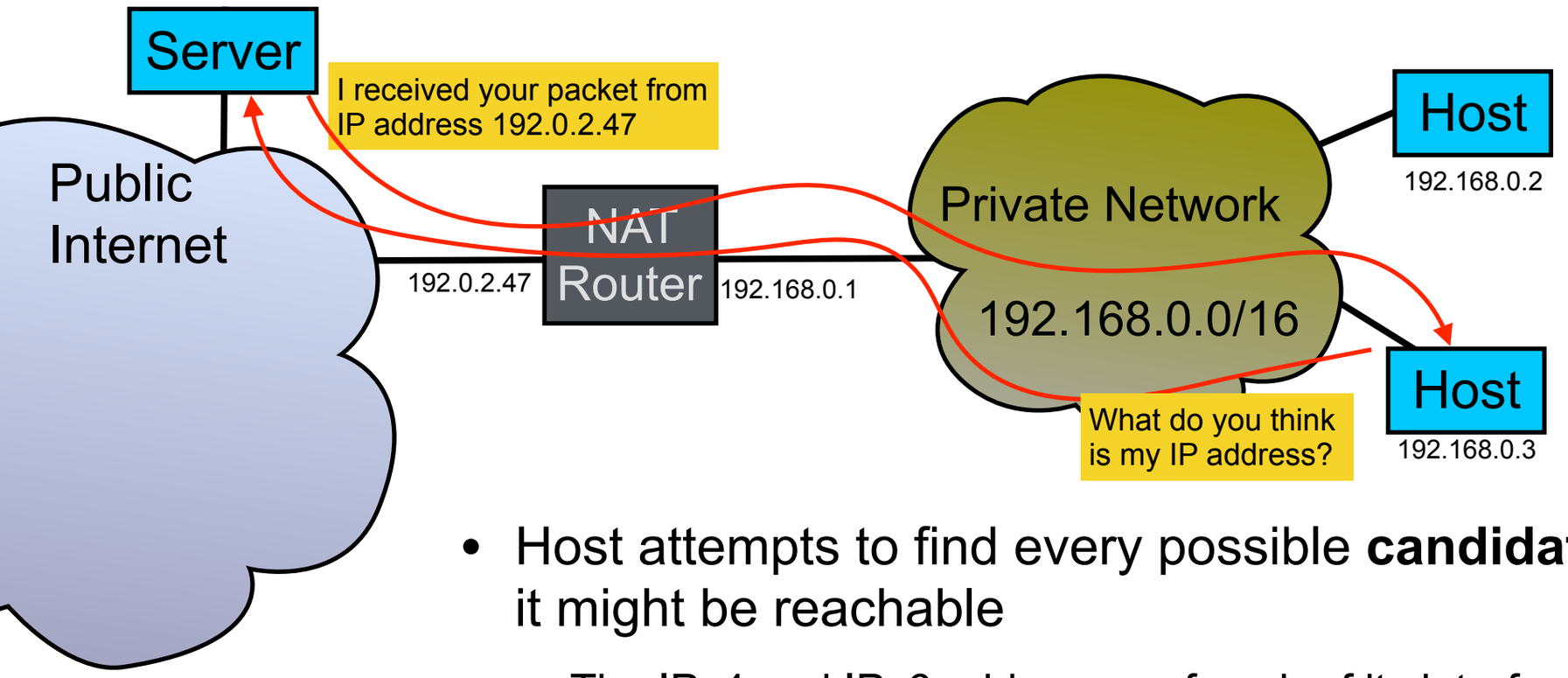
Binding Discovery (1/2)



- Packets sent from a host on a private network to a server on the public network will have source IP address and port translated
- The server can see the translated address/port, and send a reply back to the host telling it the address its packets appear to come from – the server reflexive address
- Known as **NAT binding discovery**
- The **STUN** protocol is a standard binding discovery mechanism
  - Under revision in IETF
  - RFC 5389 → https://datatracker.ietf.org/doc/draft-ietf-tram-stunbis/
  - Very widely used; open source code available

