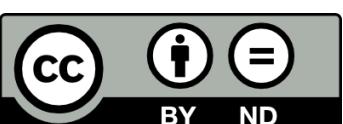




University  
ofGlasgow

# The Changing Internet

Networked Systems (H)  
Lecture 1



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Course Administration

- Aims, Objectives, Learning Outcomes
- Timetable
- Assessment
- Recommended Reading

# Course Coordinator and Materials

- Lecturer and course coordinator
  - Dr Colin Perkins ([colin.perkins@glasgow.ac.uk](mailto:colin.perkins@glasgow.ac.uk))
  - Office: S101b, Lilybank Gardens
- Course materials on Moodle and <https://csperkins.org/teaching/>
  - The website has lecture transcripts that are not on Moodle



# Aims and Objectives

- To introduce fundamental concepts and theory of communications
- To provide a solid understanding of the technology that supports modern networked computer systems
- To give students the ability to evaluate and advise industry on the use and deployment of networked systems
- To introduce low-level network programming, and give students practice with systems programming in C

# Intended Learning Outcomes

- By the end of the course, you should be able to:
  - Describe and compare capabilities of various communication technologies and techniques;
  - Know the differences between networks of different scale, and how these affect their design;
  - Understand demands of different applications on quality of service requirements for the underlying communication network;
  - Describe the issues in connecting heterogeneous networks;
  - Describe importance of layering, and the OSI reference model;
  - Demonstrate an understanding of the design and operation of an IP network, such as the Internet, and explain the purpose and function of its various components; and
  - Write simple low-level communication software, showing awareness of good practice for correct and secure programming

# Course Structure

Week	Thursday 13:00-15:00 – Labs	Thursday 16:00-18:00 – Lecture and Discussion
1		#1: The Changing Internet
2	#1: Introduction to Network Programming in C	#2: Connection Establishment in a Fragmented Network
3	#2: Connection Establishment	#3: Secure Communications
4	#3: Transport Layer Security	#4: Improving Secure Connection Establishment
5	#4: QUIC	#5: Reliability and Data Transfer
6		#6: Lowering Latency
7	#5: TCP Behaviour and Congestion Control	#7: Real-time and Interactive Applications
8		#8: Naming and the Tussle for Control
9	#6: Network Topology	#9: Networks and Inter-domain Routing
10		#10: Future Directions (video lecture only)

# Course Structure

Week	Thursday 13:00-15:00 – Labs	Thursday 16:00-18:00 – Lecture and Discussion
1		#1: The Changing Internet
2	#1: Introduction to Network Programming in C	#2: Connection Establishment in a Fragmented Network
	<ul style="list-style-type: none"><li>• Lecture recordings will be made available ahead of time</li><li>• Each lecture is accompanied by discussion questions</li><li>• Timetabled session from 16:00-18:00 on Thursdays is for discussion – you <b>must</b> watch the lecture and think about the discussion questions <b>before</b> the timetabled slot</li></ul>	#3: Secure Communications #4: Improving Secure Connection Establishment #5: Reliability and Data Transfer #6: Lowering Latency
7	#5: TCP Behaviour and Congestion Control	#7: Real-time and Interactive Applications
8		#8: Naming and the Tussle for Control
9	#6: Network Topology	#9: Networks and Inter-domain Routing
10		#10: Future Directions (video lecture only)

# Course Structure

Week	Thursday 13:00-15:00 – Labs	Thursday 16:00-18:00 – Lecture and Discussion
1		#1: The Changing Internet
2	#1: Introduction to Network Programming in C	#2: Connection Establishment in a Fragmented Network
3	#2: Connection Establishment	#3: Secure Communications
4	#3: Transport Layer Security	<ul style="list-style-type: none"><li>• Labs materials will be made available in advance, and are to be completed in your own time</li><li>• Timetabled sessions from 13:00-15:00 on Thursdays are for live support with the lab exercises – try to solve the exercise, and think of questions you might need to ask, <b>before</b> the timetabled support slot</li></ul>
5	#4: QUIC	
6		
7	#5: TCP Behaviour and Congestion Control	
8		#8: Naming and the Tussle for Control
9	#6: Network Topology	#9: Networks and Inter-domain Routing
10		#10: Future Directions (video lecture only)

# Assessed Exercise

- The assessed exercise is worth 20% of the marks for this course
  - Available on the day of Lecture 5
  - Deadline on the day of Lecture 7
  - Note well:
    - Following the code of assessment, late submissions of assessed exercises will be accepted for up to 5 working days beyond the due date. Such late submissions will receive a two band penalty for each working day, or part thereof, the submission is late. Submissions that are received more than five working days after the due date will be awarded a band of H.  
**Submissions of assessed exercises that do not follow the instructions given in the handout will be given a two band penalty. These penalties will be strictly enforced.**
    - If you are ill, or have other circumstances that may affect your submission, then you may contact the course coordinator **before the deadline** to request an extension, following the usual procedure.

# Examination

- The final exam is worth 80% of the marks for this course
- Exam format: answer all three questions
  - The aim is to test your understanding of the material, not just to test your memory of all the details – **explain why, don't just recite what**
  - Note that the University code of assessment ([https://www.gla.ac.uk/media/Media\\_106264\\_smxx.pdf](https://www.gla.ac.uk/media/Media_106264_smxx.pdf)) states that excellent performance (Grade A) is likely to be characterised by various factors, including:
    - reasoned arguments developing logical conclusions
    - evidence of wide, relevant reading
    - application of learning to new situations and problem solving
  - Material covered in the lectures and lab exercises is examinable (you will not be expected to write code in the exam) – you are also expected to follow the required readings
- Past papers are on Moodle and on the website

# Required Reading

- You are expected to read one of the following:
  - Peterson and Davie, Computer Networks: A Systems Approach, 5th Edition, Morgan Kaufman, 2011, ISBN 0123851386 (£). The authors make an updated version of this book available for free online at <https://book.systemsapproach.org>
  - Bonaventure, Computer Networking: Principles, Protocols and Practice, free online textbook available at <https://www.computer-networking.info/>
  - Kurose and Ross, Computer Networking: A Top-Down Approach, 8th Edition, Addison-Wesley, 2021, ISBN 978-1292405469 (£)
  - Tanenbaum, Feamster, and Wetherall, Computer Networks, 6th Edition, Prentice Hall, 2021, ISBN 978-1292374062 (£)
- **You are expected to read-along with the lectures – the lectures introduce the core ideas, while the books and linked papers, RFCs, and blog posts provide detail and context**

# The Changing Internet

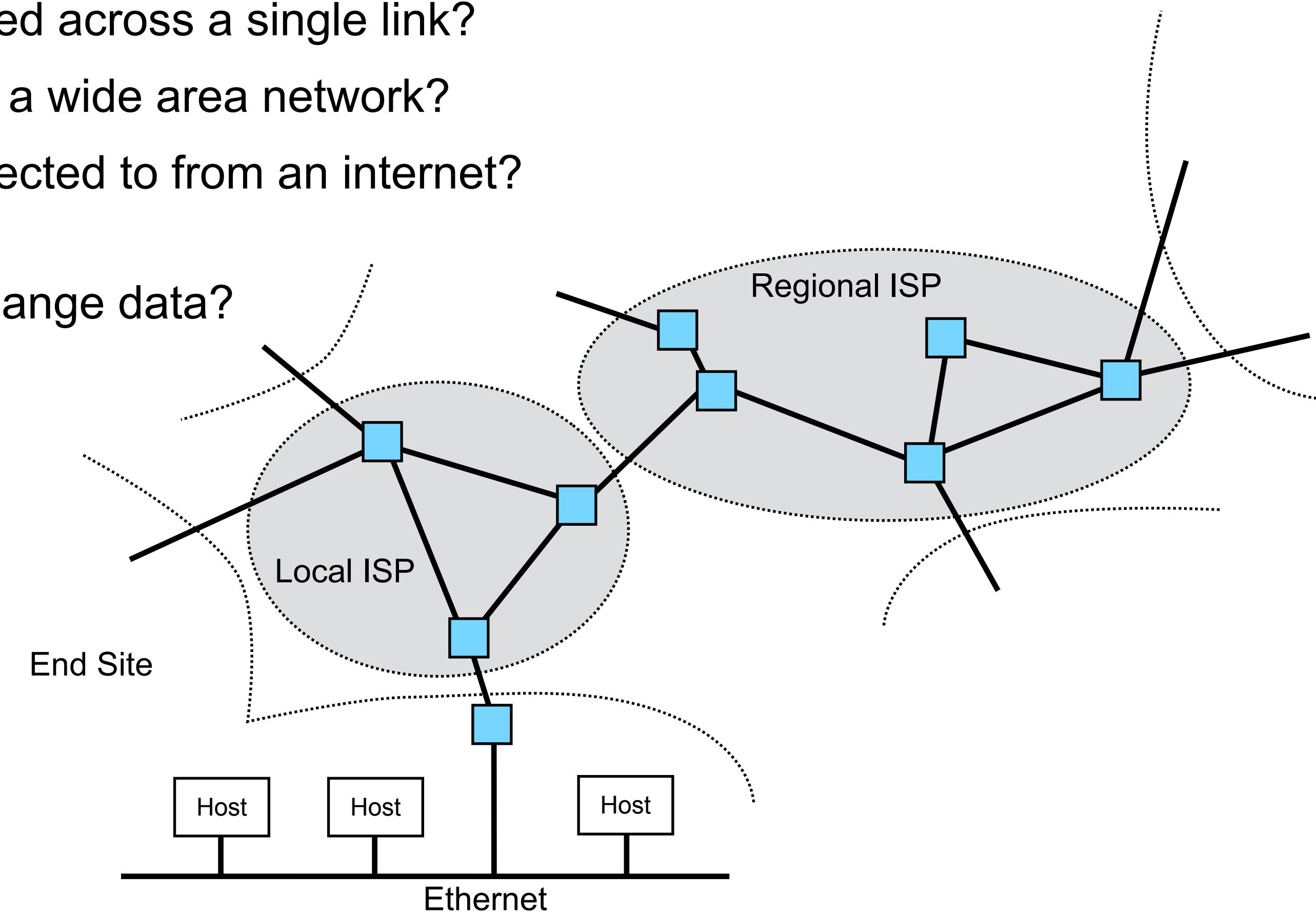
- Review of the Internet architecture
  - Protocols and Layers
  - Physical and Data Link Layers
  - Network Layer and Internet Protocols
  - Transport Layer
  - Higher Layer Protocols
- The Changing Internet

# Protocols and Layers

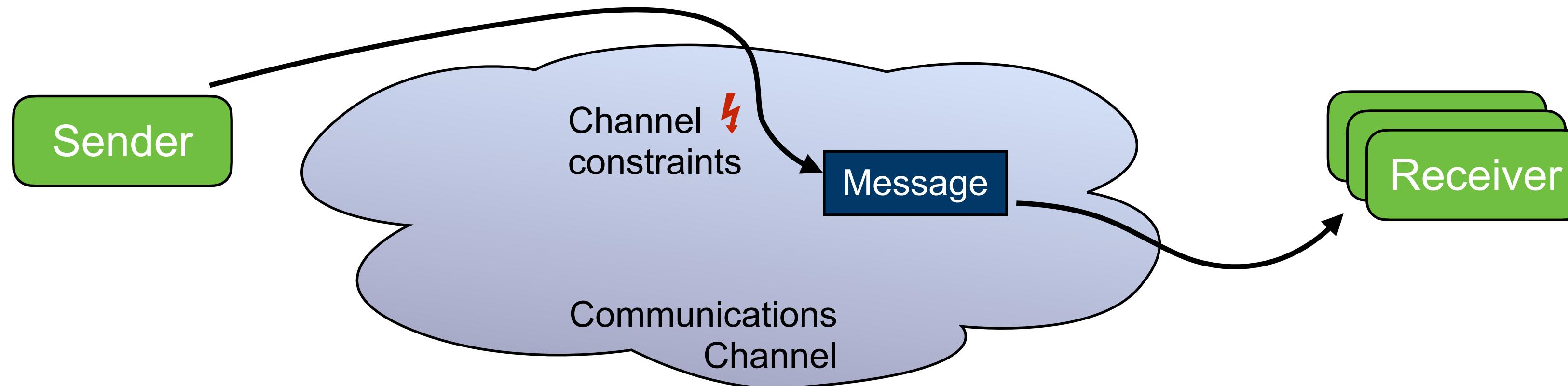
- What is a networked system?
- Protocols and Layering

# What is a Networked System?

- A cooperating set of autonomous computing devices that exchange data to perform some application goal
  - Communication – how is information exchanged across a single link?
  - Networking – how are links connected to form a wide area network?
  - Internetworking – how are networks interconnected to form an internet? How is data routed across that network?
  - Transport – how do end-systems reliably exchange data?

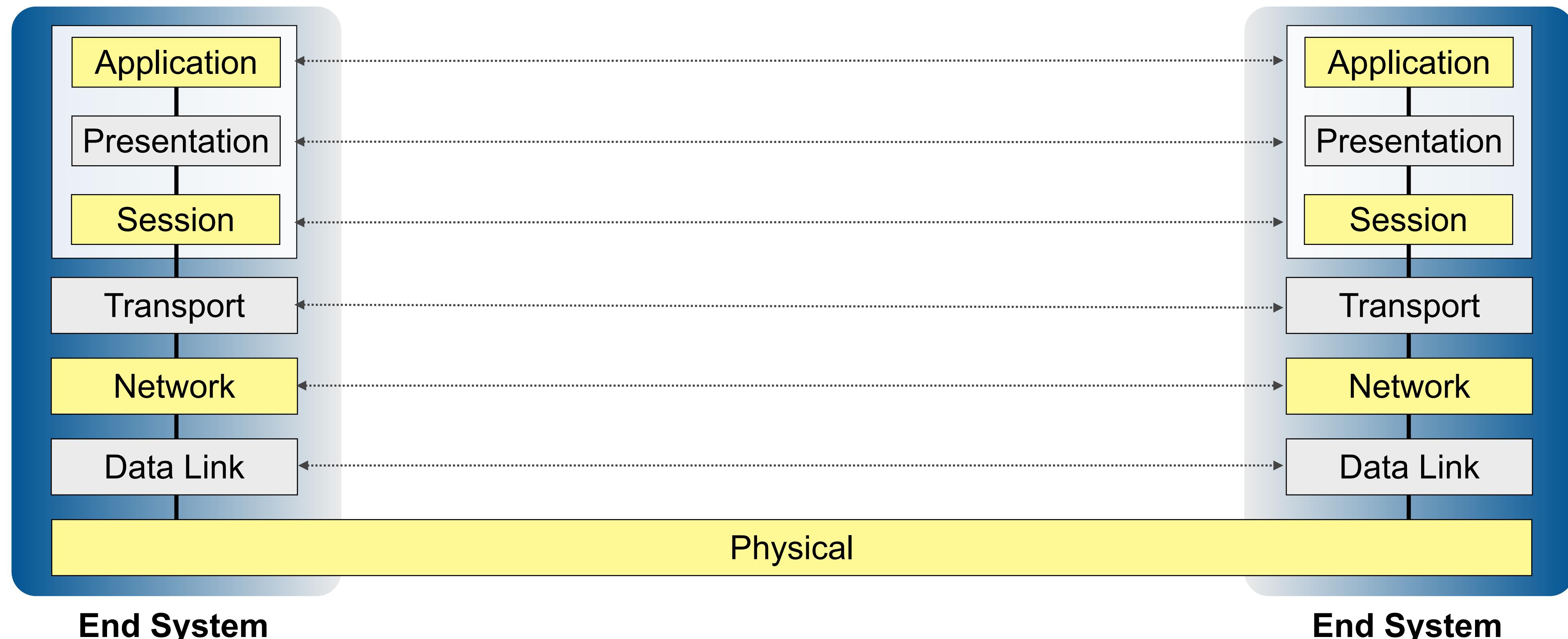


# Communication Protocols



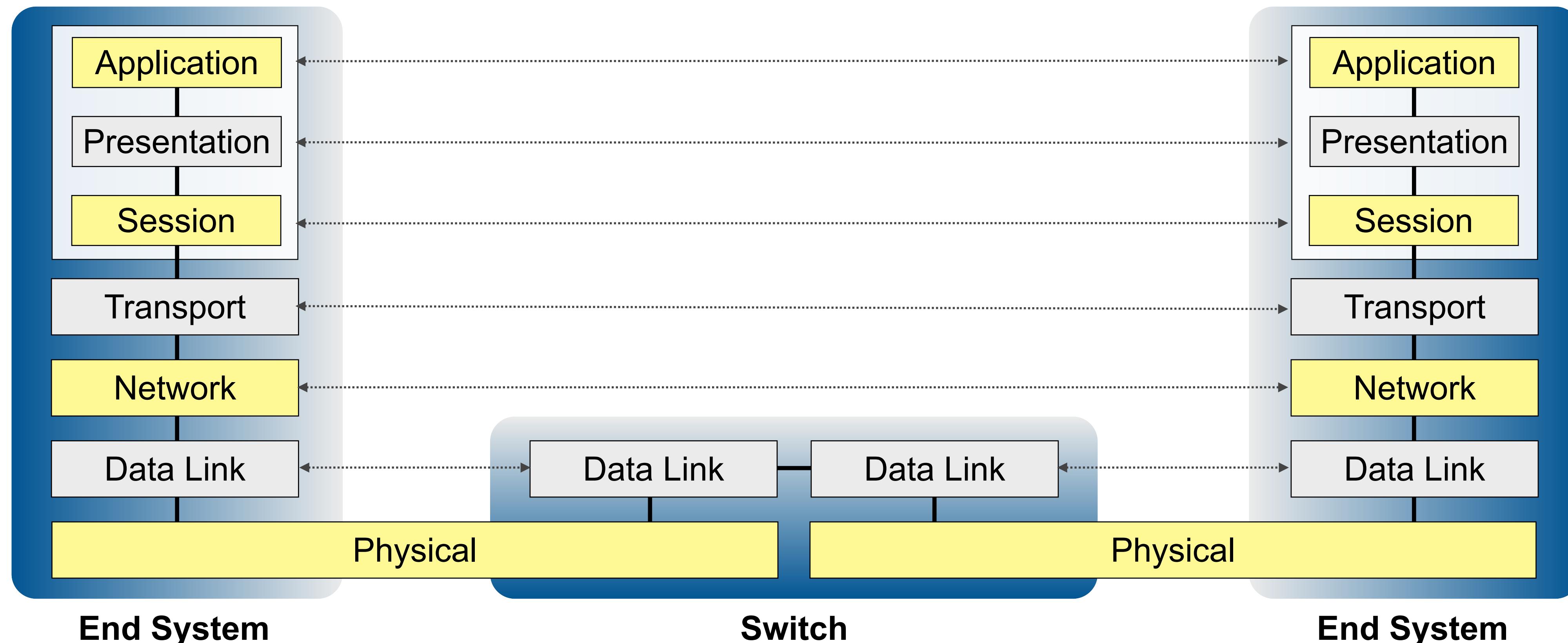
- Channel constraints bound communications speed and reliability
- The message syntax, semantics, and communication patterns form a **network protocol** – HTTP, TCP, IP, etc.
- Protocols can be composed and layered to raise level of abstraction

# Protocol Layering: The OSI Reference Model

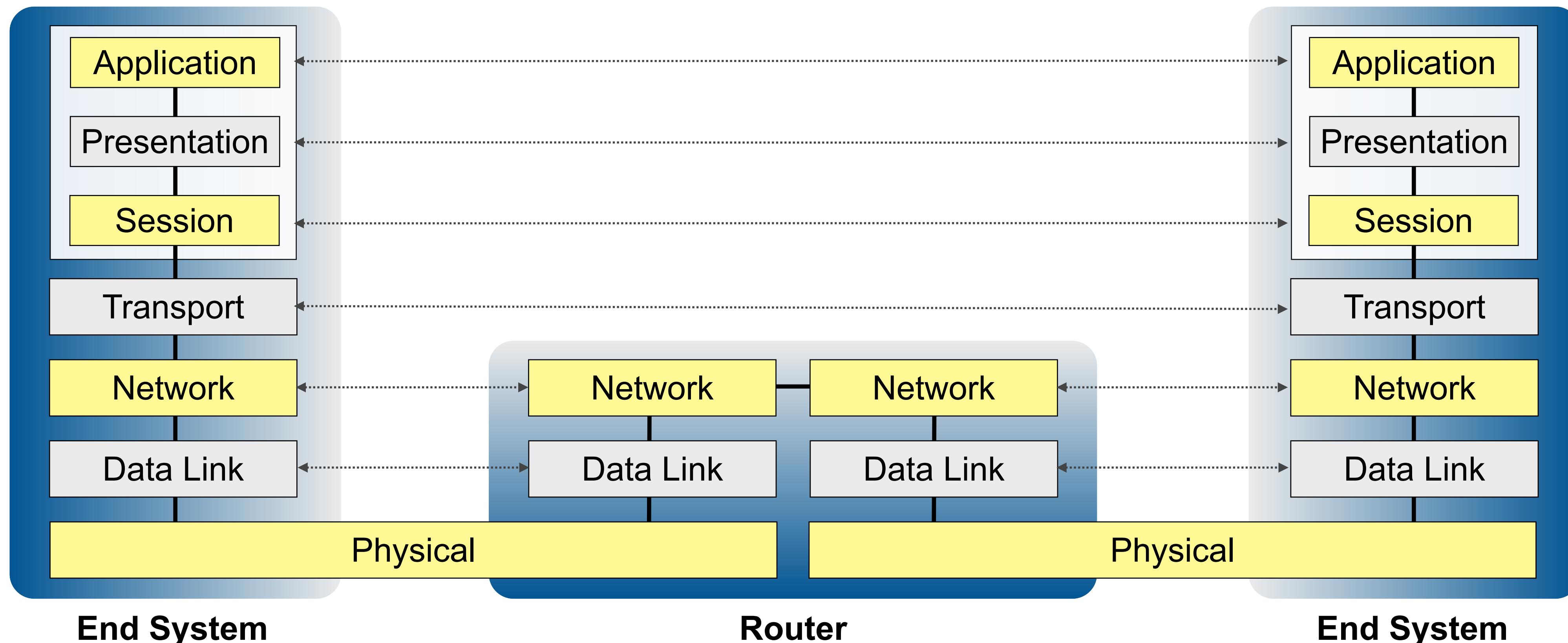


ISO/IEC 7498-1(1994) "Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model"

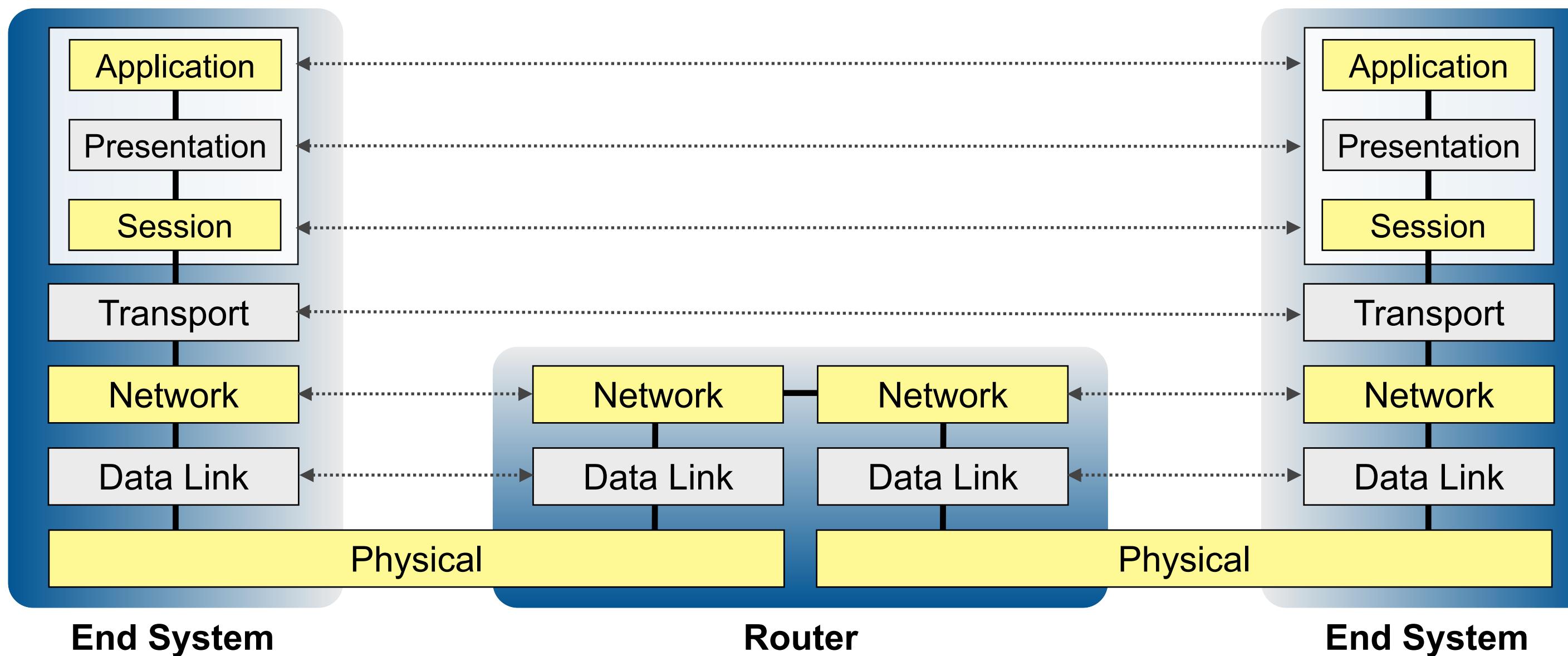
# Protocol Layering: The OSI Reference Model