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

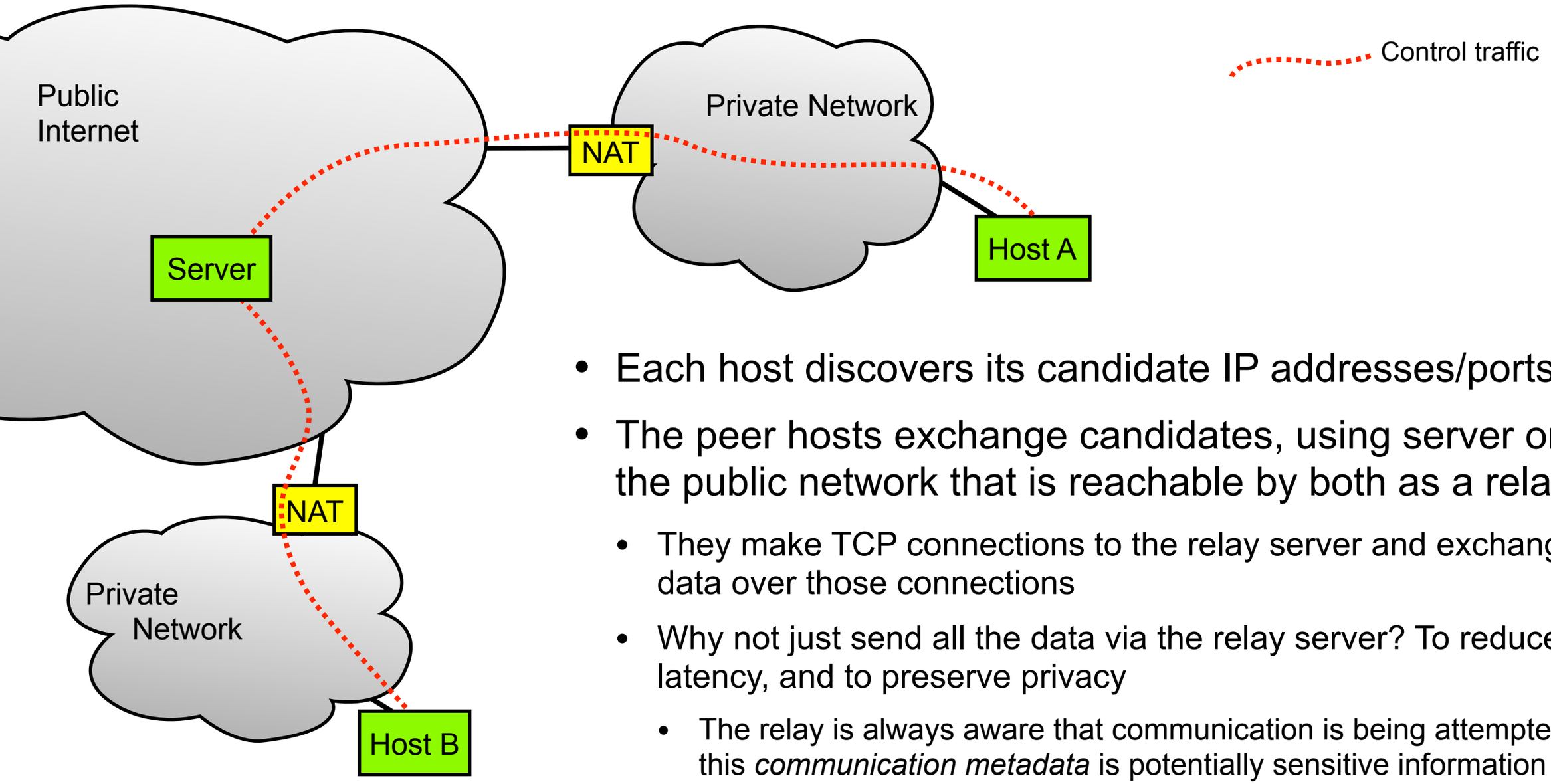
Binding Discovery (2/2)



- Host attempts to find every possible **candidate** IP address, on which it might be reachable
  - The IPv4 and IPv6 addresses of each of its interfaces (Wi-Fi, Ethernet, 4G, …)
  - For each address, any server reflexive addresses discovered using STUN from that address
  - Any relayed addresses (e.g., via a TURN or SOCKS proxy, VPN server, etc.)
- Aim to find every possible IP address from which the peer might be reachable