# Protocol Layering: The OSI Reference Model



# Protocol Layering: The OSI Reference Model



- A standard model of layered protocol design – **real networks don't follow this model** – they're more complex, and layer violations, sublayers, and tunnels are commonplace
- A layered model is extremely useful for helping structure discussions about networks

# Protocols and Layers

- What is a networked system?
- Protocols and Layering

See also: J. Macculey, S. Shenker, and G. Varghese,  
[Extracting the Essential Simplicity of the Internet](#),  
Communications of the ACM, Volume 66, Number 2,  
February 2023. DOI: 10.1145/3547137.  
<https://www.youtube.com/watch?v=XY6fJVMaawI>

# Physical and Data Link Layers

- Physical link characteristics, modulation and transmission
- Data link layer: framing and addressing, media access control

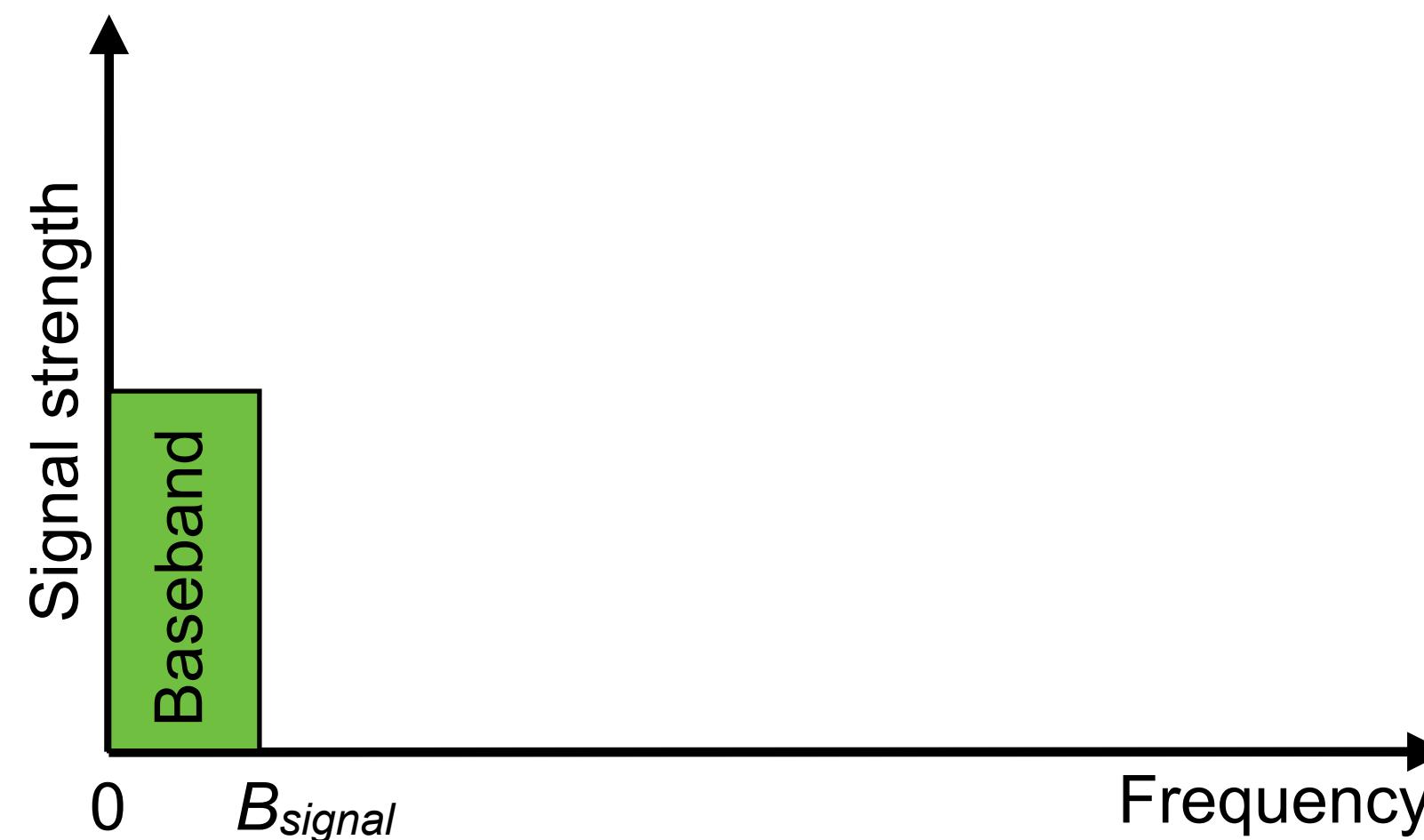
# The Physical Layer

- Physical layer enables communication – transmission of raw bitstream
  - What type of cable or wireless link do you use?
  - How to encode bits onto that channel?
    - Baseband encoding
    - Carrier modulation
  - What is the capacity of the channel?

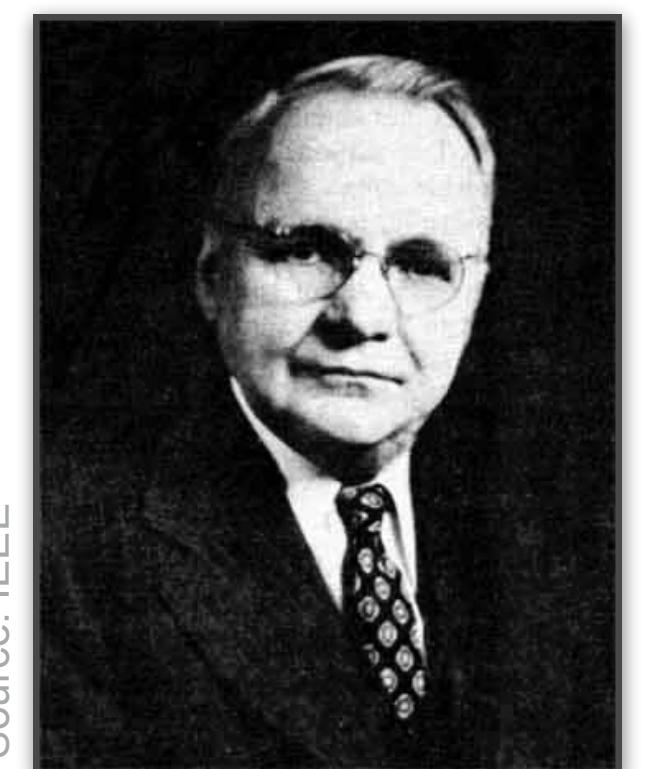


Image: Maria Theresa Perez, "Community Cellular Networks in the Philippines: Experiences from the VBTS project", Presentation to IRTF GAIA RG, November 2019, <https://www.ietf.org/proceedings/106/slides/slides-106-gaia-up-vbts-philippines-00>

# Encoding Data onto Wired Channels



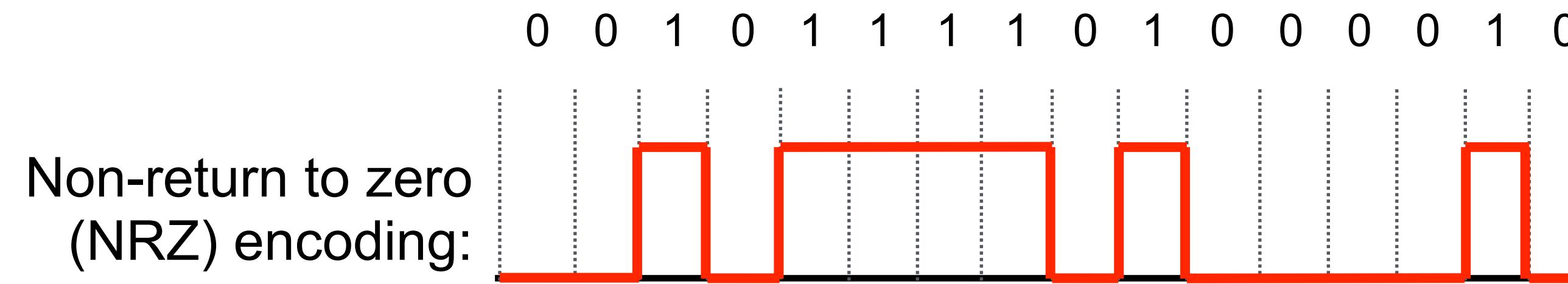
- Signal usually directly encoded onto the channel, by varying some property of the channel, and occupies **baseband** region
- The maximum bitrate of a channel is described by Nyquist's theorem:  $R_{max} \leq 2B \log_2 V$ 
  - $B$  = bandwidth of the channel
  - $V$  = number of discrete values per symbol
- Maximum data rate only reached with a noise-free channel



Source: IEEE

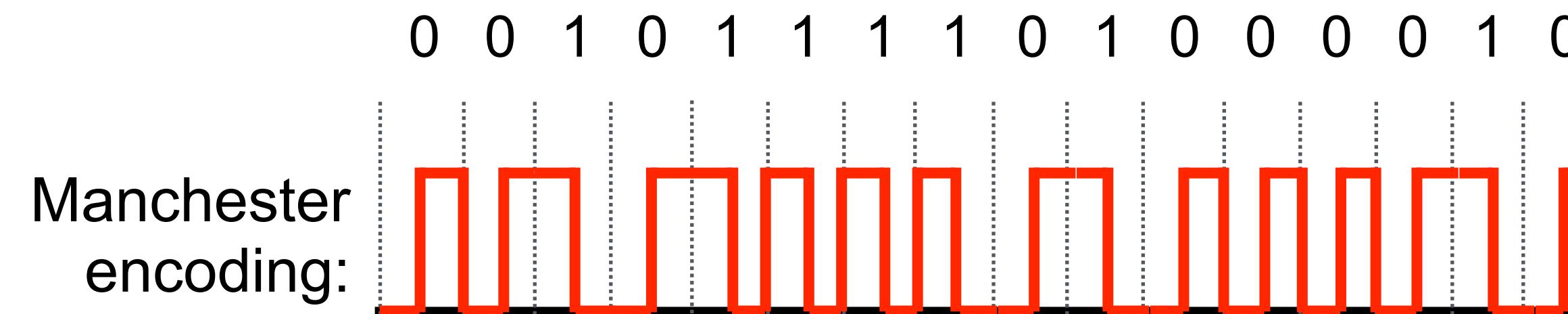
Harry Nyquist (1889-1976)

# Baseband Data Encoding



Encode a 1 as a high signal, 0 as low signal

No variation in waveform if long runs of same value sent – easy to miscount number of bits

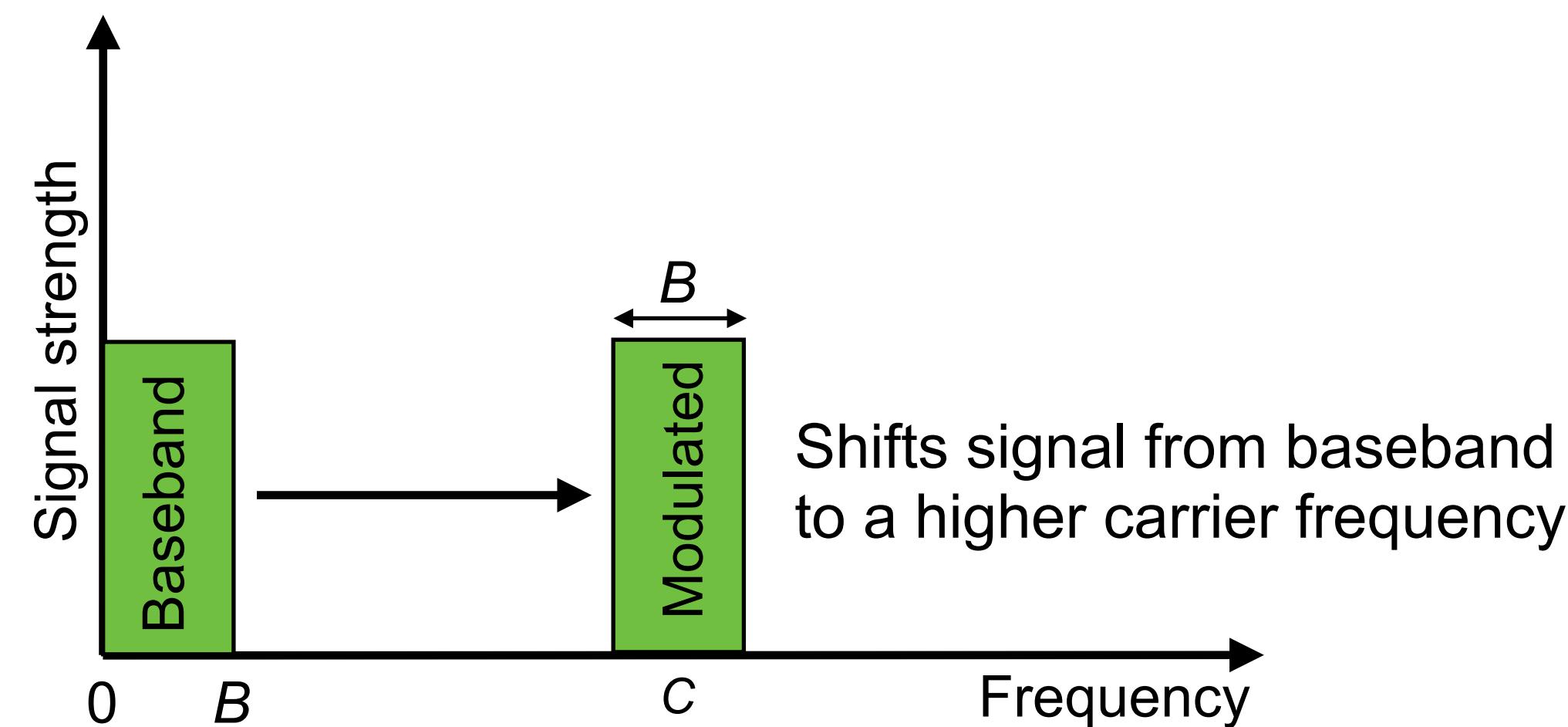


Encode a 1 as high-low transition, a 0 as a low-high transition

Doubles bandwidth needed, since transition on every bit, but avoids miscount

- Signal encoded onto the channel by varying channel characteristics – voltage applied to an electrical cable, intensity of laser in optical fibre

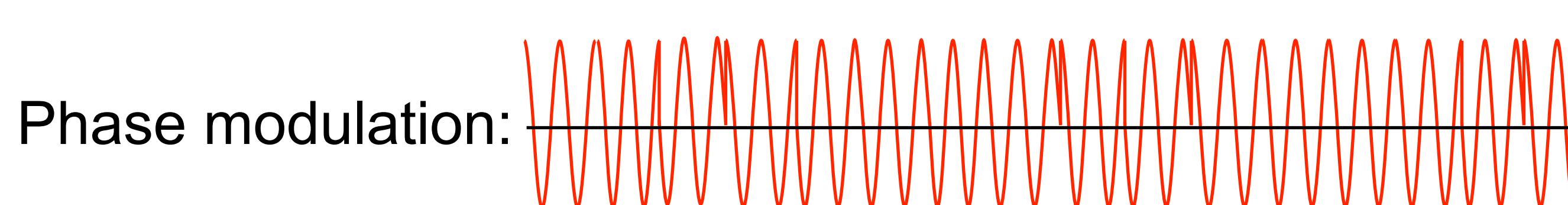
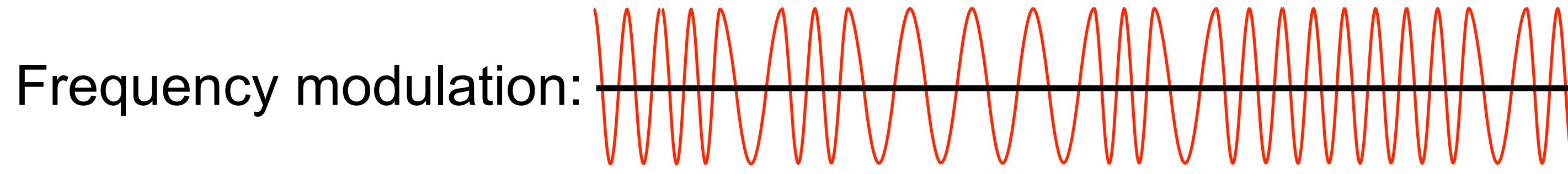
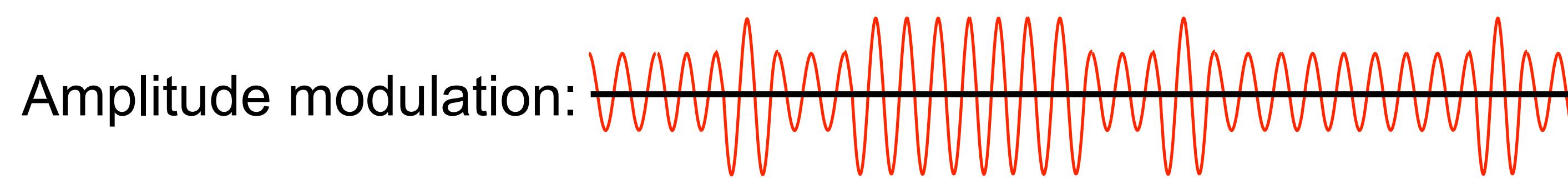
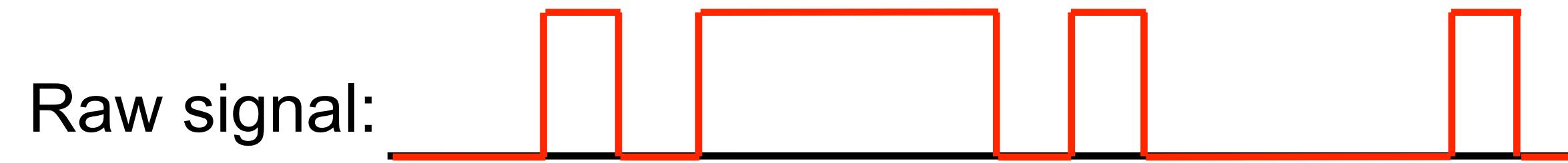
# Encoding Data onto Wireless Channels



- Carrier wave applied to channel at frequency,  $C$ ; signal modulated onto the carrier
- Bandwidth unchanged, but shifted to range centred on the carrier frequency

- Wireless links use carrier modulation, rather than baseband transmission
- Allows multiple signals on a channel, modulated onto carriers of different frequency
  - Compare: FM or AM radio stations
  - Can be used on wired links – this is how ADSL and voice telephones share a phone line
- Performance affected by carrier frequency, transmission power, modulation scheme, type of antenna, etc.

# Amplitude, Frequency, Phase Modulation



Complex modulations are possible:  
Gigabit Ethernet uses amplitude  
modulation with five different amplitudes

Modulation schemes are often combined:  
for example, dial-up modems vary phase  
and amplitude → **quadrature amplitude  
modulation** with 12 phase shift values at  
two different amplitudes

# Spread Spectrum Communication

- Single frequency channels prone to interference
  - Mitigate by repeatedly changing carrier frequency, many times per second: noise unlikely to affect all frequencies
  - Use a pseudo-random sequence to choose which carrier frequency is used for each time slot
  - Seed of pseudo-random number generator is shared secret between sender and receiver, ensuring security
- Example: 802.11b Wi-Fi uses spread spectrum using several frequencies centred at either 2.4 GHz or 5 GHz

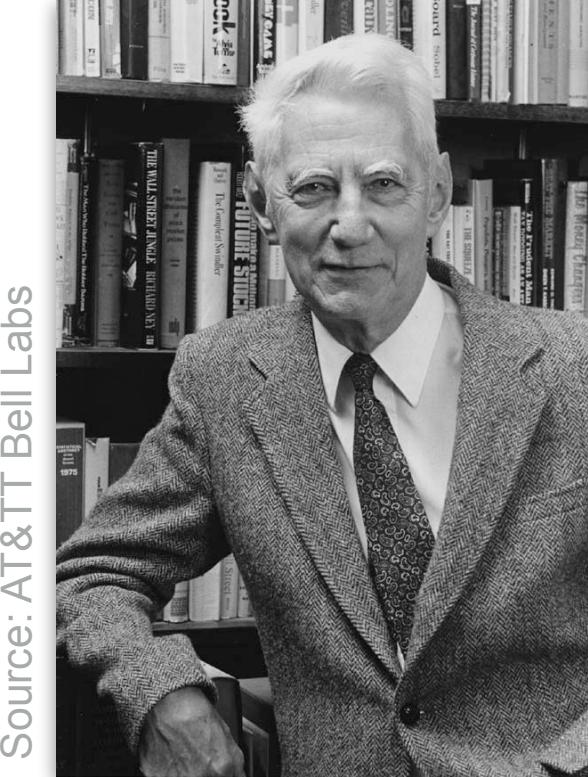


Source: Wikipedia/Public Domain

Hedy Lamarr (1914-2000)

# Physical Link Characteristics and Limitations

- Real-world network links are imperfect, and subject to **noise**
  - Electrical or radio interference, imperfections in optical fibre
- The Shannon-Hartley theorem predicts the maximum data rate of a channel subject to noise:  $R_{max} = B \log_2 (1 + S/N)$ 
  - $B$  = bandwidth of the channel
  - $S$  = signal strength
  - $N$  = noise strength
  - Assuming Gaussian noise: interference affects all frequencies equally
- Physical limit on maximum transmission rate
  - $B$  and  $N$  depend on properties of the channel
  - $S$  depends on transmission power → trade battery life for performance



Source: AT&T Bell Labs

Claude Shannon  
(1916-2001)

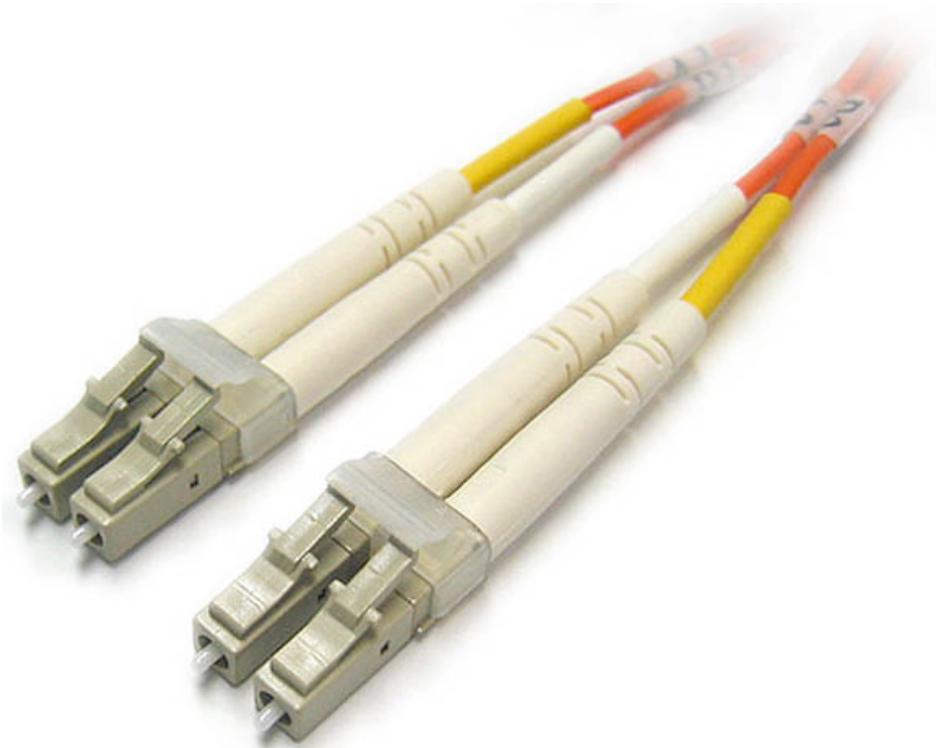
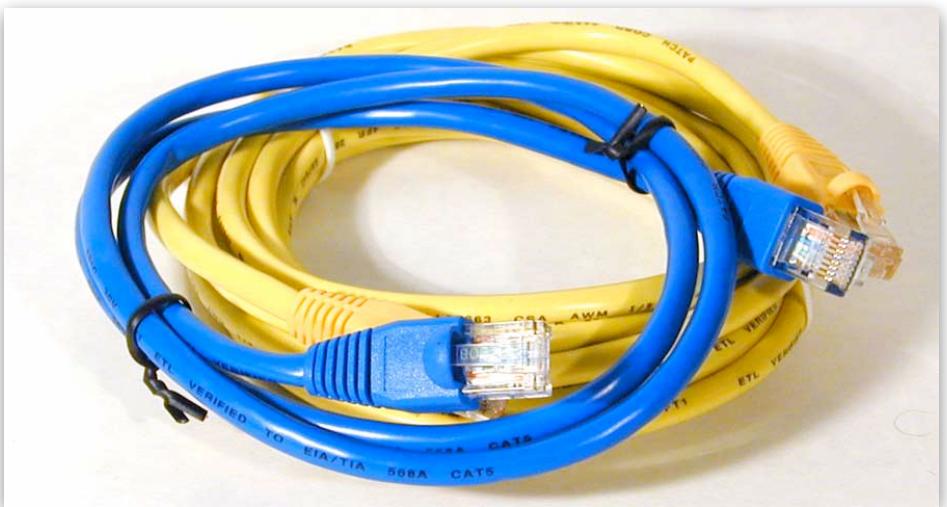


Source: Wikipedia

Ralph Hartley  
(1888-1970)

# The Data Link Layer

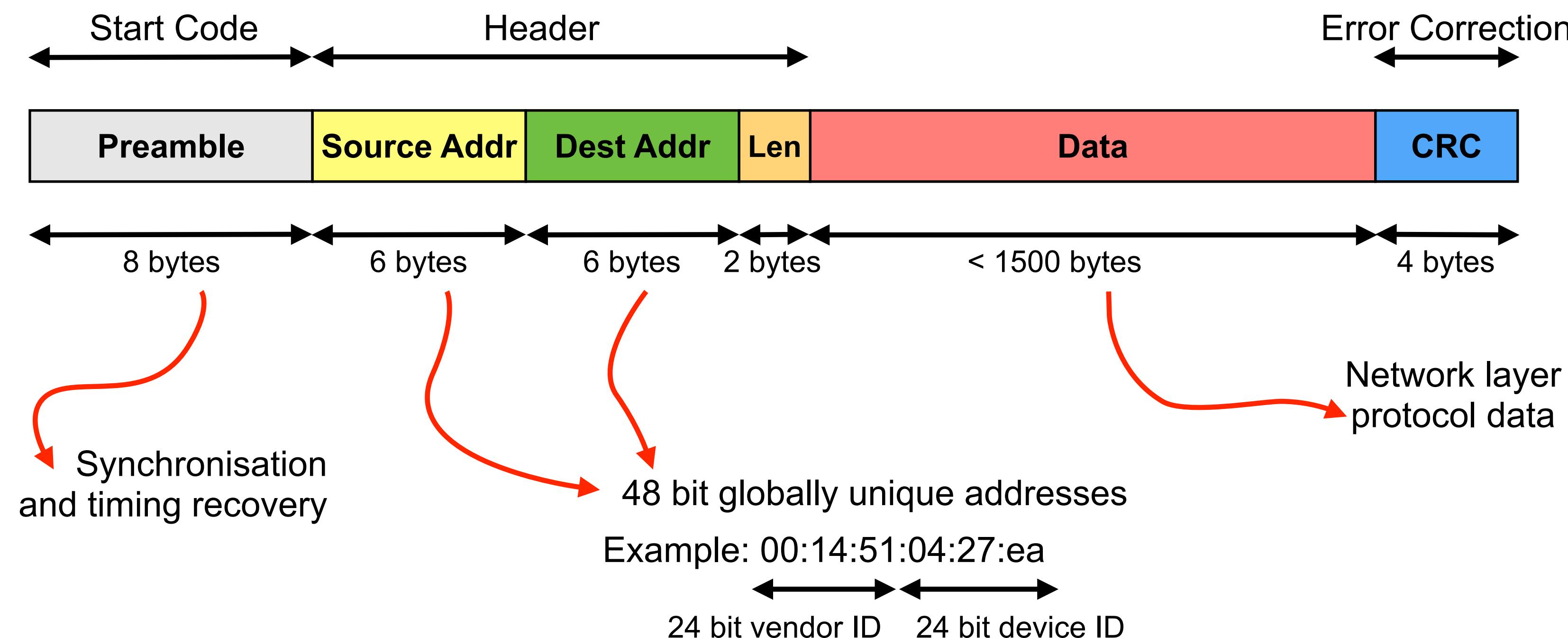
- Physical layer enables communication
- Data link layer provides framing, addressing, media access control
  - Structure the bitstream into meaningful frames of data
  - Detect and correct transmission errors
  - Identify devices
  - Arbitrate access to the channel



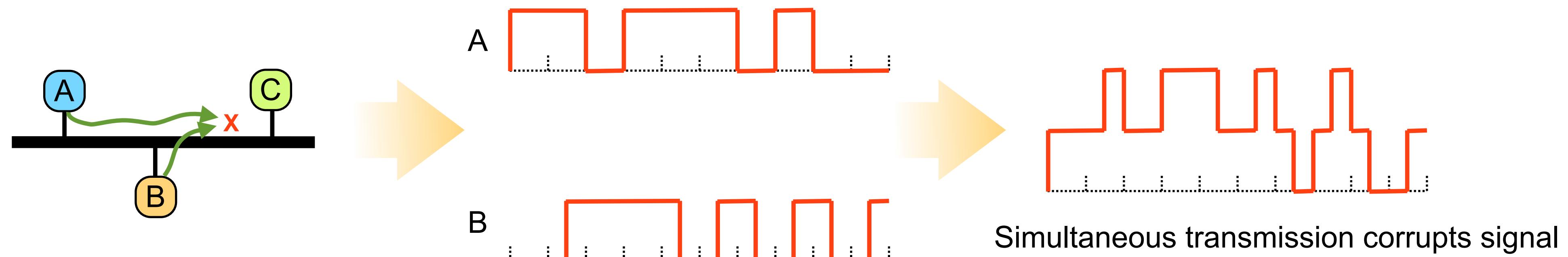
# Framing and Addressing

Separate the bitstream into meaningful frames of data

Example: Ethernet frame format



# Media Access Control

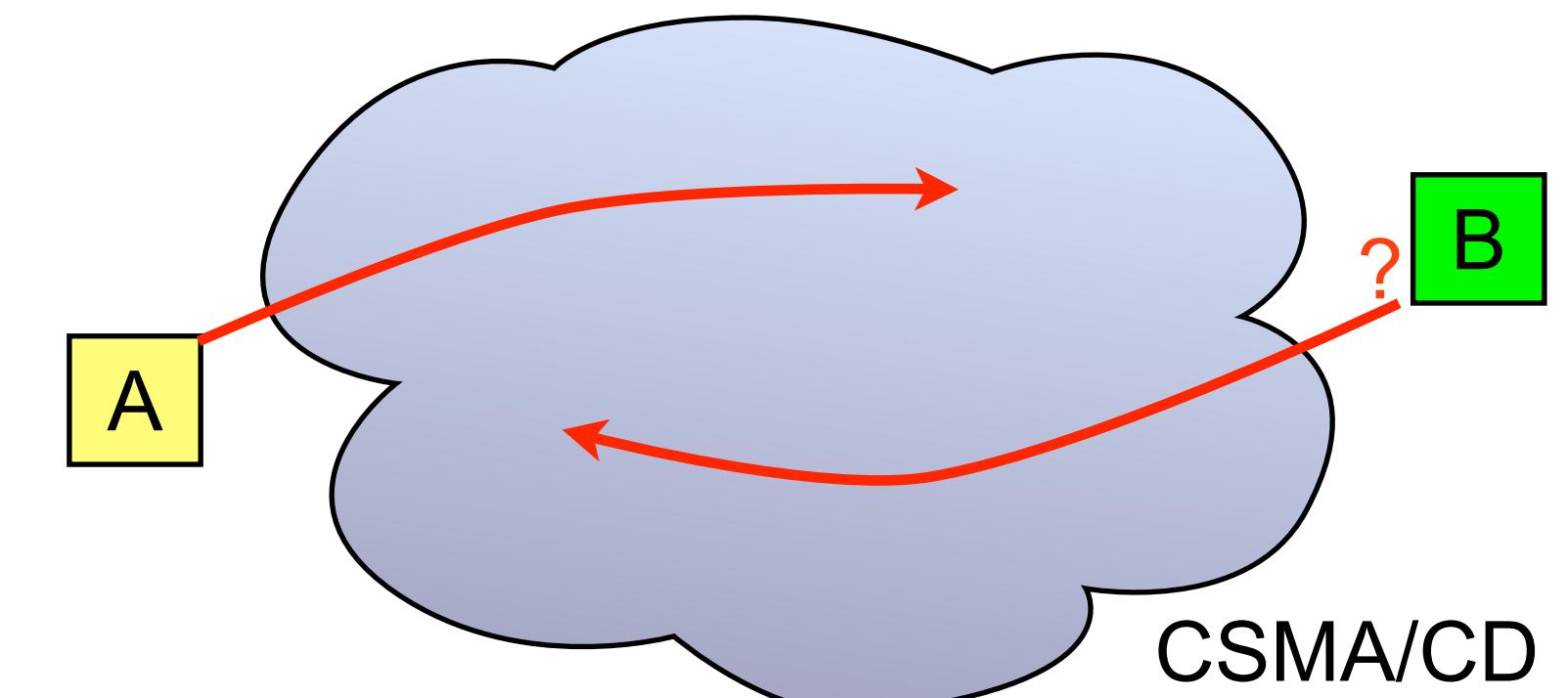


**Control when frames are sent, to avoid collisions**

# Media Access Control

- When propagation delay low, listen before sending
  - If link is idle, send data immediately
  - If another transmission is active, or if collision occurs, stop sending, wait, then retransmit
    - Wait time should be random – to avoid deterministic repeated collisions; pick a random initial back-off interval of  $x$  seconds  $\pm 50\%$
    - Wait time should increase with number of collisions – repeated collisions signal congestion; reduce transmission rate allows network to recover; each repeated collision before success,  $x \rightarrow 2x$
- Improves utilisation
  - Active transmissions not disrupted by collisions
  - Only the new sender backs-off if the channel is active
- Examples: Ethernet, Wi-Fi

Why does propagation delay matter?



CSMA/CD

A starts transmitting

B listens, hears no traffic (message from A hasn't reached it yet)

B starts transmitting

Collision occurs, as messages overlap in transit; smaller propagation delay → less likely to occur

# Physical and Data Link Layers

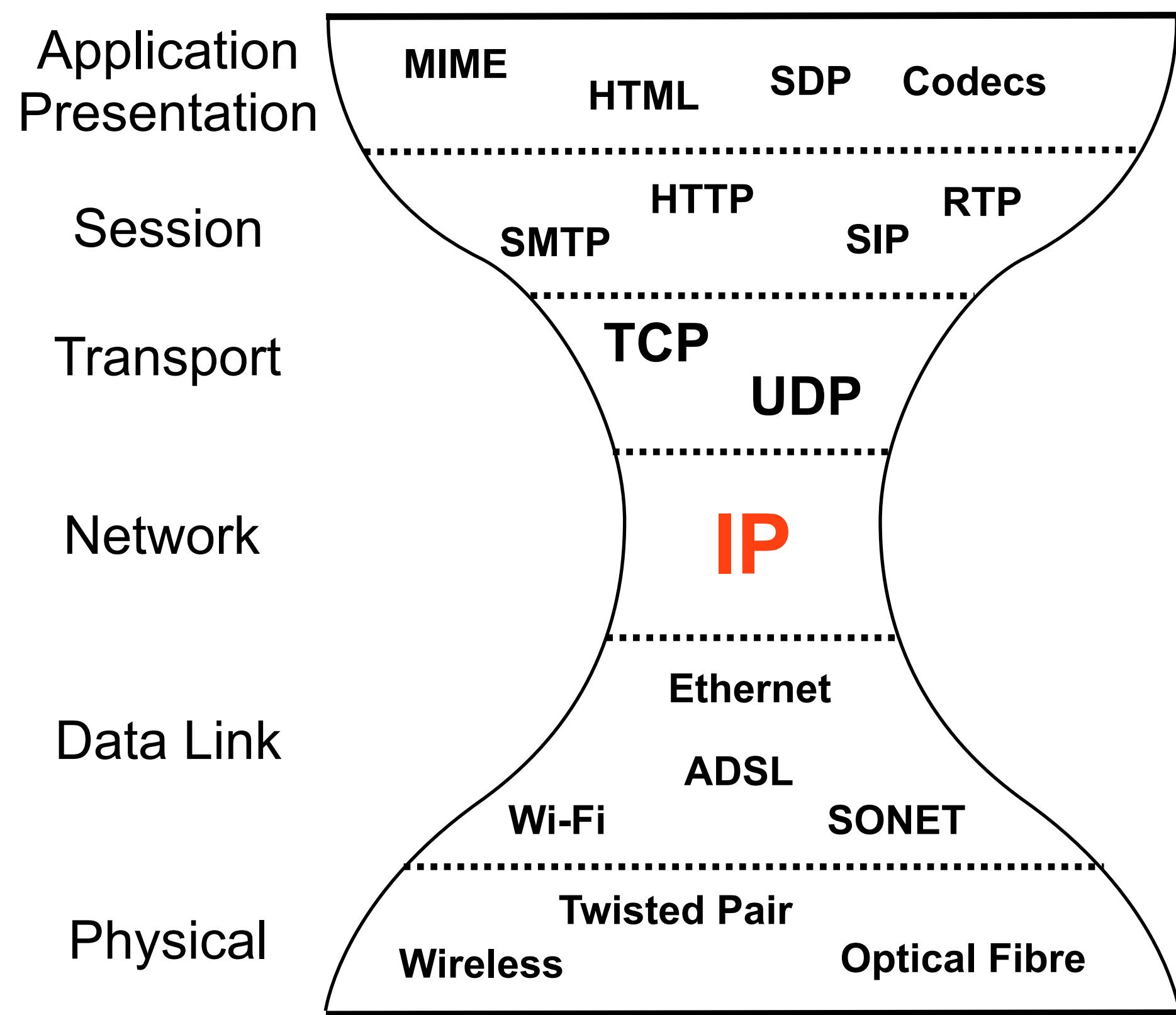
- Physical link characteristics, modulation and transmission
- Data link layer: framing and addressing, media access control

# The Network Layer and the Internet Protocols

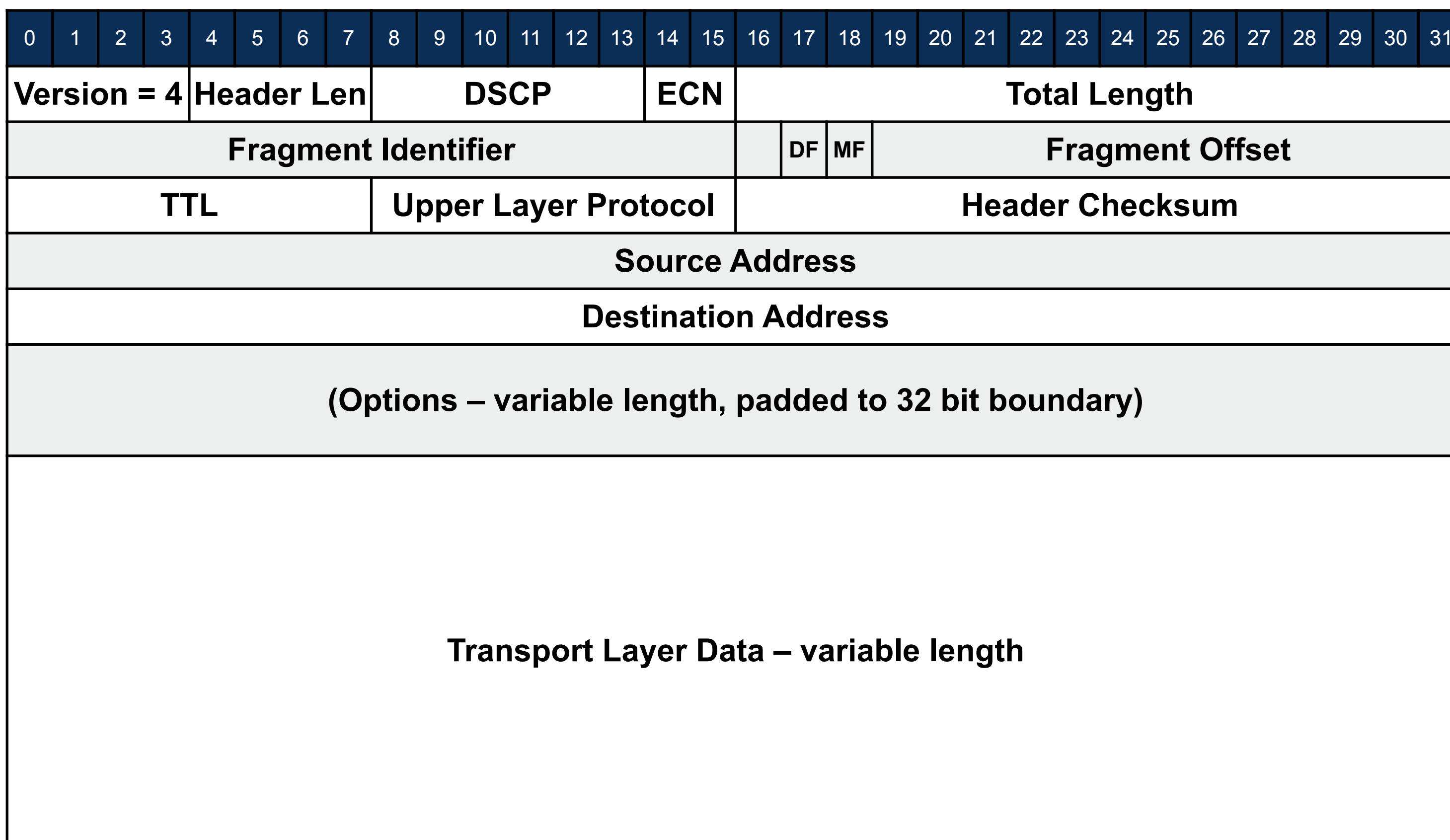
- Network Layer Concepts
  - Addressing
  - Routing
  - Forwarding
- The Internet Protocols: IPv4 and IPv6

# The Network Layer as an Internet Protocol

- Global inter-networking protocol
- Hour glass protocol stack
  - Single standard network layer protocol (IP)
    - Packet switched network, best effort service
    - Uniform network and host addressing
    - Uniform end-to-end connectivity – subject to firewall policy
  - Many transport & application layer protocols
  - Range of link-layer technologies supported
  - Decouples end-to-end functionality from per-hop functionality

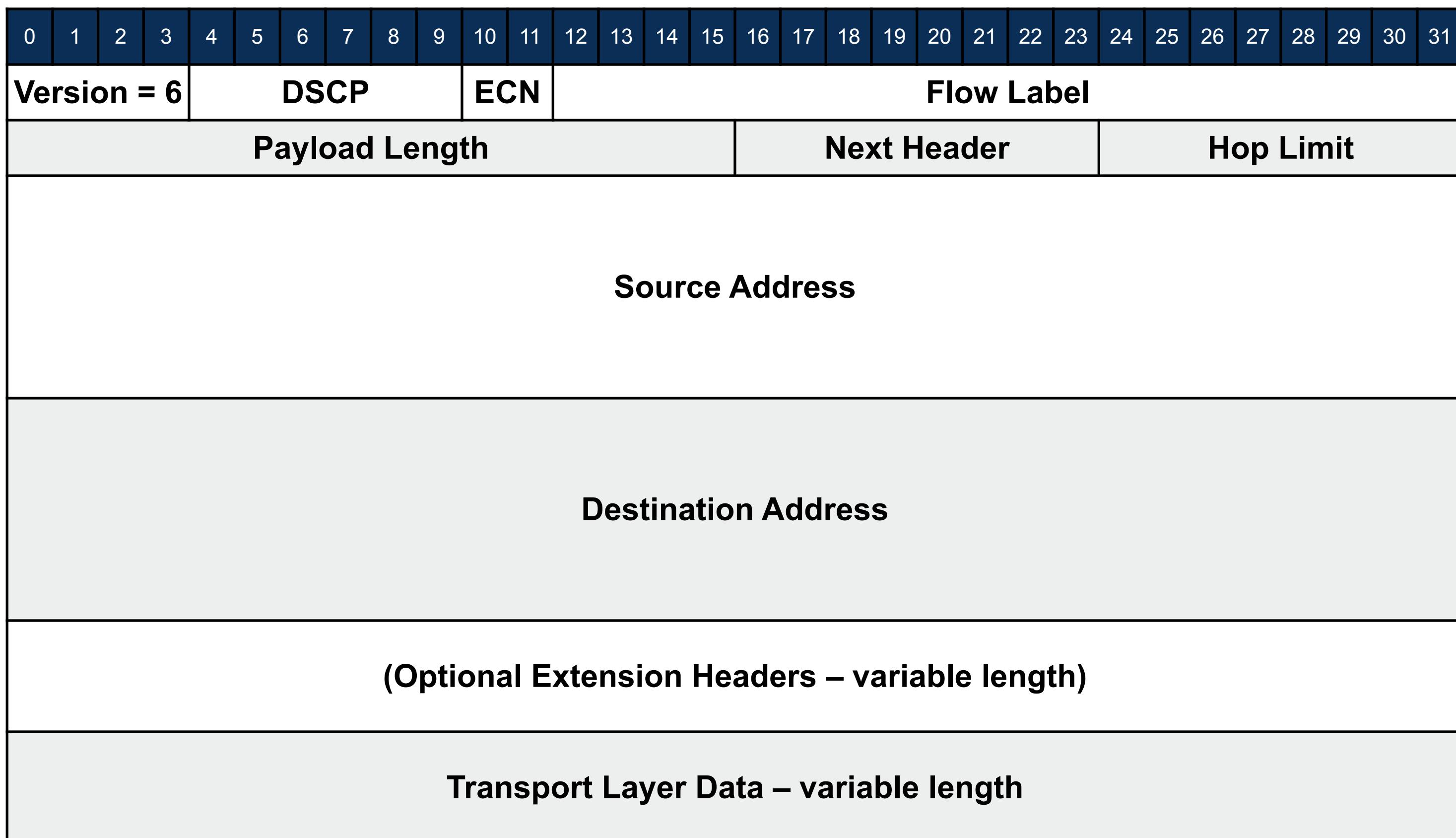


# IPv4



- 32 bit addresses insufficient
- Fragmentation difficult at high data rates
- Limited extensibility

# IPv6



- Larger address space
- No in-network fragmentation
- No unnecessary checksum
- Simpler header format

# What About IPv5?

- Experiments with voice over the ARPAnet – the precursor to the Internet – started in the early 1970s
  - Network Voice Protocol <https://www.ietf.org/rfc/rfc741.txt>
- This evolved into the Internet Stream Protocol, ST-II, that was assigned IPv5 – an experimental multimedia streaming protocol developed between 1979 and 1995, but never widely deployed
  - ST-II+ specification: <http://www.ietf.org/rfc/rfc1819.txt>

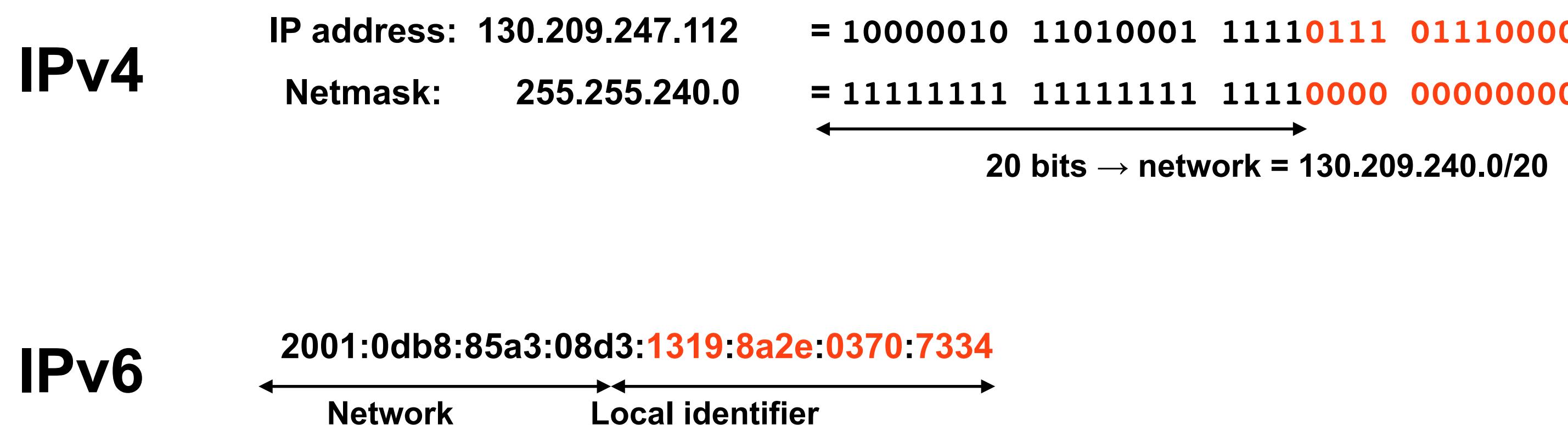


Danny Cohen, developer of the Network Voice Protocol

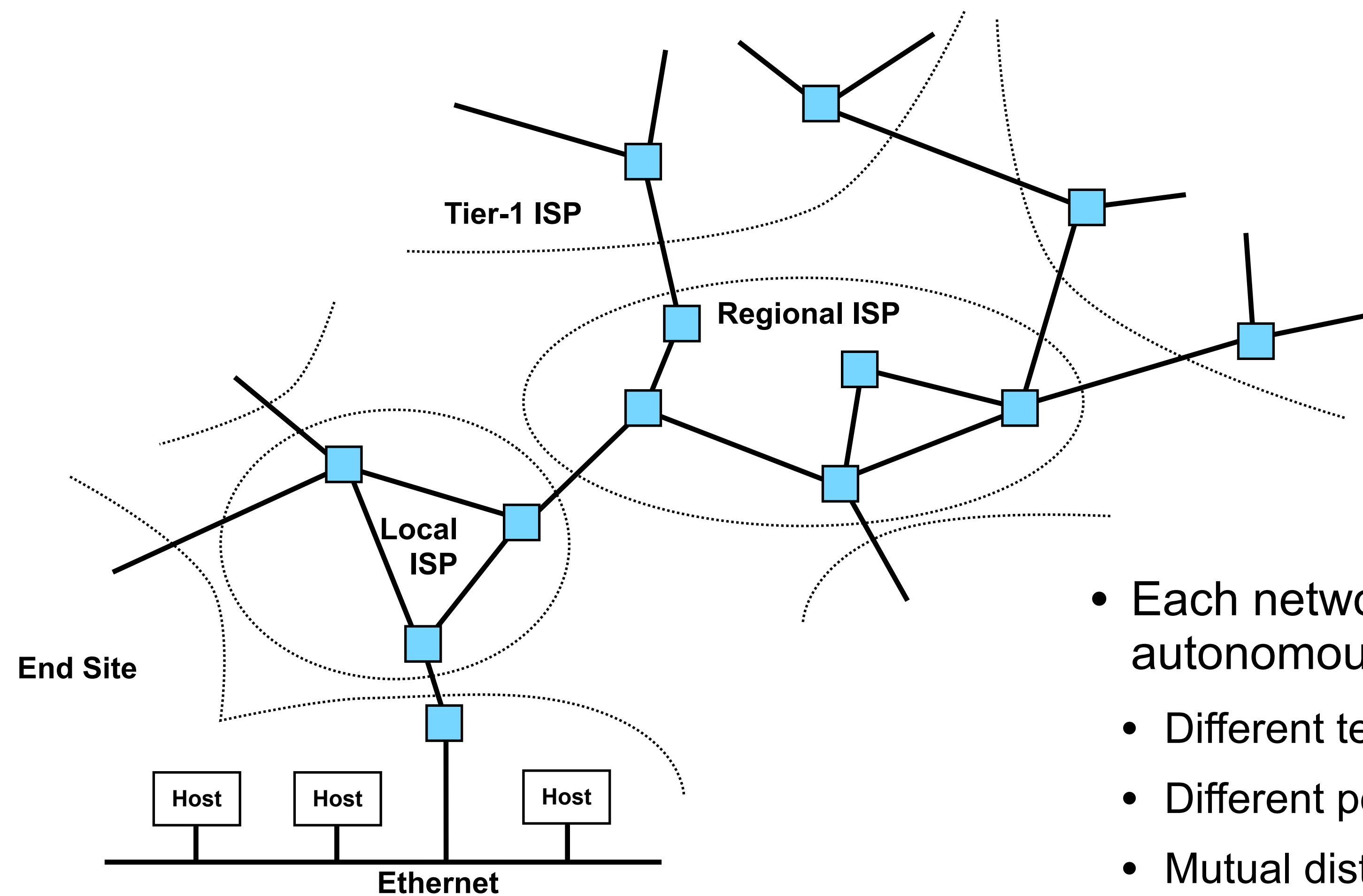
Aside: Danny Cohen also wrote “On Holy Wars and a Plea for Peace” (<https://www.ietf.org/rfc/ien/ien137.txt>) that introduced the terms big- and little-endian for numeric data and message byte order

# IP Addressing

- IP addresses encode location of network interface
  - If a host has multiple network interfaces (e.g., Wi-Fi and Ethernet), it will have multiple IP addresses
  - A host can support both IPv4 and IPv6 on each interface
  - A host can have multiple IP addresses of each type assigned to each interface
- DNS names are an application concept – not used by the network



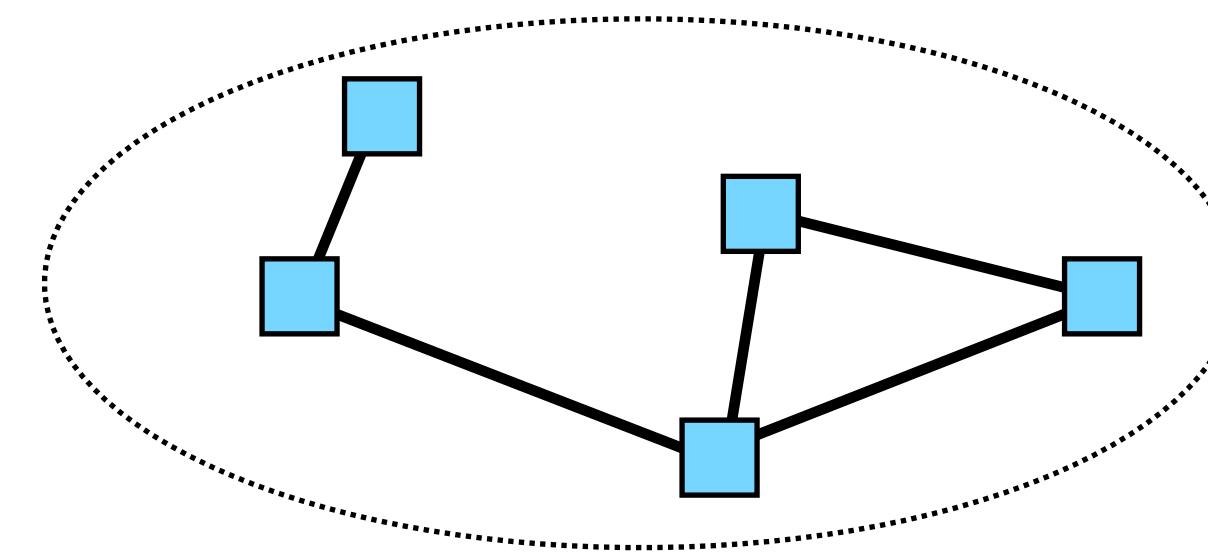
# Routing



- Each network administered separately - an autonomous system (AS)
  - Different technologies
  - Different policies
  - Mutual distrust – between AS and its peers; between AS and its customers

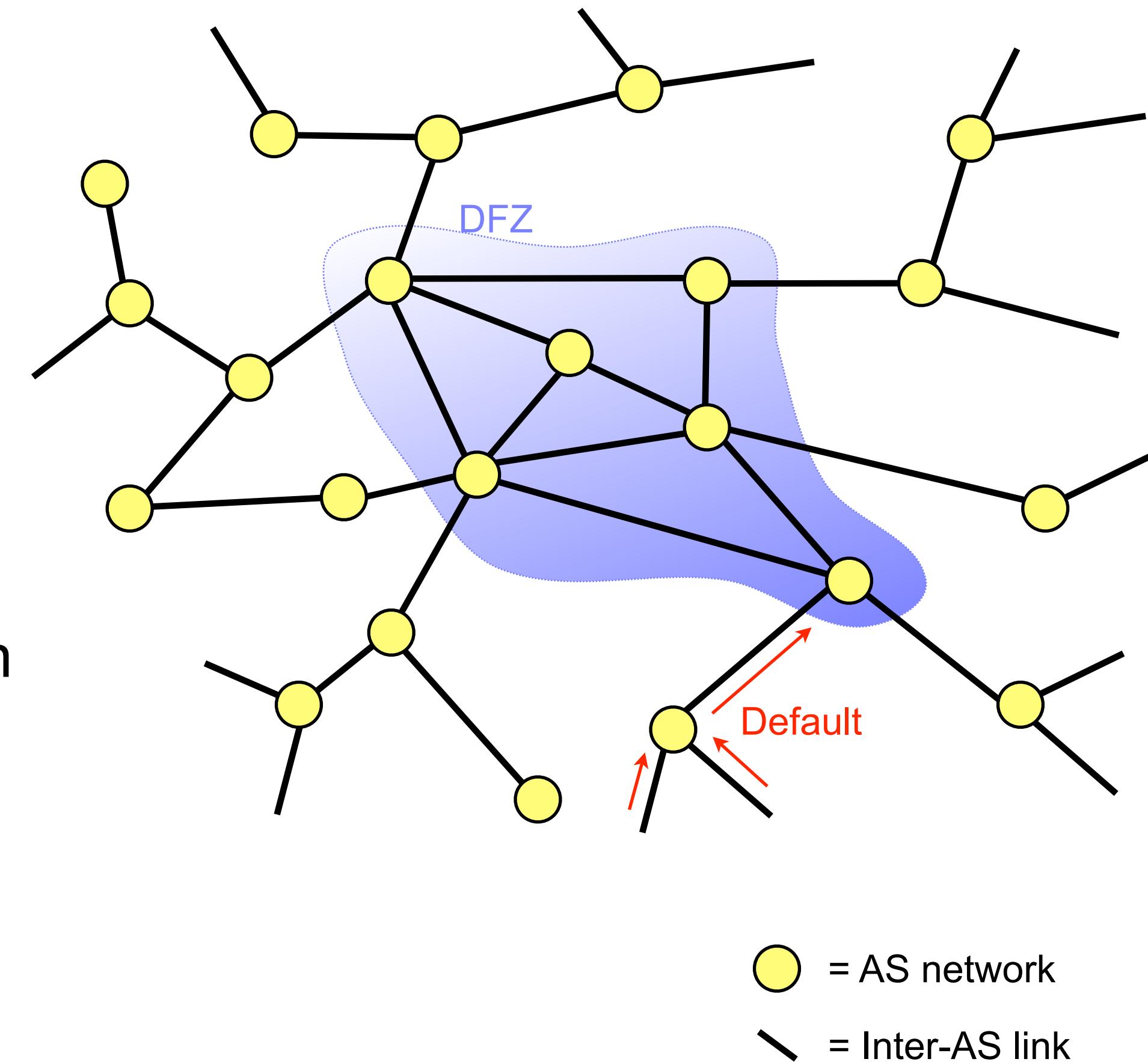
# Autonomous Systems and Inter-domain Routing

- Each network administered separately – an autonomous system (AS)
  - Separately administered; different technologies
  - Shortest path routing
  - Distance vector or link state algorithms

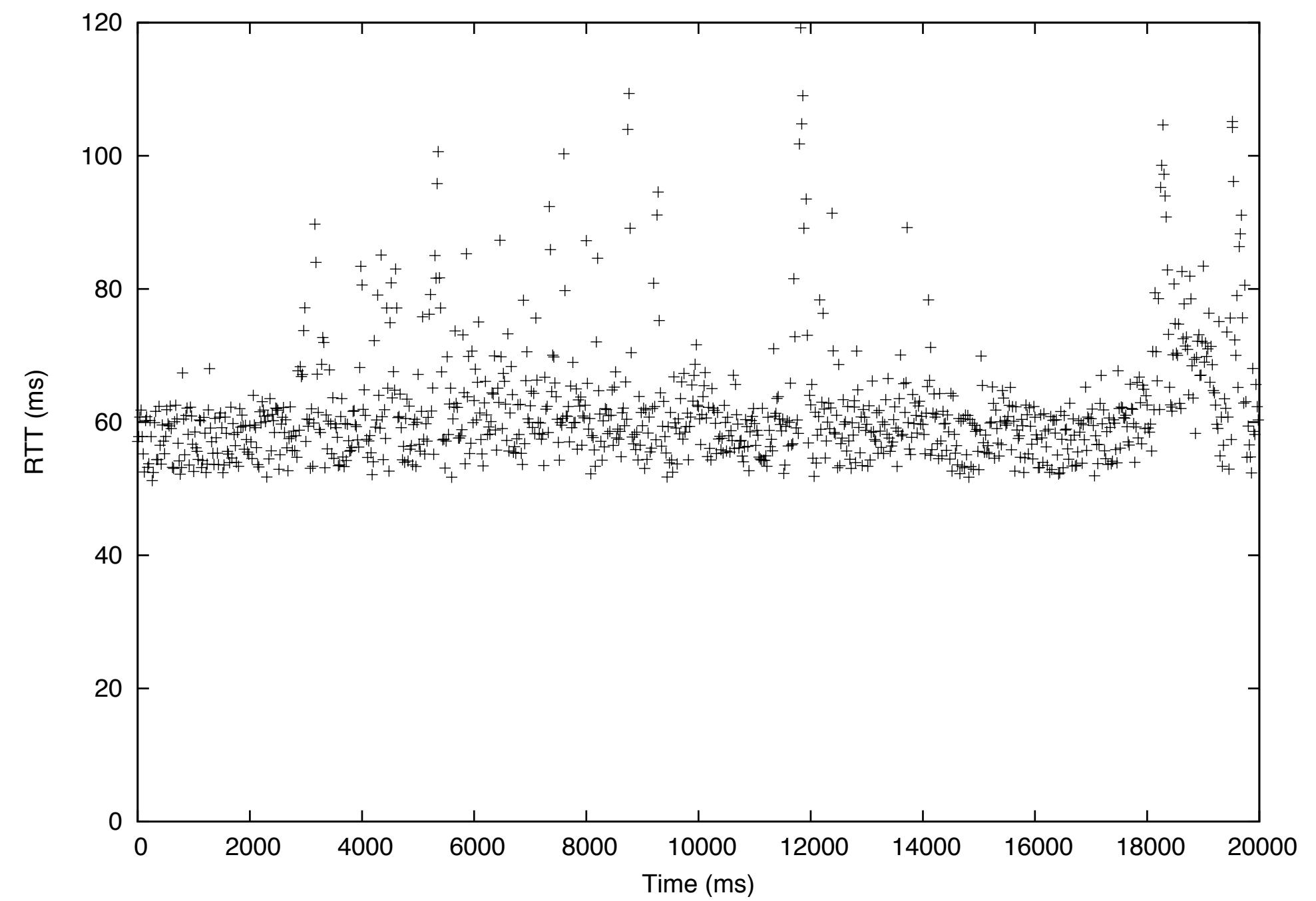


# Inter-domain Routing and BGP

- Treats each network as a graph node and routes between ASes
- The AS-level topology:
  - Well connected core; sparse edges
  - Edge networks can use default route to the core
  - Core networks need full routing table: the default free zone (DFZ)
- Routing between competitors – no real trust between organisations
  - Policy, politics, and economics are key constraints; not shortest path
  - BGP routing protocol



# Forwarding



- Best effort, connectionless, packet delivery
  - Just send – no need to setup a connection first
  - Network makes its best effort to deliver packets, but provides no guarantees
  - Time taken to transit the network may vary
  - Packets may be lost, delayed, reordered, duplicated or corrupted
  - The network discards packets it can't deliver
- Easily run over any type of link layer

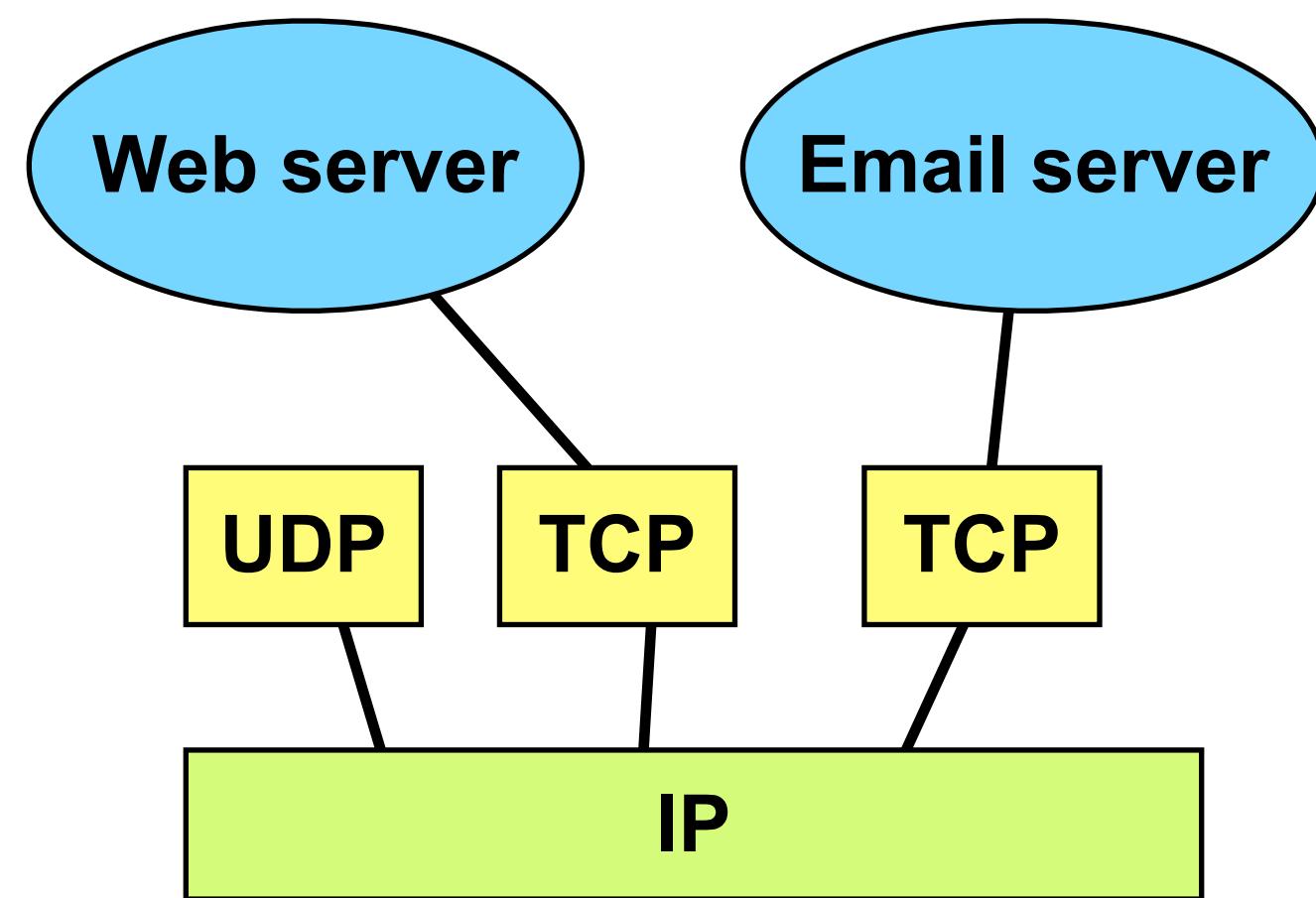
# The Network Layer and the Internet Protocols

- Network Layer Concepts
- The Internet Protocols: IPv4 and IPv6
- Addressing
- Routing
- Service model: best effort transport

# The Transport Layer

- Concepts
- UDP
- TCP
- Congestion control

# TCP and UDP Transport

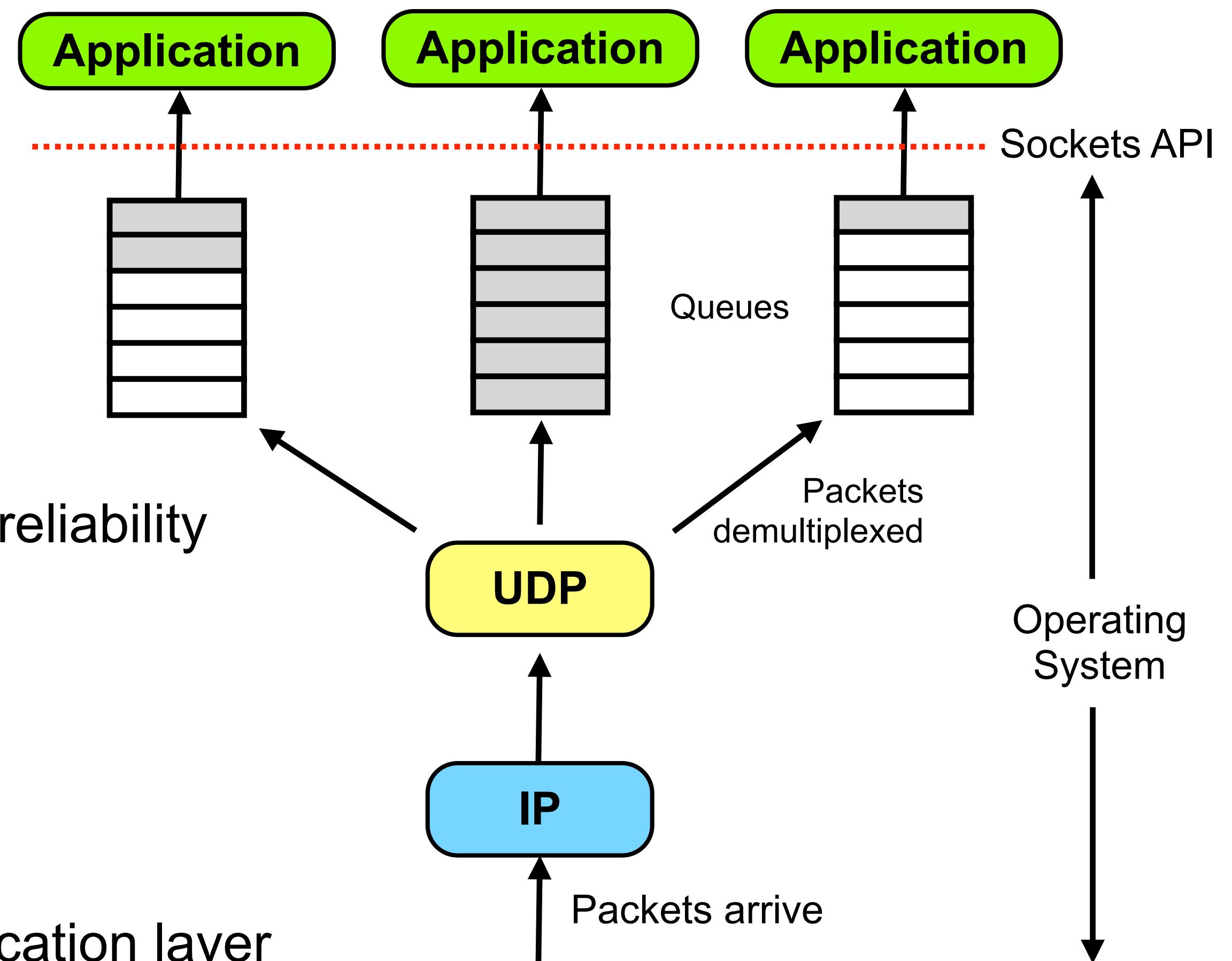


- IP network provides best effort service
  - Packets can be lost, duplicated, delayed, or re-ordered
- Transport isolates applications from the network
  - Demultiplexes traffic for different applications
  - Enhances network quality of service to offer appropriate reliability
  - Performs congestion control, adapts to network capacity

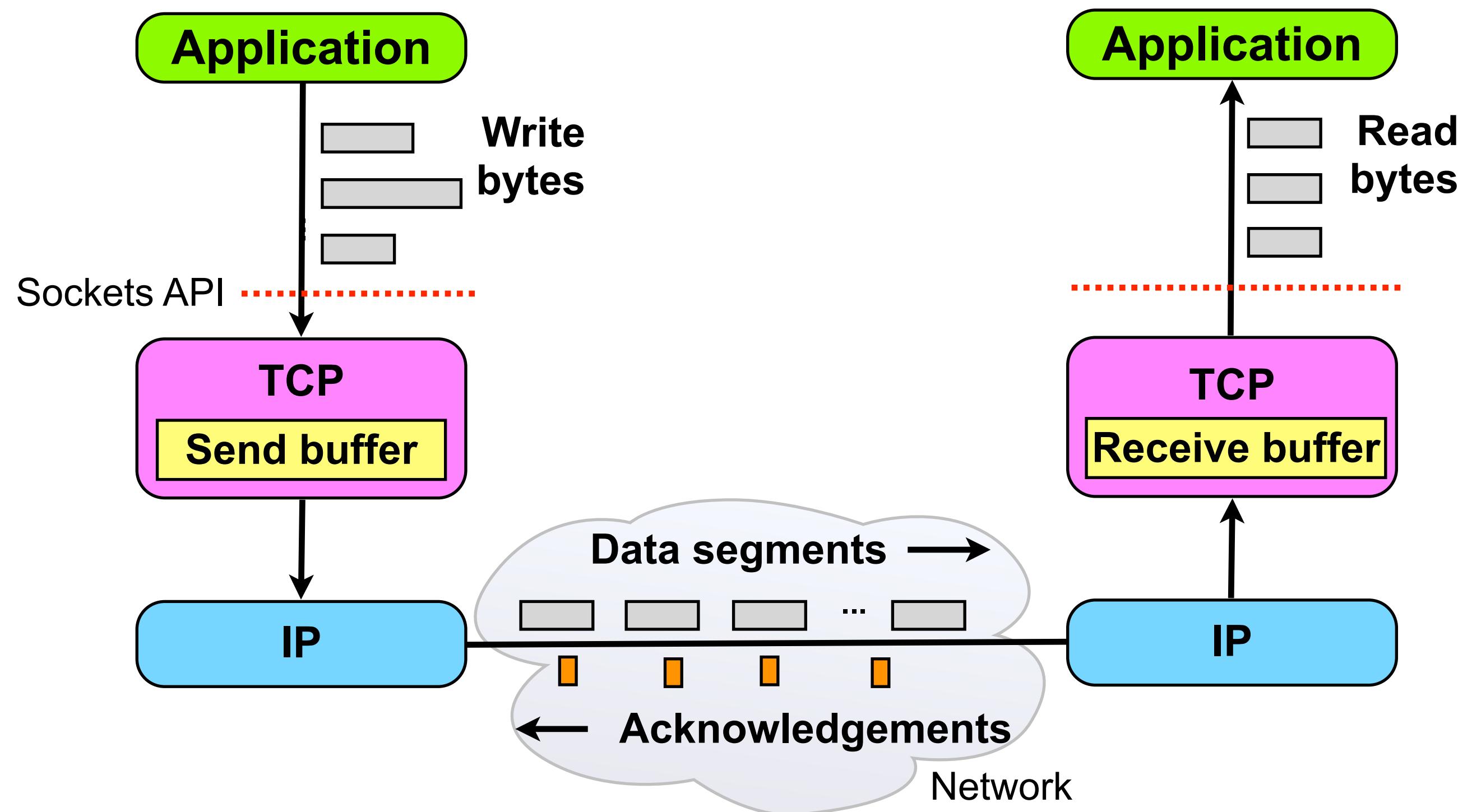
- Only two deployable transport protocols in the Internet:
  - **UDP** - user datagram protocol
  - **TCP** - transmission control protocol

# UDP

- Simplest transport protocol
- Exposes raw IP service to applications
  - Connectionless, best effort packet delivery: framed, but unreliable
  - No congestion control
  - Adds 16 bit port number to identify services
- Used by applications preferring timeliness over reliability
  - Voice-over-IP
  - Streaming video
  - Gaming
- Must be able to tolerate some loss of data
- Must be able to adapt to congestion in the application layer

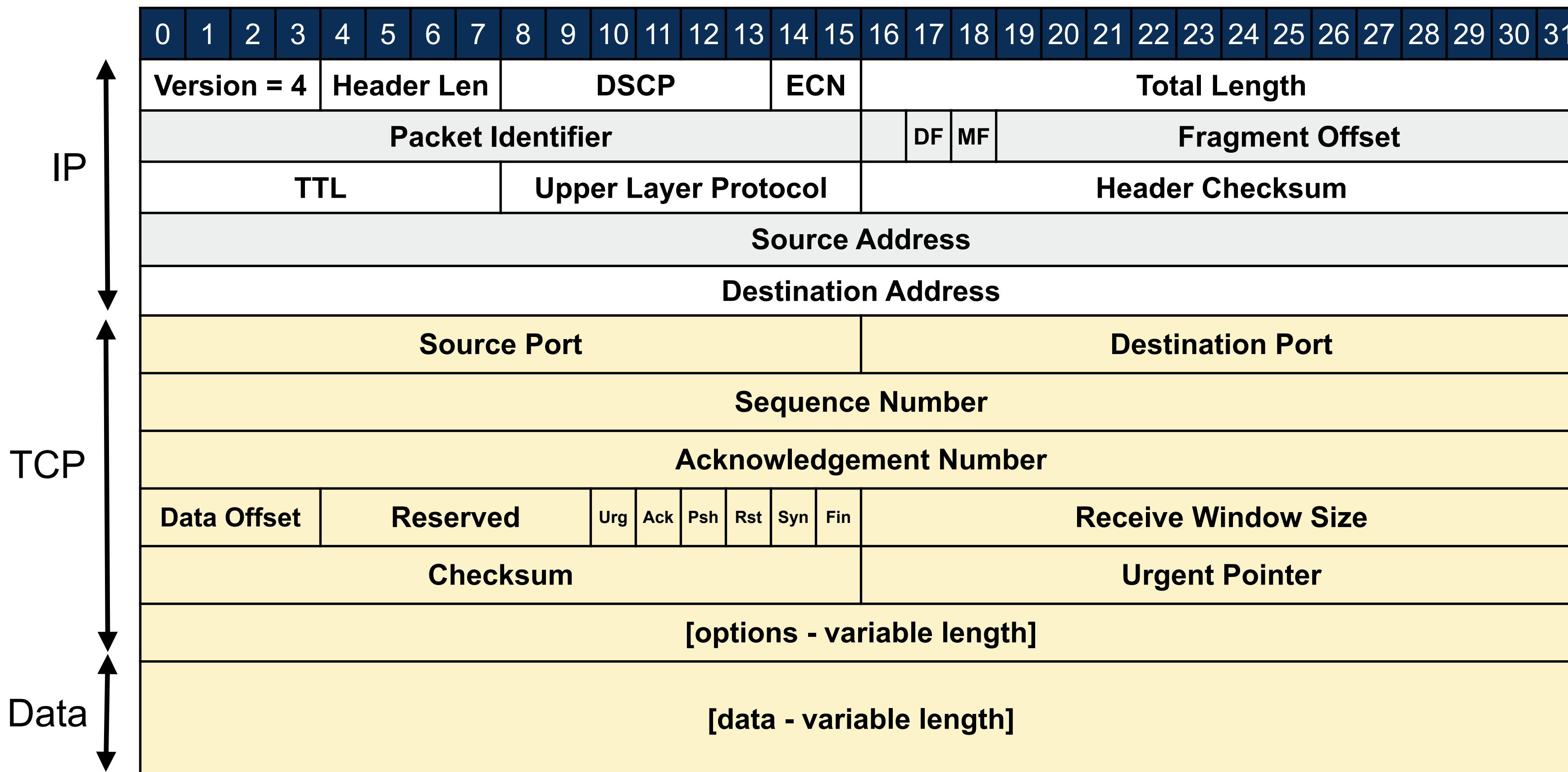


# TCP

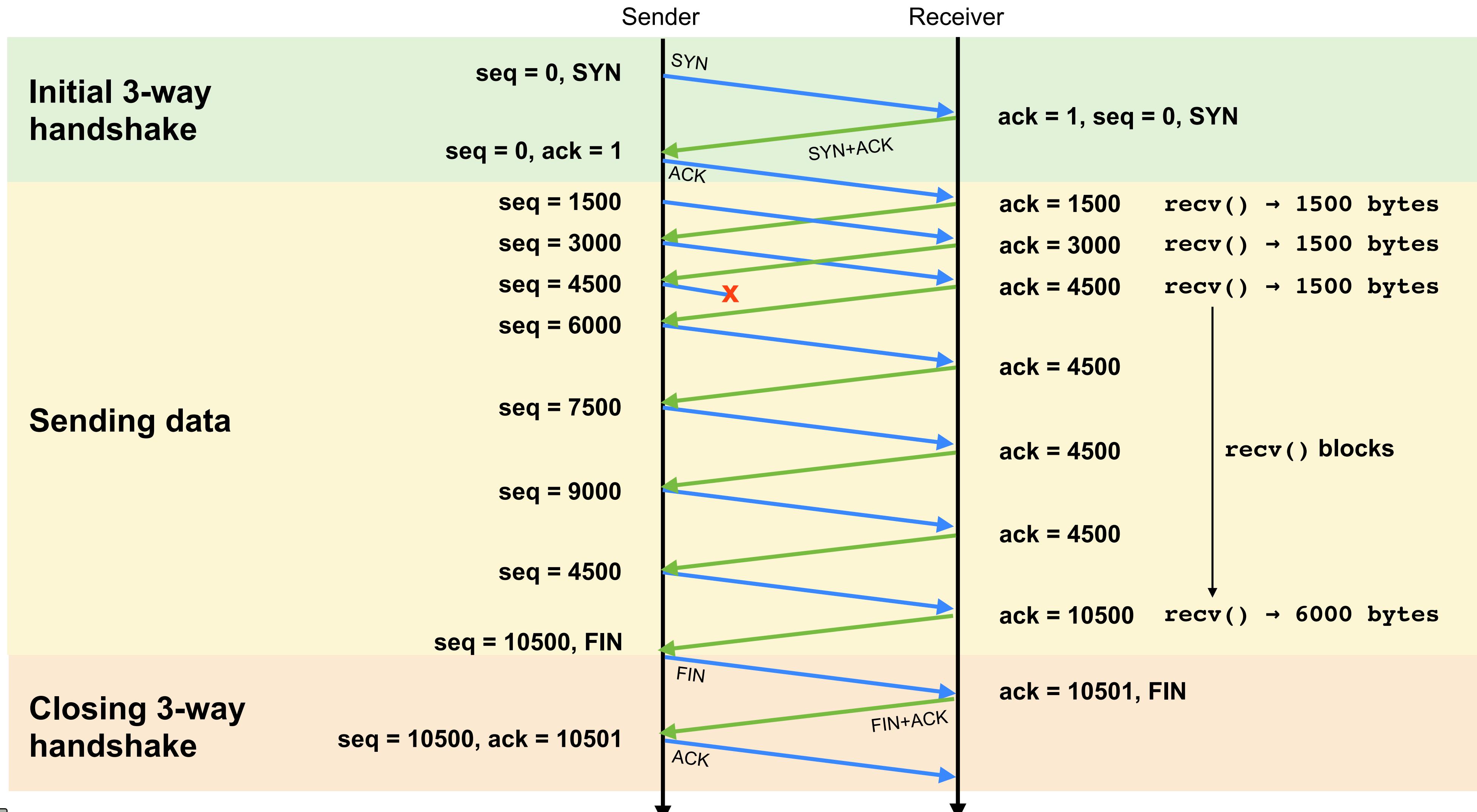


- Reliable, ordered, byte stream delivery service running over IP
  - Lost packets are retransmitted; ordering is preserved; message boundaries **are not** preserved
  - Adapts sending rate to match network capacity → congestion control
  - Adds port number to identify services
- Used by applications needing reliability → default choice for most applications

# TCP Packet Format

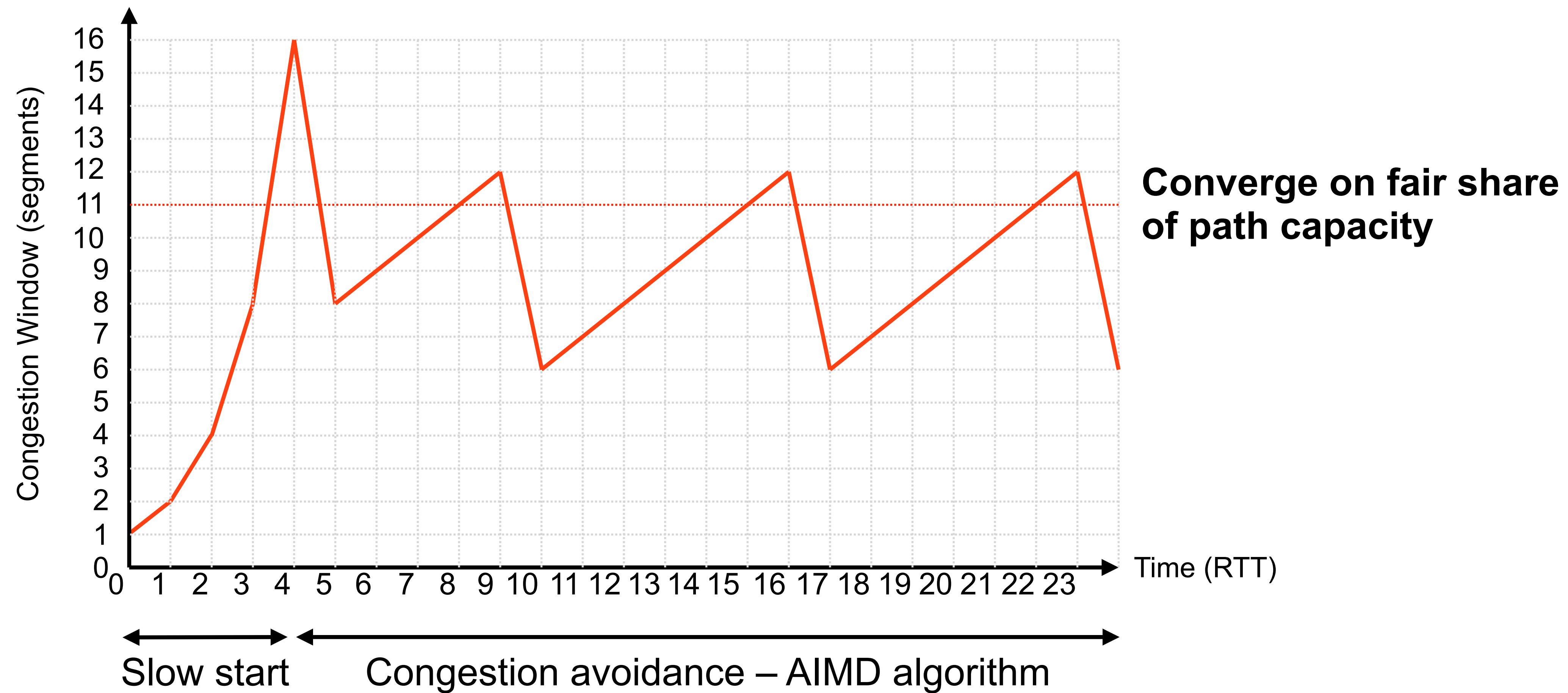


# TCP Connection Timeline (Example)



# TCP Congestion Control

Typical evolution of TCP window, assuming  $W_{init} = 1$



# The Transport Layer

- Concepts
- UDP
- TCP
- Congestion control

# Higher Layer Protocols

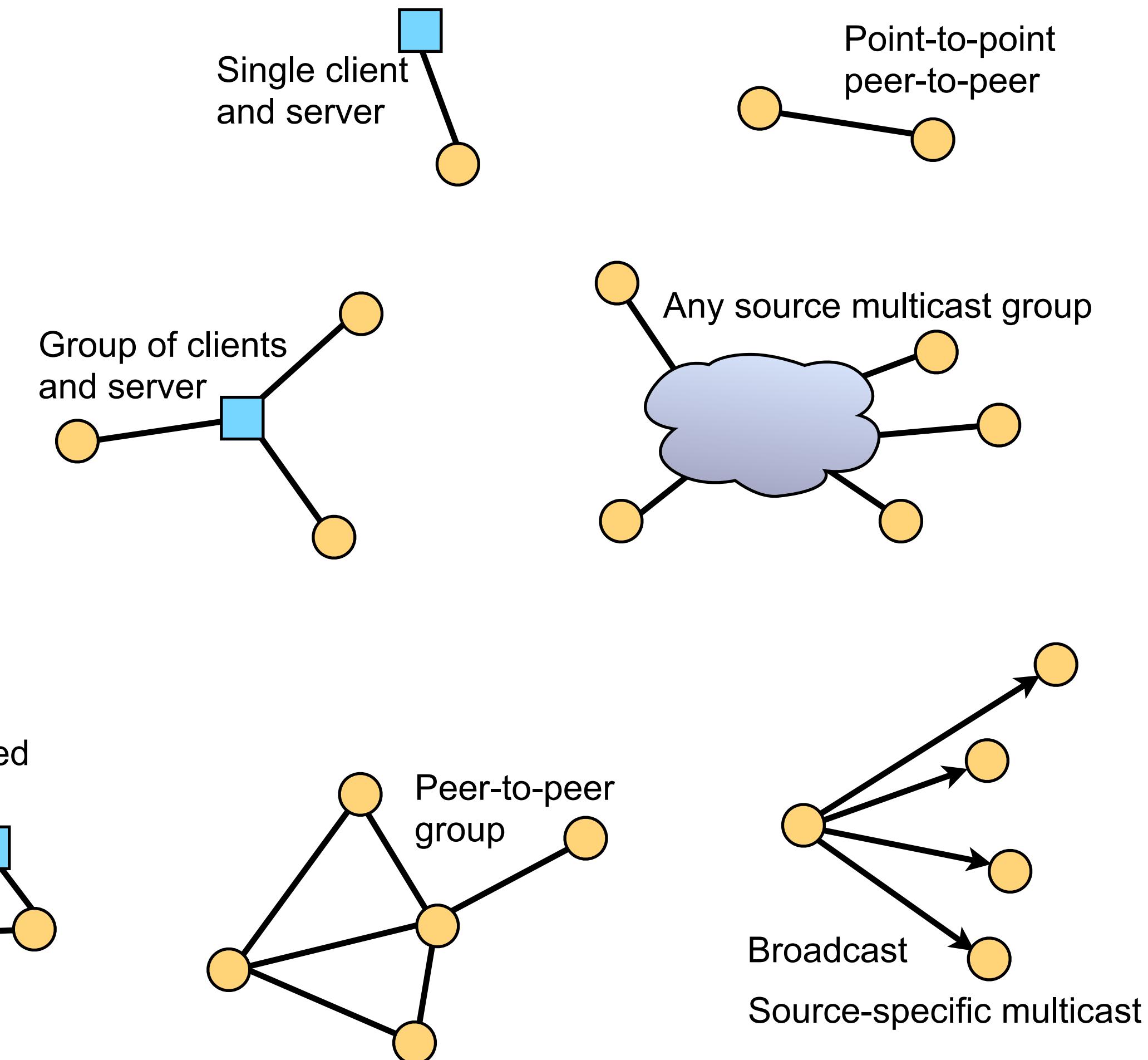
- Session Layer
- Presentation Layer
- Application Layer
- The importance of standards

# Higher Layer Protocols

- The OSI reference model defines three layers above the transport:
  - Session layer
  - Presentation layer
  - Application layer
- Internet architecture makes no clear distinction between these layers
- Goal – support application needs:
  - Manage transport layer connections
  - Name and locate application-level resources
  - Negotiate data formats, and perform format conversion if needed
  - Present data in appropriate manner
  - Implement application-level semantics

# Session Layer: Managing Connections

- What connections does the application need?
- How to find participants?
- How to setup connections?
- How does session membership change?
  - Does the group size vary greatly?
  - How rapidly do participants join and leave?
  - Are participants aware of other members?



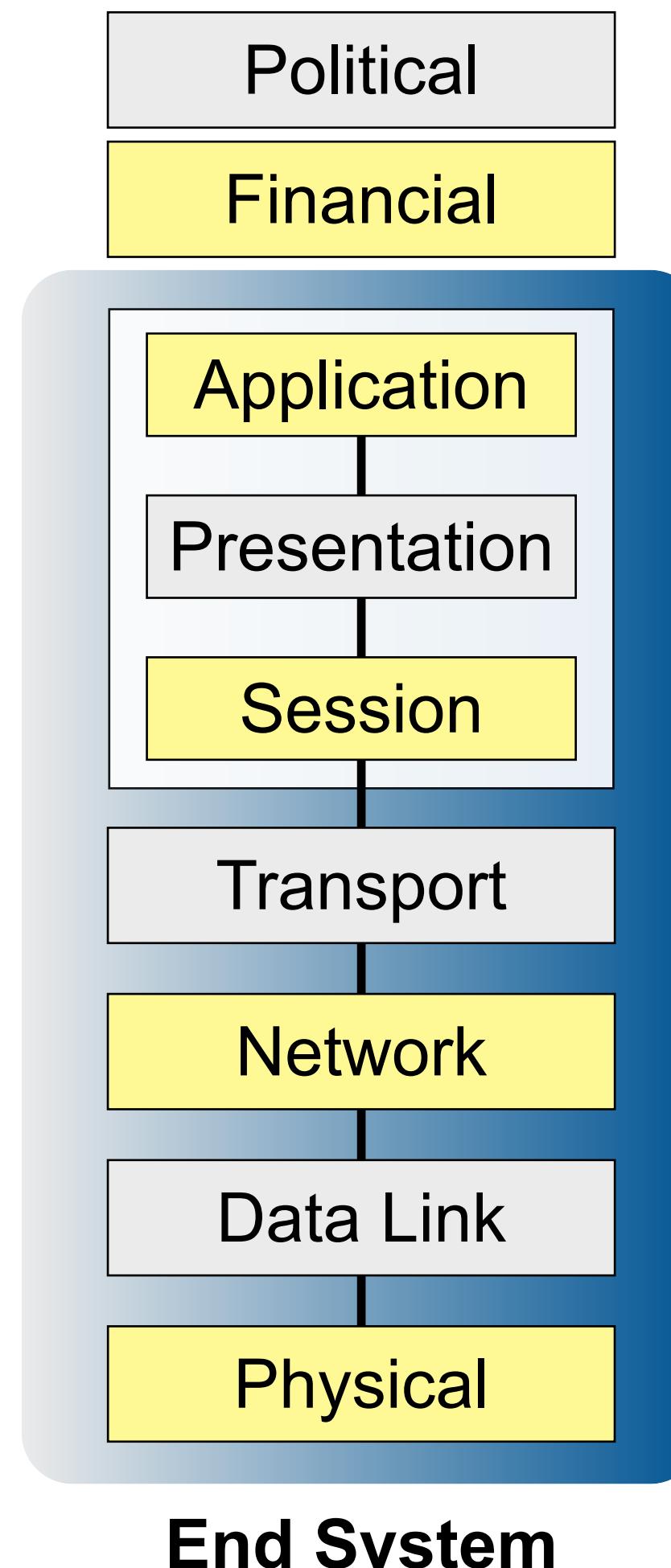
# The Presentation Layer

- Managing the presentation, representation, and conversion of data:
  - Media types and content negotiation
  - Channel encodings
  - Internationalisation, languages, and character sets
  - Common services used by many applications

# The Application Layer

- Protocol functions specific to the application logic
  - Deliver email
  - Retrieve a web page
  - Stream video
  - ...

# Protocol Standards (1/2)



- The OSI model is a reasonable way of thinking about network protocols
- But – it misses two key layers:
  - Financial
  - Political
- Successful network protocols support interoperability between different vendors – this interoperability exists because those vendors work to standardise the protocols



# Protocol Standards (2/2)



**Rough consensus and running code** → standards are the result of much discussion and negotiation

# Higher Layer Protocols

- Session Layer
- Presentation Layer
- Application Layer
- The importance of standards

# The Changing Internet

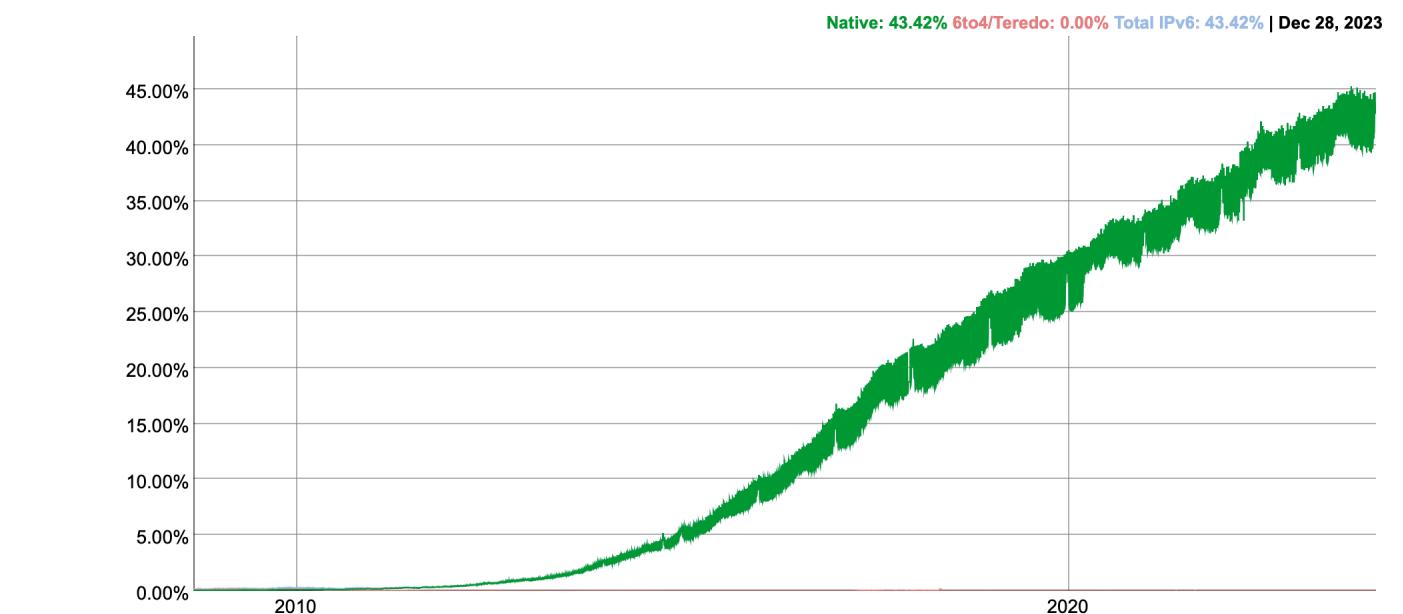
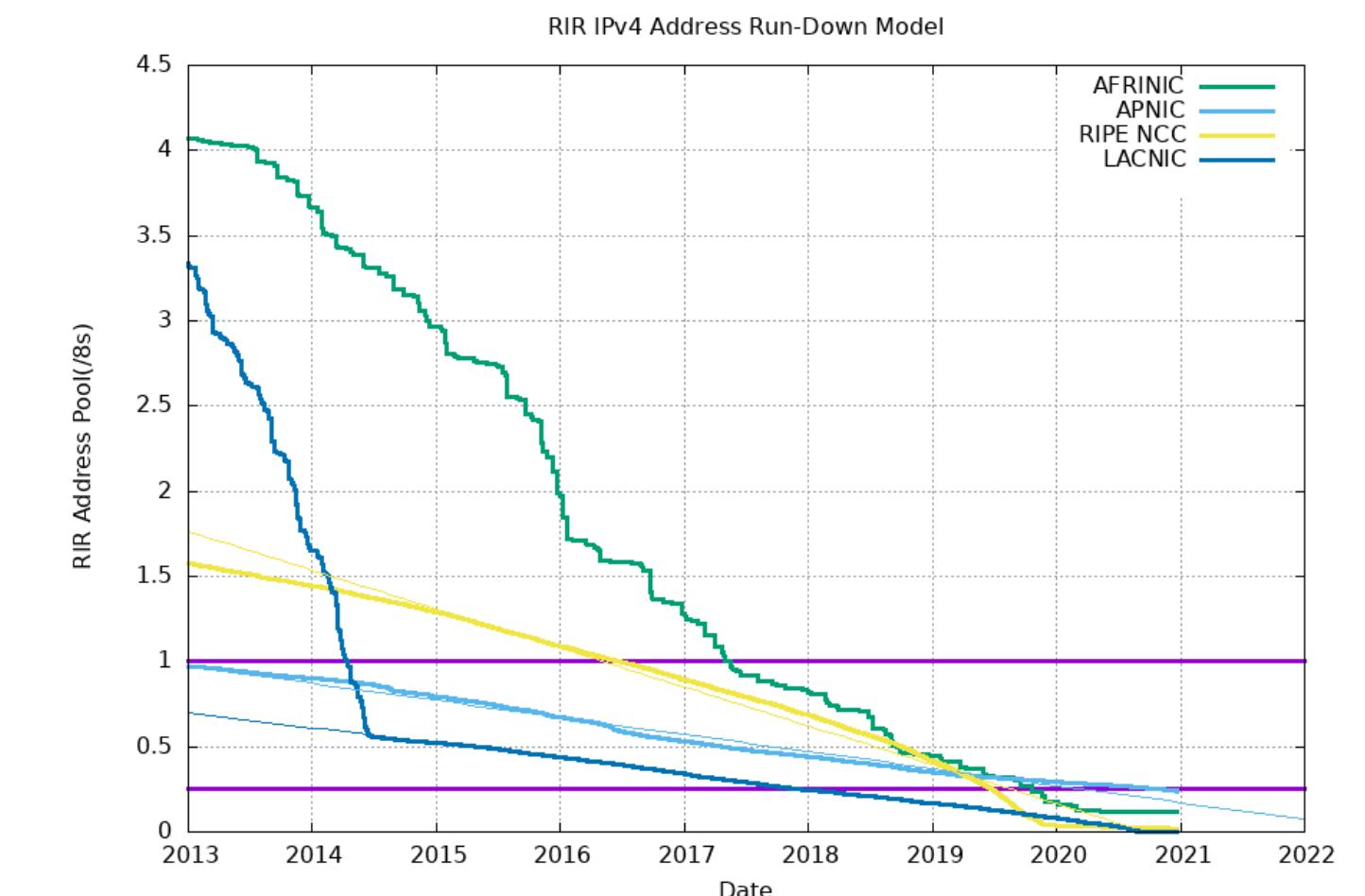
- Exhaustion of IPv4 address space
- Wireless and mobility
- Hypergiants and Centralisation
- Real-time and low-latency
- Overcoming ossification

# The Changing Internet: Architectural Assumptions

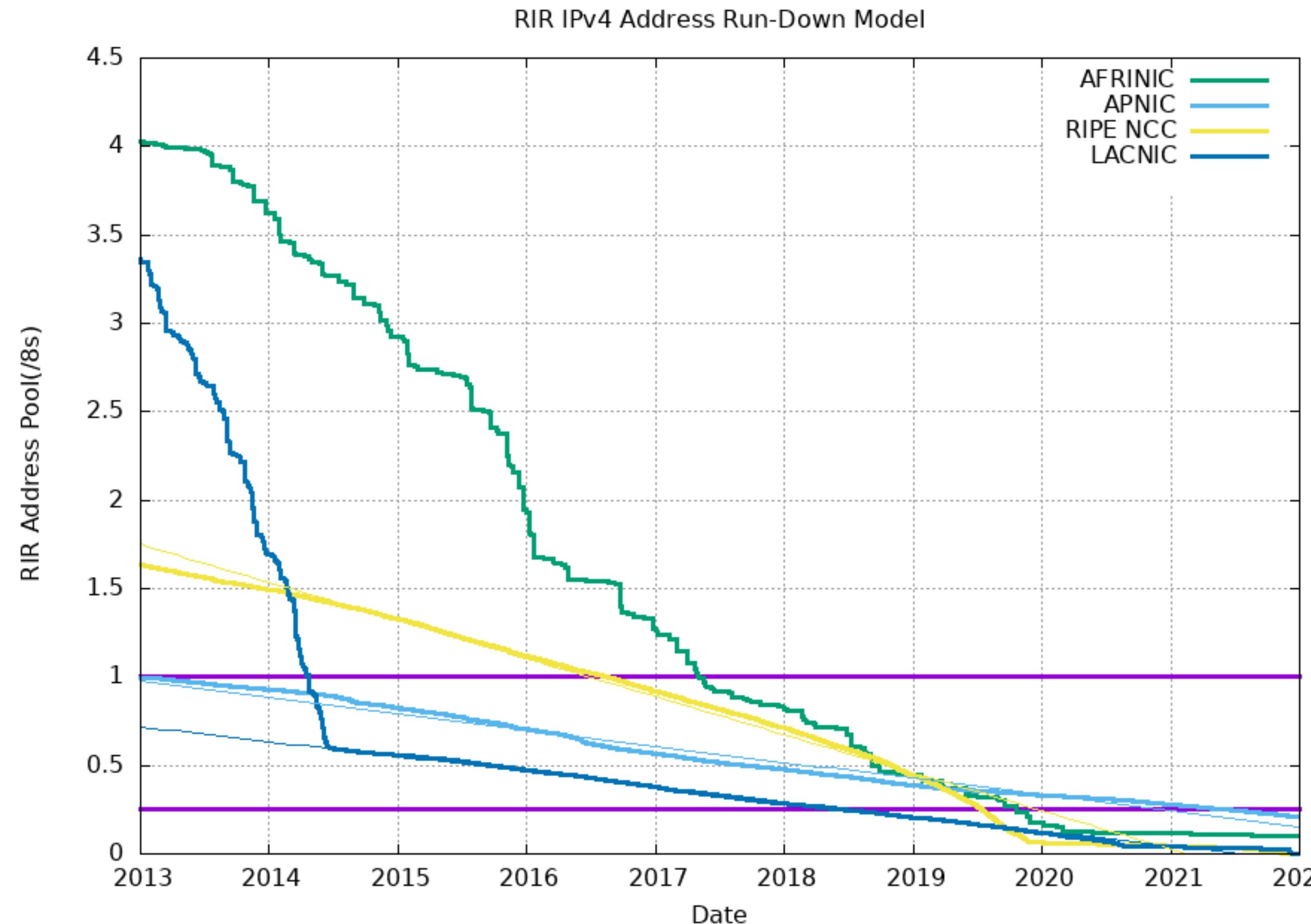
- The original design of the Internet made certain assumptions:
  - Devices are generally located in a fixed location, and have a small number of network interfaces that are uniquely addressable
  - The network and services are decentralised
  - Best effort service provides sufficient quality
  - The network is trusted
  - Innovation can happen at the edges; the network is dumb
- Reasonable for the 1980s, when the IP architecture was defined, but the network has changed – how have the protocols changed?

# Changes in Addressing and Reachability

- Assumption: every network interface has unique IP address
    - Devices uniquely addressable with uniform connectivity
    - In principle, any device can connect to any other; policy enforced by firewalls, privacy protected by changing address assignments
  - IPv4 has insufficient addresses to support this model
  - IPv6 deployment is slow; NAT is widespread
- 
- Implication: connectivity becomes difficult
    - Dual-stack IPv4/IPv6 connection racing (“happy eyeballs”)
    - NAT traversal for peer-to-peer applications
    - Complicates software design and implementation
    - Forces reliance on cloud services and encourages centralisation



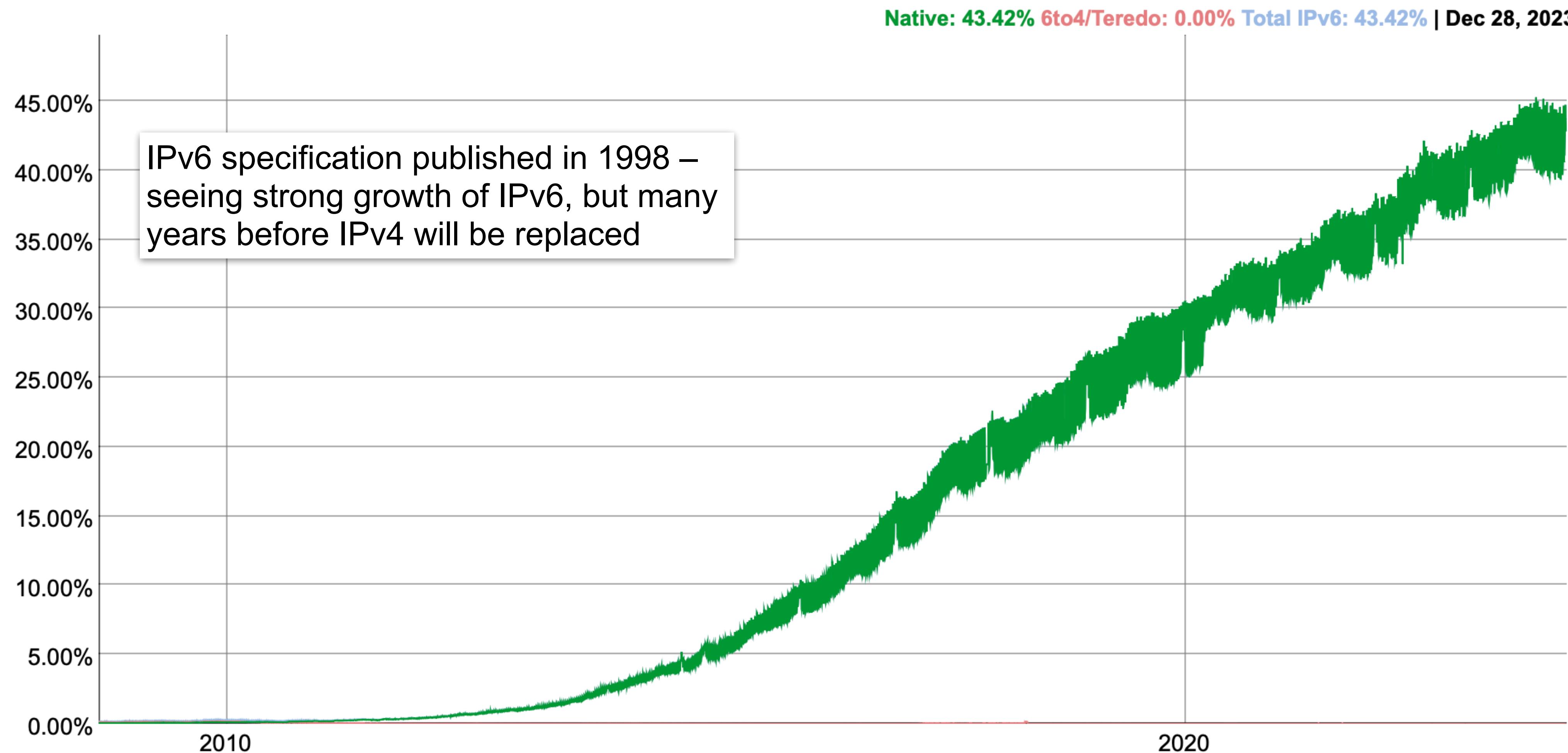
# IPv4 Address Exhaustion



- IANA assigned last IPv4 block to regional Internet registries (RIRs) in 2011
- RIRs have exhausted the available IPv4 addresses, or soon will

Source: Geoff Huston, APNIC  
December 2021 <http://ipv4.potaroo.net/>

# IPv6 Deployment is Slow

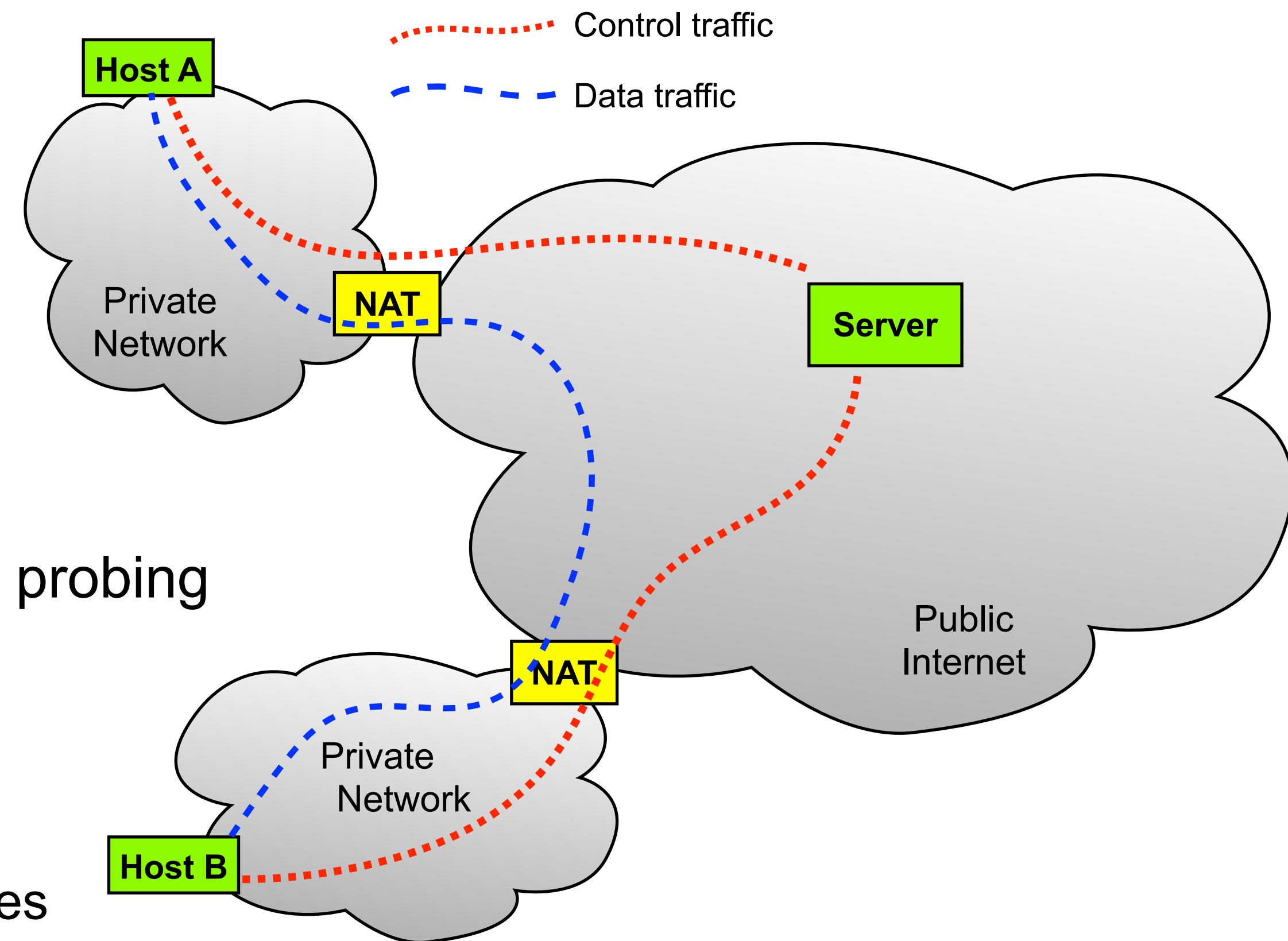


Source: Google, December 2021

<https://www.google.com/intl/en/ipv6/statistics.html>

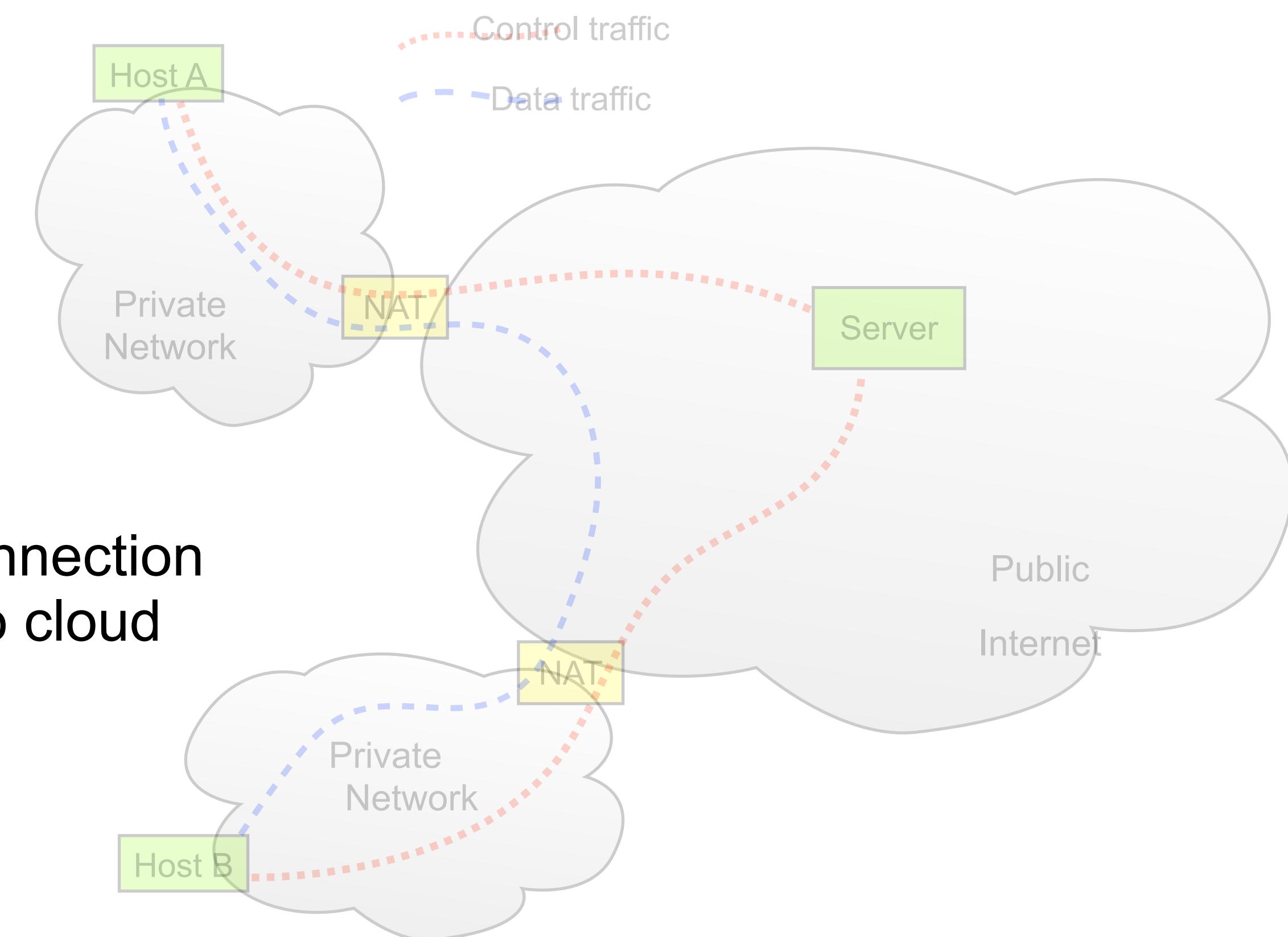
# Establishing Connectivity in the Modern Internet

- NAT traversal → binding discovery, exchange, probing
  - STUN, TURN, and the ICE algorithm
  - Requires a server with public IP address for binding discovery; centralised point of control
  - Complex, slow, unreliable, wastes power, generates unnecessary traffic



# Establishing Connectivity in the Modern Internet

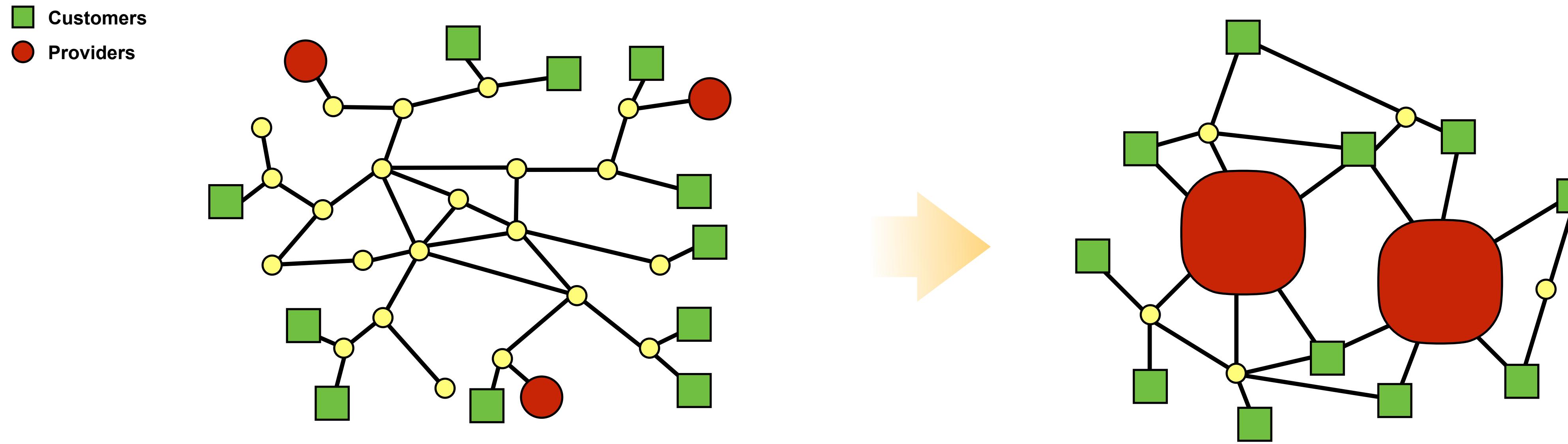
- Dual stack → connection racing
  - Some paths support IPv4, some IPv6; probing and connection timeouts slow for client-server applications
  - Race connections by probing in parallel: hard to implement, wastes resources
- NAT and dual stack operation complicates connection establishment; encourages centralisation onto cloud services



# Increasing Mobility

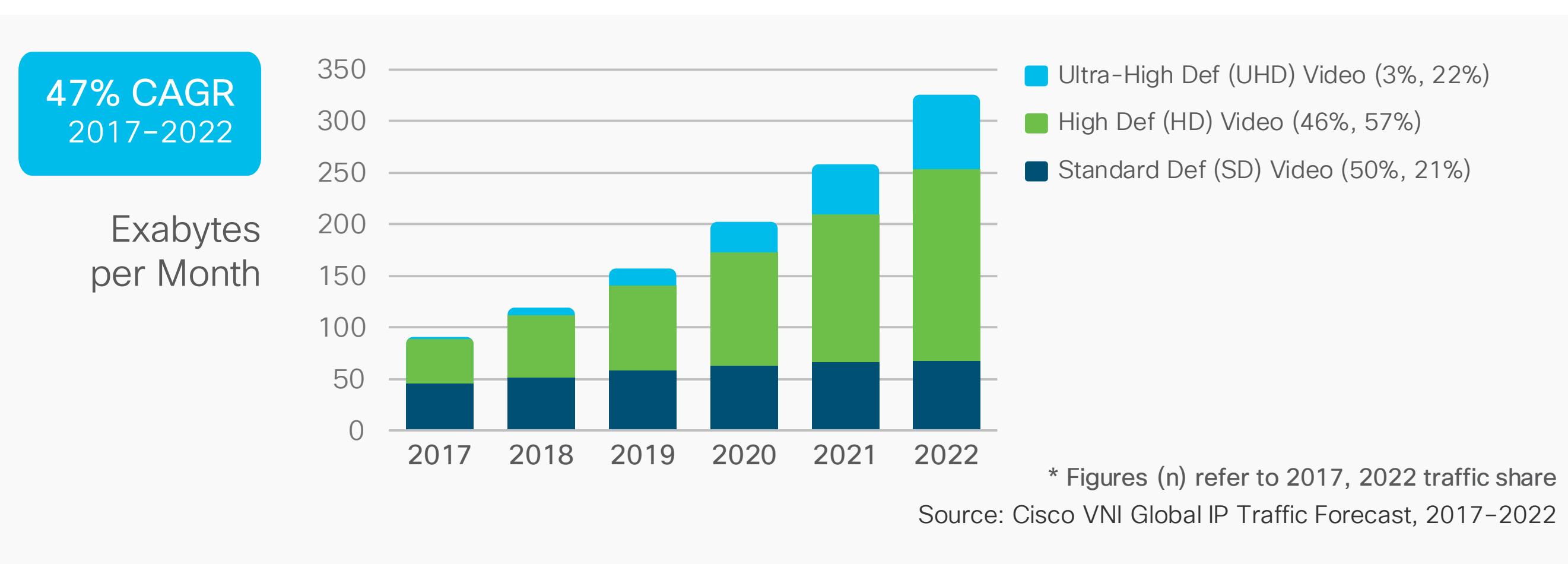
- IP addresses intended to encode location in the network
- Ubiquitous mobile devices – *smartphones* – complicate addressing and reachability since devices change their IP address each time they move:
  - TCP connections must be re-established after each move
  - UDP packets must be redirected after each move
  - Complex signalling and connection management if devices move frequently

# Hypergiants and Centralisation



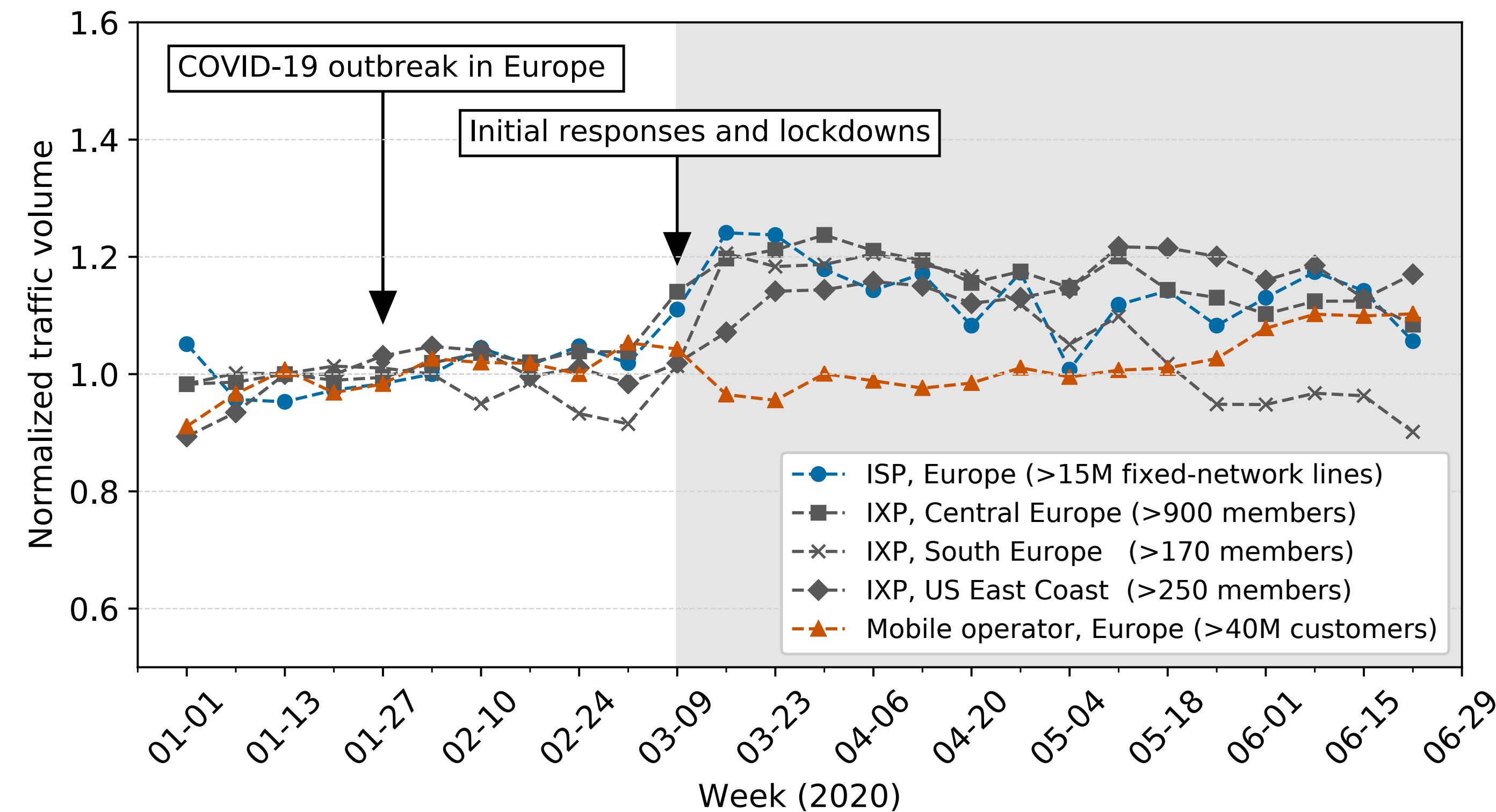
- Internet topology is flattening, becoming increasingly centralised
- Transit network ecosystem being replaced by direct connections from “eyeball” networks to content providers – Google, Facebook, Amazon, Akamai, ...
- Implications for network neutrality, competition, innovation

# Supporting Real-time Traffic



- Video streaming dominates Internet traffic, growing >40% per year
  - Packet loss → quality impairments or increased delay
  - Driving centralisation and direct CDN connections → easier to manage quality
  - Driving TCP congestion control and TCP replacements → BBR algorithm, QUIC; low-latency
- Interactive real-time media has stricter constraints
  - WebRTC, Zoom, VoIP, gaming, ... → non-TCP transport to lower latency

# Supporting Real-time Traffic: Impact of COVID-19 (1/2)

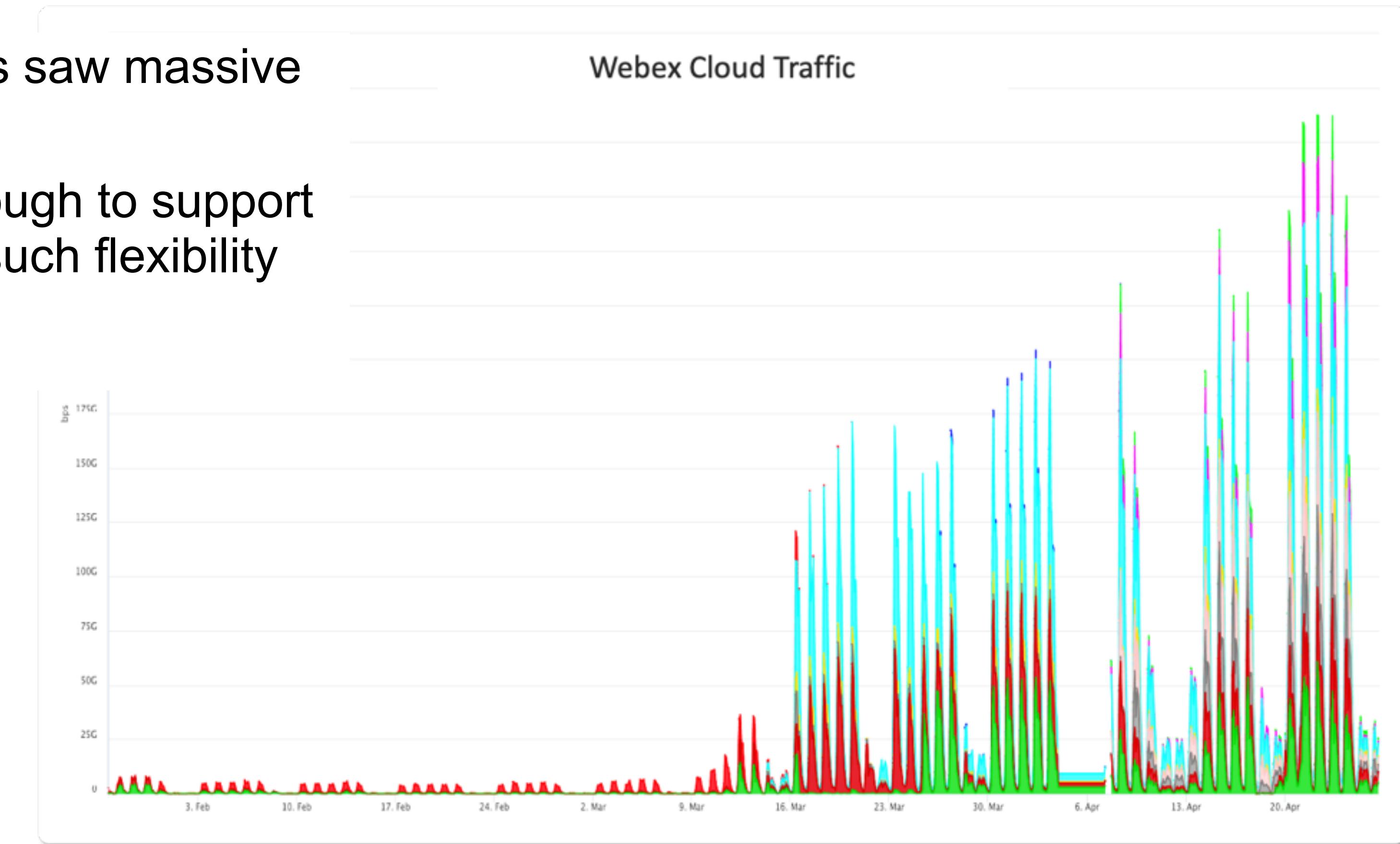


- Residential networks saw >20% increase in traffic overnight at the start of the COVID-19 lockdown
- Mobile networks saw ~10% drop in traffic

Anja Feldmann et al., "The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic", ACM Internet Measurement Conference, October 2020, <https://doi.org/10.1145/3419394.3423658>

# Supporting Real-time Traffic: Impact of COVID-19 (2/2)

- Video conferencing providers saw massive growth in traffic
- The Internet was flexible enough to support this shift – can we maintain such flexibility while also improving quality?

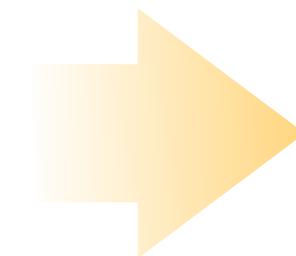


Source: Cullen Jennings and Panos Kozanian, "WebEx Scaling During Covid", Proceedings of the Internet Architecture Board COVID-19 Network Impacts Workshop, 2020, <https://www.iab.org/activities/workshops/covid-19-network-impacts-workshop-2020/>

# Pervasive Monitoring and Trust in the Network



Edward Snowden



Internet Engineering Task Force (IETF)  
Request for Comments: 7258  
BCP: 188  
Category: Best Current Practice  
ISSN: 2070-1721

S. Farrell  
Trinity College Dublin  
H. Tschofenig  
ARM Ltd.  
May 2014

Pervasive Monitoring Is an Attack

**Abstract**

Pervasive monitoring is a technical attack that should be mitigated in the design of IETF protocols, where possible.

**Status of This Memo**

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in [Section 2 of RFC 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7258>.

**Copyright Notice**

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Farrell & Tschofenig      Best Current Practice      [Page 1]

S. Farrell and H. Tschofenig, "Pervasive Monitoring is an Attack", Internet Engineering Task Force, RFC 7258, May 2014. <https://www.rfc-editor.org/rfc/rfc7258.html>

# Innovation in the Network

- Can we still change the network protocols?
  - NATs, firewalls, middleboxes → TCP is *hard* to change now
  - Transport encryption, tunnel over UDP → QUIC? Pervasive encryption to defeat pervasive monitoring and surveillance?
  - Protocol ossification is a real concern
  - Is the model smart edges and a dumb network appropriate?
- Can new startups compete with the hypergiants?
  - Several forces pushing towards centralisation – can we evolve the network to counter these?
  - Can we enable a decentralised network?

## Is it Still Possible to Extend TCP?

Michio Honda\*, Yoshifumi Nishida\*, Costin Raiciu†, Adam Greenhalgh‡,  
Mark Handley‡, Hideyuki Tokuda\*  
Keio University\*, Universitatea Politehnica Bucuresti\*, University College London‡  
{micchie,nishida}@sfc.wide.ad.jp, costin.raiciu@cs.pub.ro  
{a.greenhalgh,m.handley}@cs.ucl.ac.uk, hxt@ht.sfc.keio.ac.jp

### ABSTRACT

We've known for a while that the Internet has ossified as a result of the race to optimize existing applications or enhance security. NATs, performance-enhancing-proxies, firewalls and traffic normalizers are only a few of the middleboxes that are deployed in the network and look beyond the IP header to do their job. IP itself always that it was *stupid*; it did no task especially well, but it was extremely flexible and general, allowing a proliferation of protocols and applications that the original designers could never have foreseen.

In this paper we develop a measurement methodology for evaluating middlebox behavior relating to TCP extensions and present the results of measurements conducted from multiple vantage points. The short answer is that we can still extend TCP, but extensions' design is very constrained as it needs to take into account prevalent middlebox behaviors. For instance, absolute sequence numbers cannot be embedded in options, as middleboxes can rewrite ISN and preserve undefined options. Sequence numbering also must be consistent for a TCP connection, because many middleboxes only allow through contiguous flows.

We used these findings to analyze three proposed extensions to TCP. We find that MPTCP is likely to work correctly in the Internet or fallback to regular TCP. TepCrypt seems ready to be deployed, however it is fragile if resegmentation does happen—for instance with hardware offload. Finally, TCP extended options in its current form is not safe to deploy.

### Categories and Subject Descriptors

C.2.2 [Computer-communication Networks]: Network Protocols;  
C.2.6 [Computer-communication Networks]: Internetworking

### General Terms

Measurement, Design, Experimentation, Standardization

### Keywords

Middleboxes, Measurements, TCP, Protocol design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC'11, November 2–4, 2011, Berlin, Germany.  
Copyright 2011 ACM 978-1-4503-1013-0/11/11 ...\$10.00.

### 1. INTRODUCTION

The Internet was designed to be extensible; routers only care about IP headers, not what the packets contain, and protocols such as IP and TCP were designed with options fields that could be used to add additional functionality. The great virtue of the Internet was always that it was *stupid*; it did no task especially well, but it was extremely flexible and general, allowing a proliferation of protocols and applications that the original designers could never have foreseen.

Unfortunately the Internet, as it is deployed, is no longer the Internet as it was designed. IP options have been unusable for twenty years[10] as they cause routers to process packets on their slow path. Above IP, the Internet has benefited (or suffered, depending on your viewpoint) from decades of optimizations and security enhancements. To improve performance [2, 7, 18, 3], reduce security exposure [15, 29], enhance control, and work around address space shortages [22], the Internet has experienced an invasion of middleboxes that *do care* about what the packets contain, and perform processing at layer 4 or higher *within* the network.

The problem now faced by designers of new protocols is that there is no longer a well defined or understood way to extend network functionality, short of implementing everything over HTTP[25]. Recently we have been working on adding both multipath support[11] and native encryption[5] to TCP. The obvious way to do this, in both cases, is to use TCP options. In the case of multipath, we would also like to stripe data across more than one path. At the end systems, the protocol design issues were mostly conventional. However, it became increasingly clear that no one, not the IETF, not the network operators, and not the OS vendors, knew what will and what will not pass through all the middleboxes as they are currently deployed and configured. Will TCP options pass unchanged? If the sequence space has holes, what happens? If a retransmission has different data than the original, which arrives? Are TCP segments coalesced or split? These and many more questions are crucial to answer if protocol designers are to extend TCP in a deployable way. Or have we already lost the ability to extend TCP, just like we did two decades ago for IP?

In this paper we present the results from a measurement study conducted from 142 networks in 24 countries, including cellular, WiFi and wired networks, public and private networks, residential, commercial and academic networks. We actively probe the network to elicit middlebox responses that violate the end-to-end transparency of the original Internet architecture. We focus on TCP, not only because it is by far the most widely used transport protocol, but also because while it is known that many middleboxes modify TCP behavior [6], it is not known how prevalent such middleboxes are, nor precisely what the *emergent behavior* is with TCP extensions that were unforeseen by the middlebox designers.

181

M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda,  
“Is it still possible to extend TCP?” Proc. ACM Internet Measurement Conference, Berlin, Nov. 2011. <https://dx.doi.org/10.1145/2068816.2068834>



# Can the network evolve to address these challenges?

- How to establish connections in a fragmented network?
- How can encryption protect against pervasive monitoring and prevent transport ossification?
- How can we reduce latency and support real-time and interactive content?
- How can we adapt to the vagaries of wireless networks?
- How can we identify and distribute content? How can we manage the tussle for control of the DNS and naming?
- How can we manage interdomain routing to efficiently delivery content?
- How can we re-decentralise the network?

# Connection Establishment in a Fragmented Network

Networked Systems (H)  
Lecture 2

# Lecture Outline

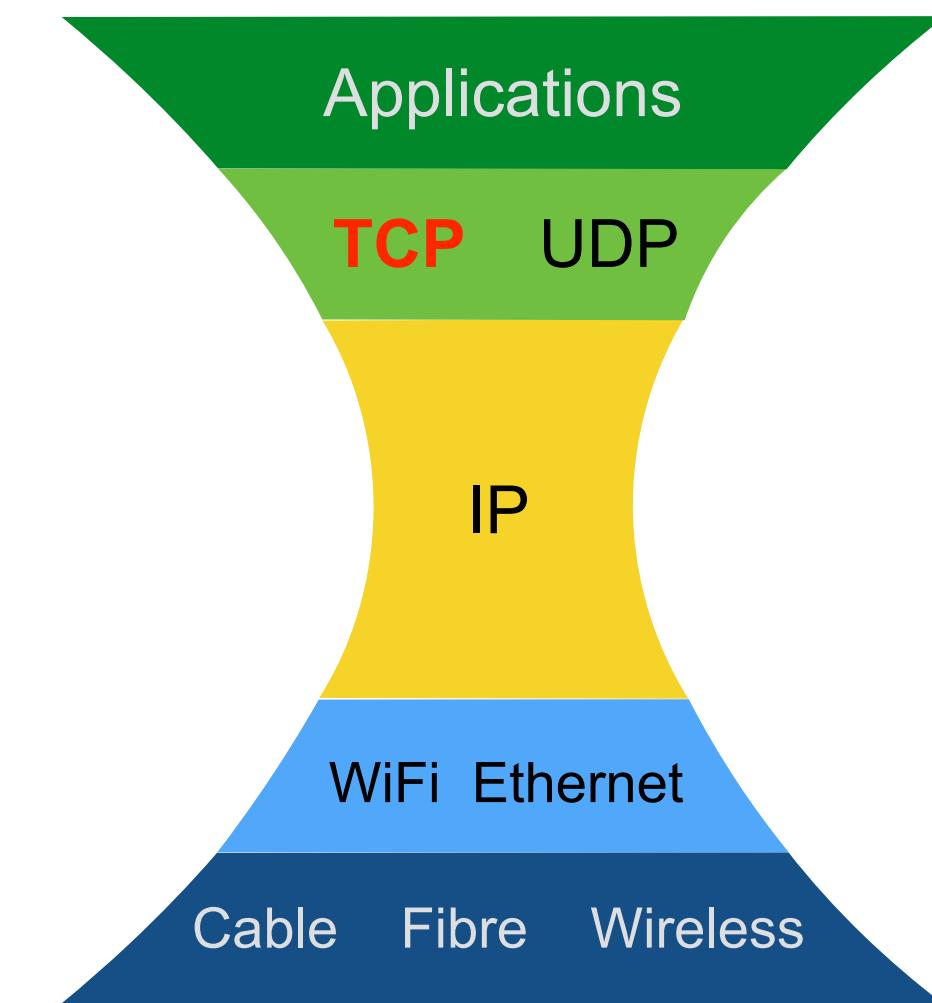
- Review of TCP and client-server connection establishment
- Impact of TLS and IPv6 on connection establishment
- Peer-to-peer connections and Network Address Translation
- Problems caused by NAT
- NAT traversal and peer-to-peer connection establishment

# Review of TCP

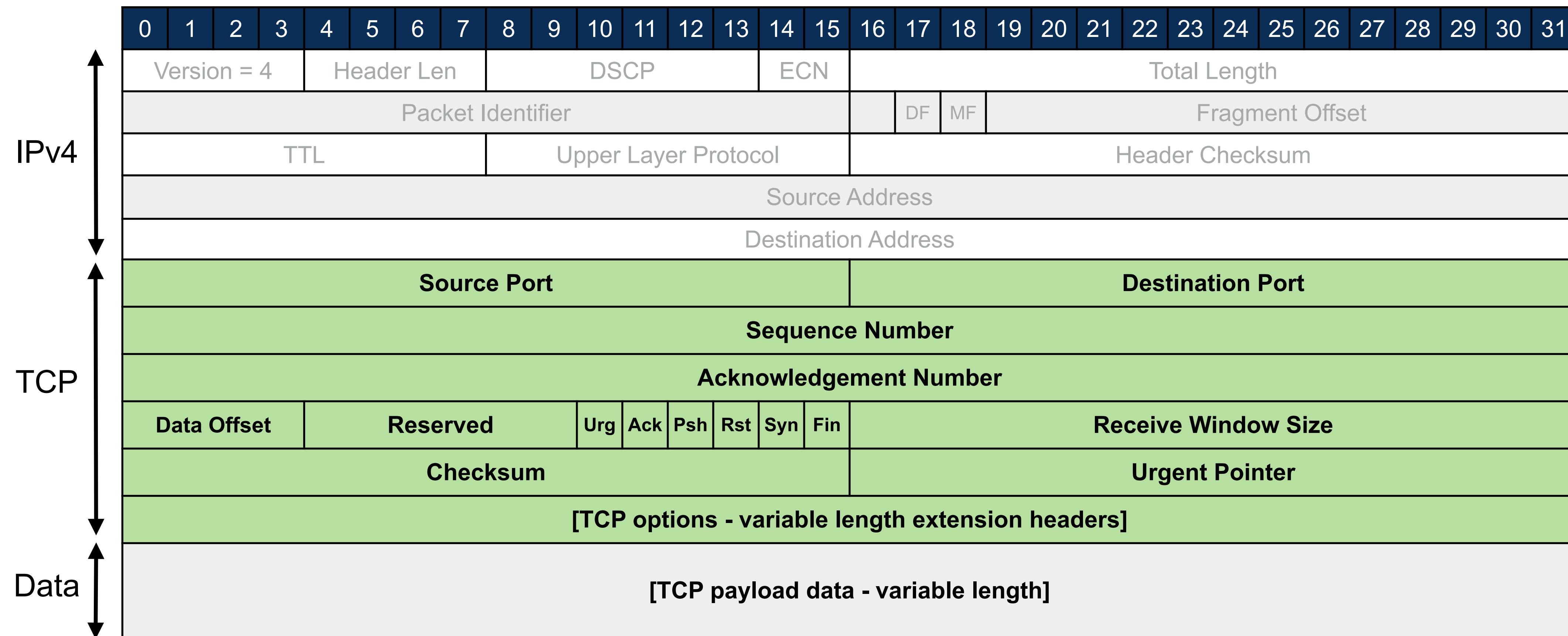
- The purpose of TCP
- Segment format
- Service model
- TCP programming with BSD Sockets

# TCP: Reliable Data Transport for the Internet

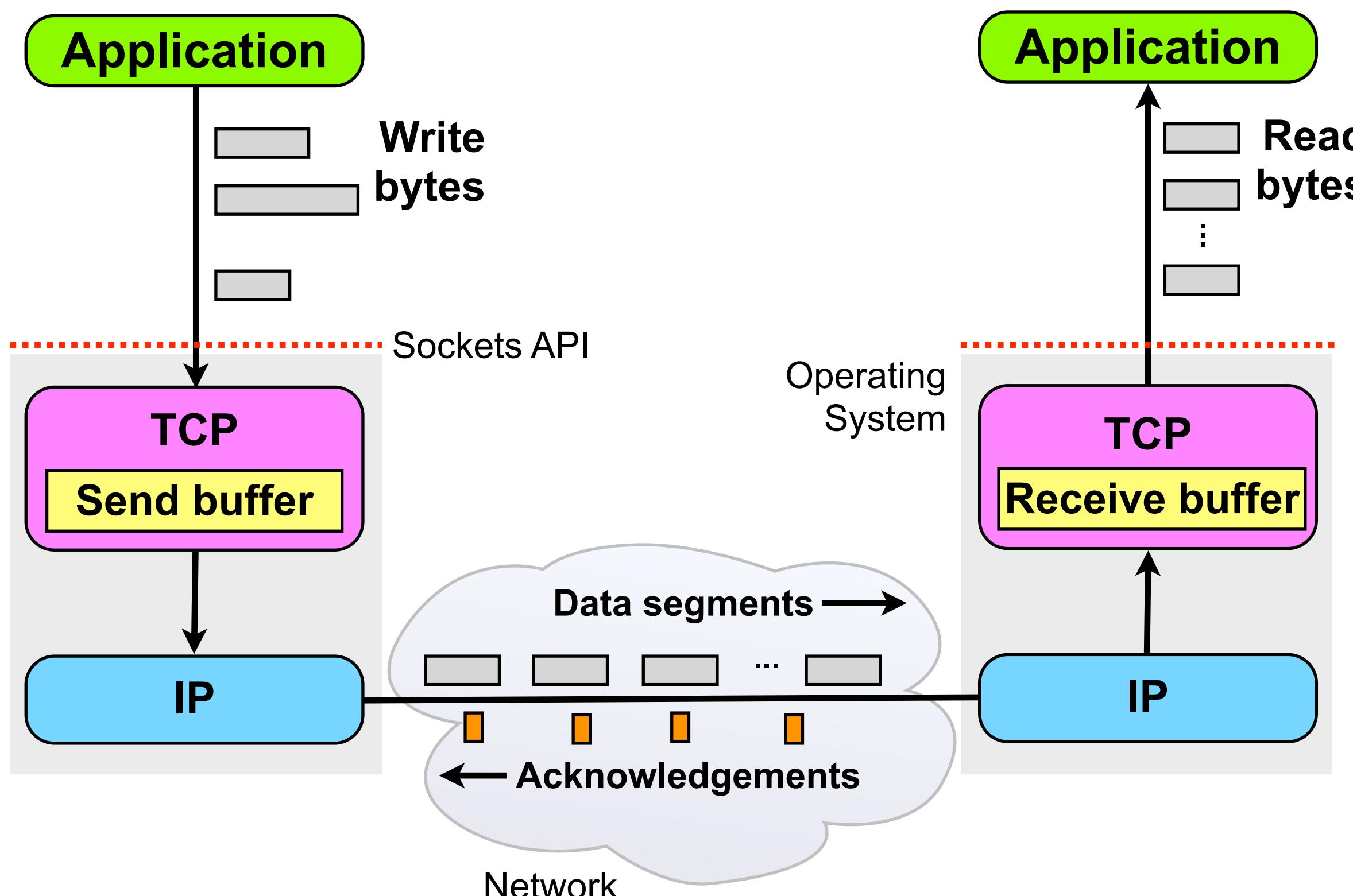
- TCP transport layer
  - Provides a reliable, ordered, byte stream service over the best-effort IP network
  - Provides congestion control, to adapt to network capacity
  - Most widely used Internet transport protocol
- Delivered within IP
  - TCP **segments** carried as data in IP packets
  - IP **packets**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



# TCP Segment Format

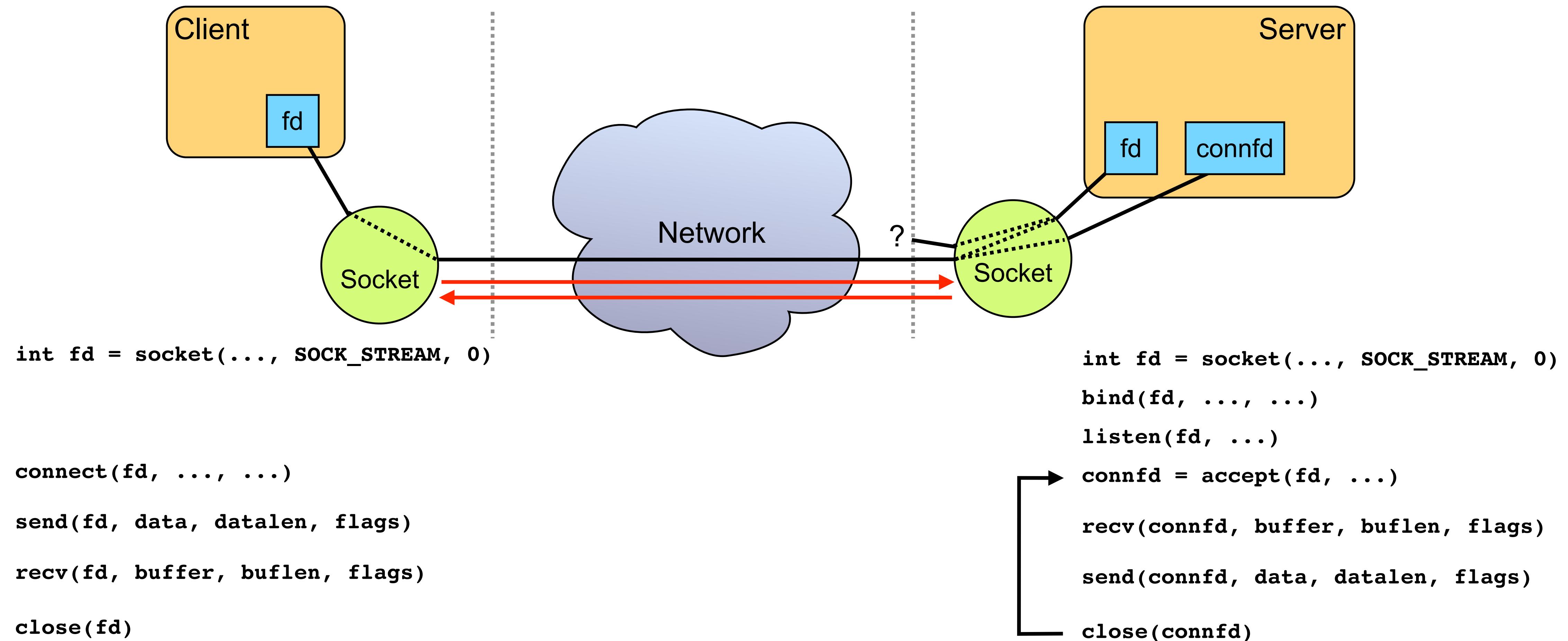


# TCP



- Reliable, ordered, byte stream delivery service running over IP
  - Lost packets are retransmitted; ordering is preserved; message boundaries **are not** preserved
  - Adapts sending rate to match network capacity → congestion control
  - Adds port number to identify services
- Used by applications needing reliability → default choice for most applications

# TCP Programming Model

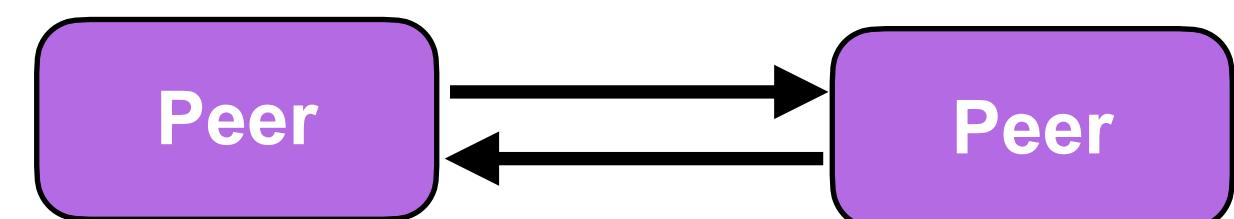
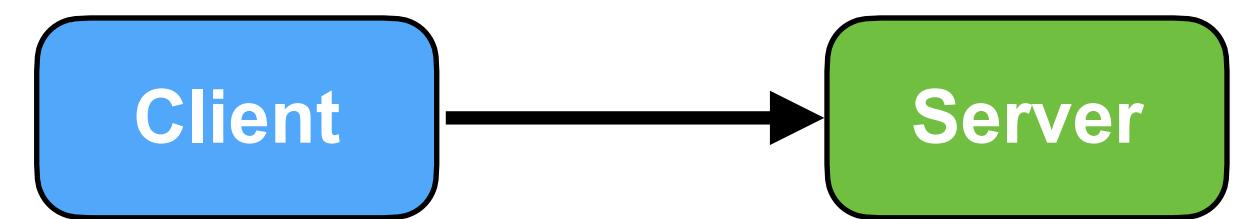


# Client-server Connections

- Connection Establishment and the Impact of the RTT on performance

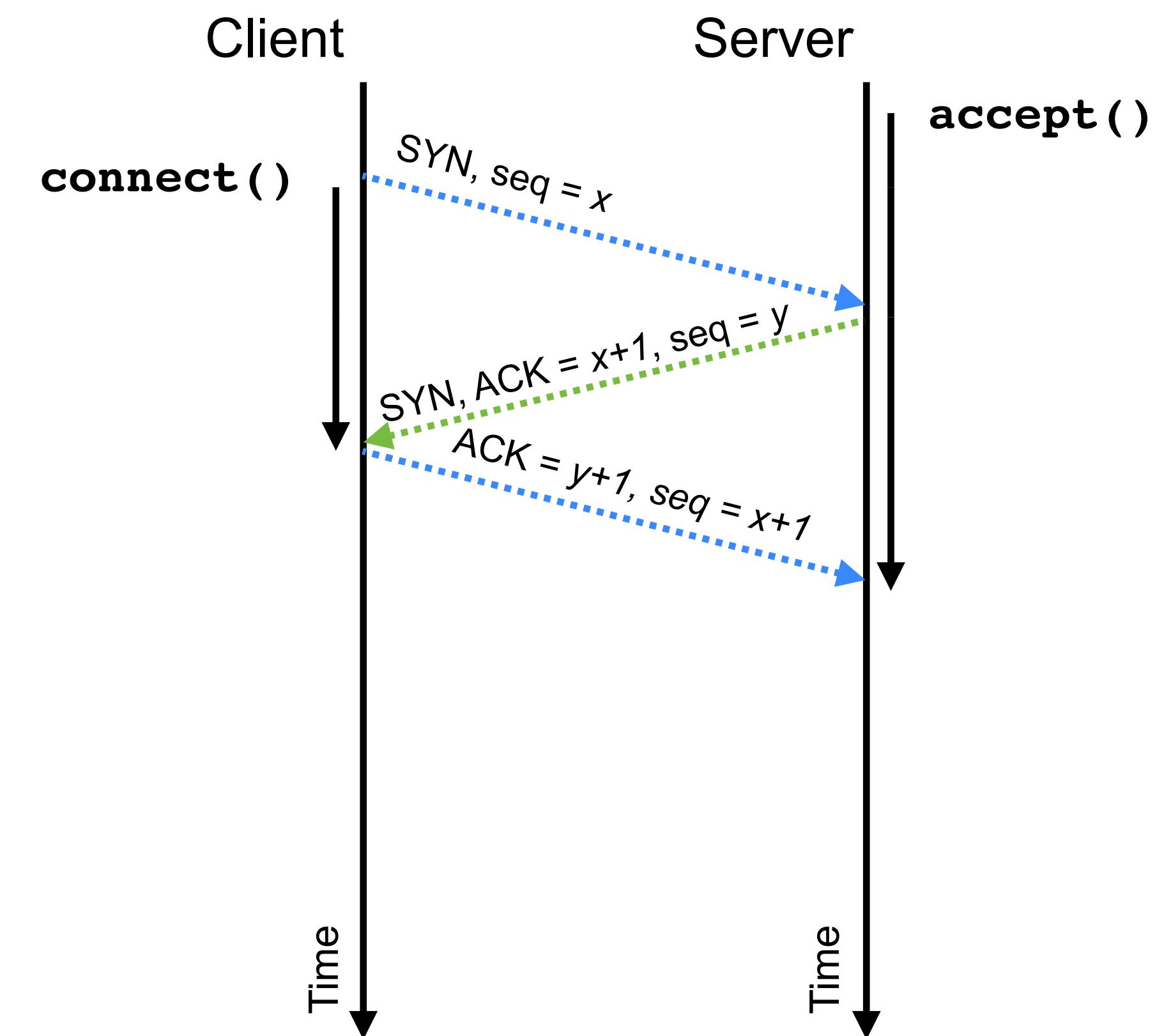
# TCP Connections

- Typical TCP-based applications are client-server
  - Server listens for connections on a well-known 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



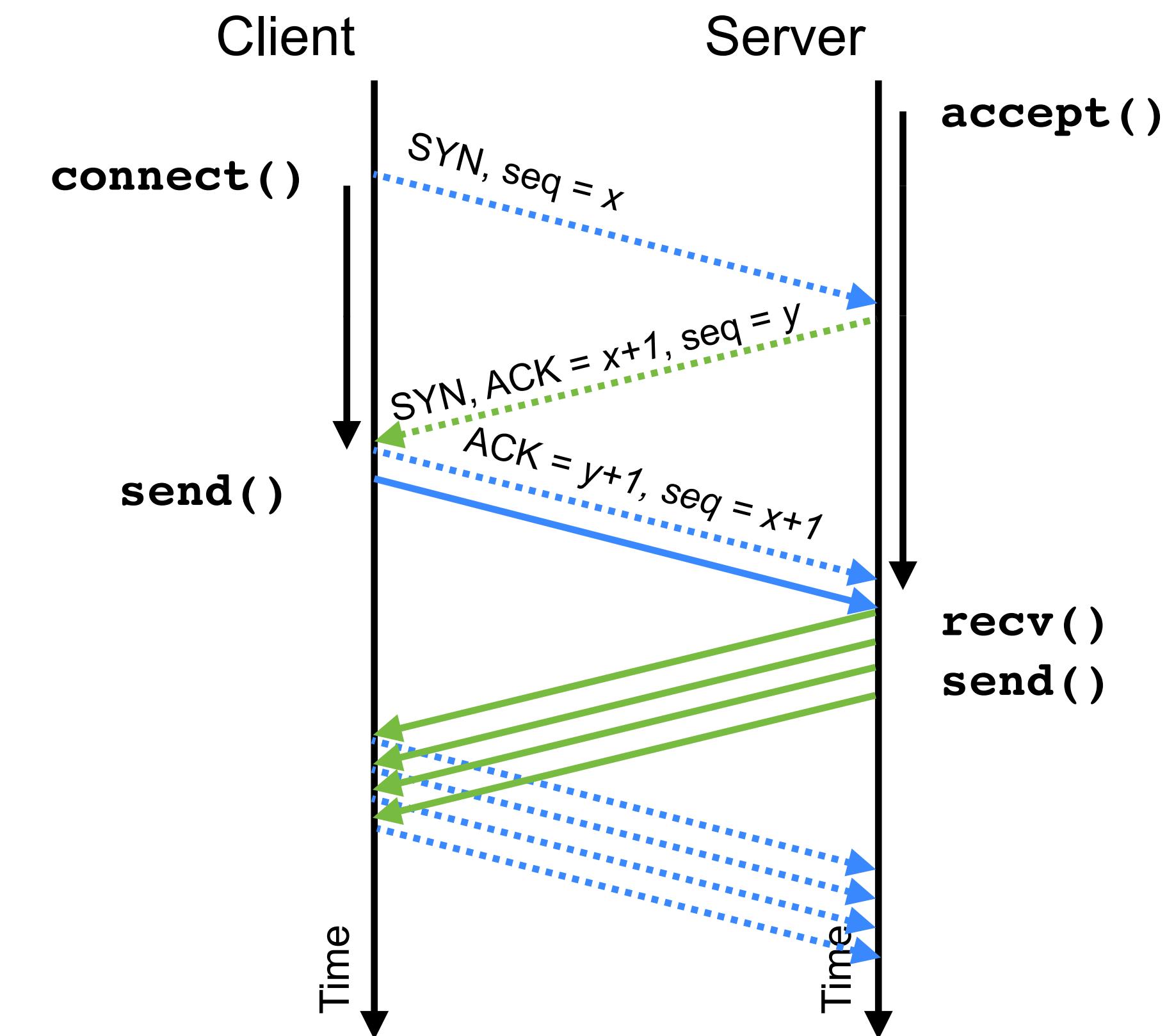
# TCP Client-Server Connection Establishment

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



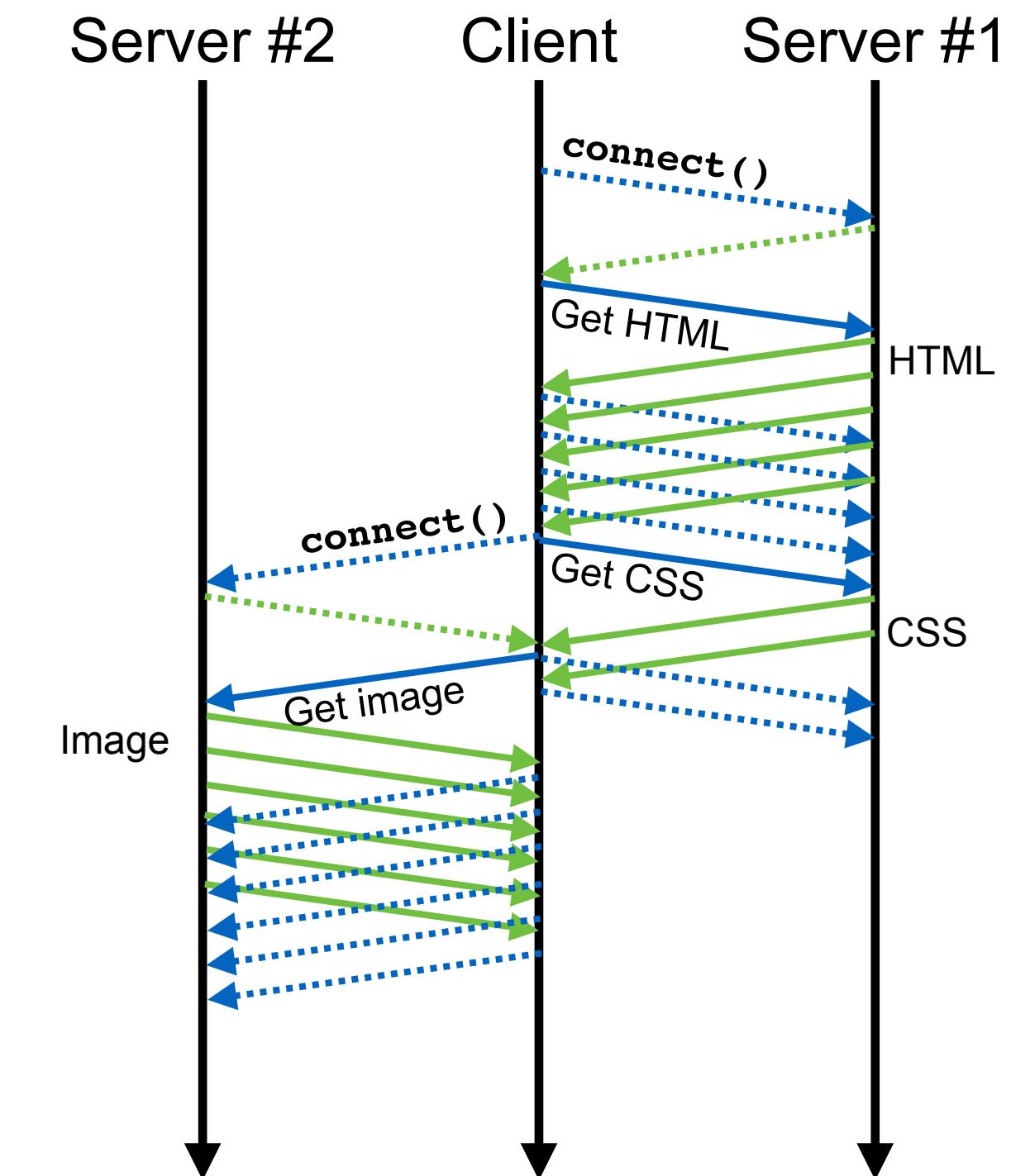
# TCP Client-Server Connection Establishment

- Calls **send()**/**recv()** transmit data
- For short flows, initial three-way handshake can take a significant fraction of connection lifetime



# 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. Example:

- 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

# 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
- (all assuming otherwise idle network)

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

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

# 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 → -90%
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 → -45%
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 → -22%

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?

# 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:
```

```
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...
```

- Example: fetch web page using HTTP/1.1
  - Once a connection is open, each object requires only a single request-response exchange
  - Connections are reused for different objects, when possible

# 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
```

- Example: sending email using SMTP
  - Line-by-line request and response:
  - Many unnecessary round trips to server
  - Compare to HTTP/1.1, that bundles all parameters into a single request
  - Poor performance

# Client-server connections using TCP

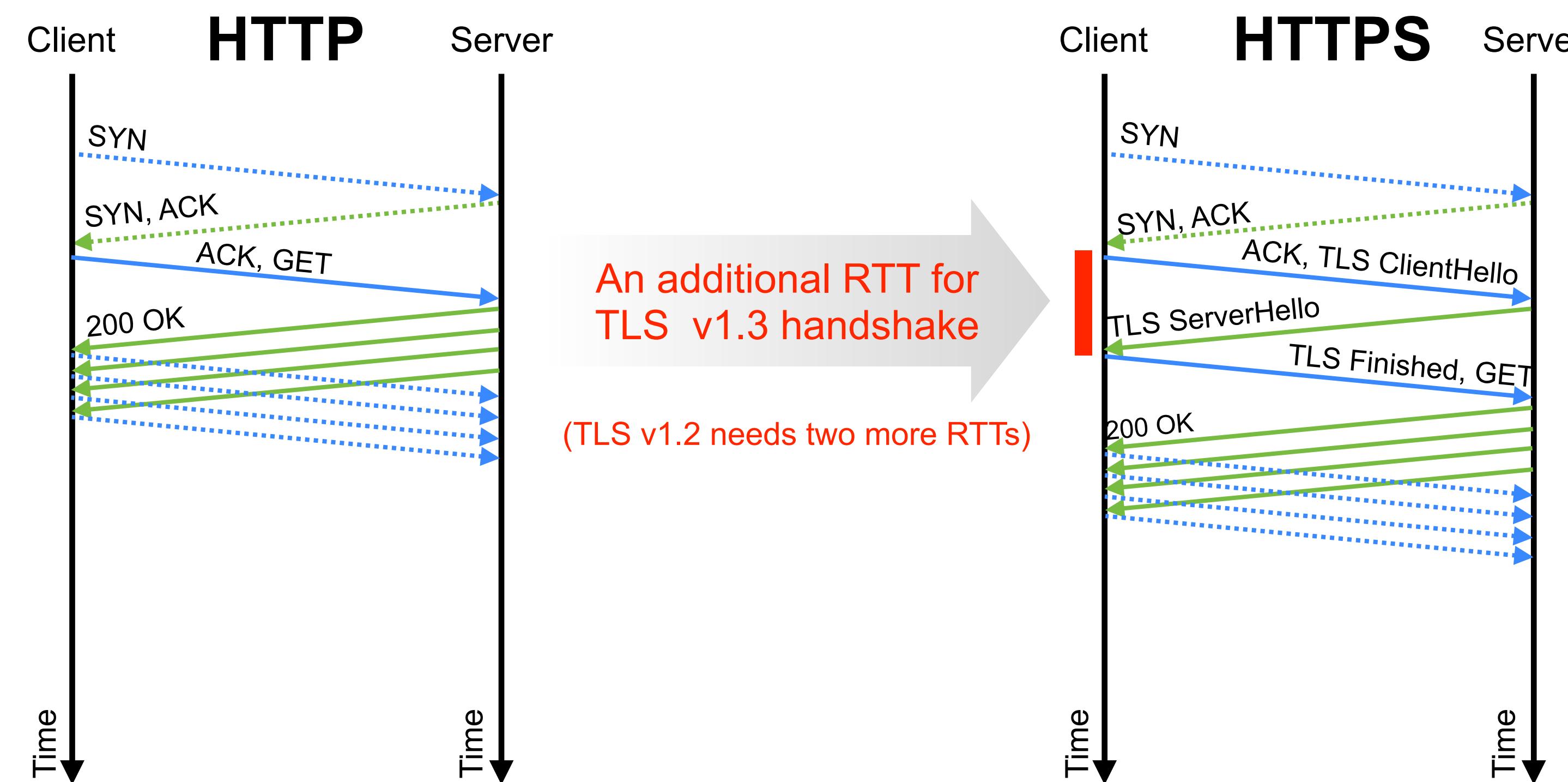
- TCP establishes connections using a three-way handshake
- Performance of a connection depends on latency and bandwidth
- Latency is often dominant factor

# Impact of TLS and IPv6 on Connection Establishment

- Transport Layer Security (TLS) needs extra RTTs to negotiate security
- Dual-stack IPv4 and IPv6 hosts must race connections for good performance

# 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?



# Impact of Transport Layer Security (2/2)

- Revisit simple web page download example:
  - 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 **HTTPS** over TCP/IP
  - TLS v1.3 introduces two RTTs extra latency
    - One extra RTT per connection to negotiate security parameters
    - Further causes impact of RTT and connection establishment to dominate performance

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 → -90%
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 → -36%
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 → -16