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

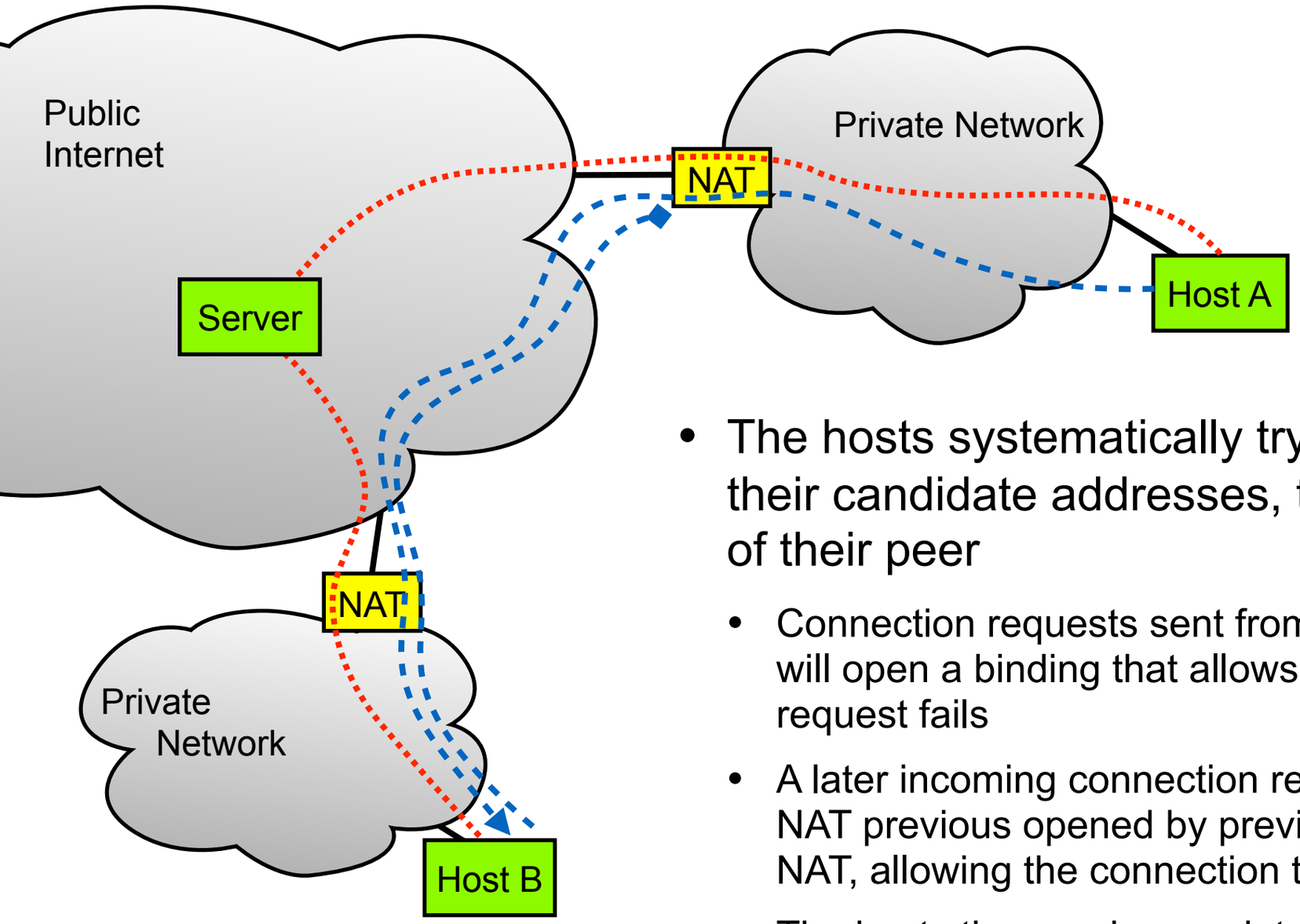
Exchange Candidates



- Each host discovers its candidate IP addresses/ports
- The peer hosts exchange candidates, using server on the public network that is reachable by both as a relay
  - They make TCP connections to the relay server and exchange data over those connections
  - Why not just send all the data via the relay server? To reduce latency, and to preserve privacy
    - The relay is always aware that communication is being attempted; this *communication metadata* is potentially sensitive information

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Connectivity Checks: The ICE Algorithm



- The hosts systematically try **`connect()`** from each of their candidate addresses, to every candidate address of their peer
  - Connection requests sent from a host that passes through a NAT will open a binding that allows a response, even if the connection request fails
  - A later incoming connection request that reaches the port on the NAT previous opened by previous outgoing request will pass the NAT, allowing the connection to succeed
  - The hosts then exchange data over the path to confirm success
- **The ICE algorithm,** RFC 8445, describes how to do this

https://datatracker.ietf.org/doc/rfc8445/

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Peer-to-peer NAT Traversal

- Binding discovery and systematic connection probing is complex, slow, and generates a lot of unnecessary traffic
- Effective for UDP traffic
  - Developed to support VoIP applications, that send UDP packets
  - Connectionless nature of UDP means NATs tend to be permissive about allowing incoming packets, provided they reach the correct port
- Less effective for TCP connections
  - NATs often require an exact match for incoming packets to a previous outgoing TCP packet – addresses, ports, TCP sequence numbers – before they allow a connection to be established

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Connection Establishment in a Fragmented Network

- Client-server connections sensitive to latency → reducing RTTs, racing
- Peer-to-peer connections require complex NAT traversal mechanisms
- Connection establishment in the modern network is complex

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow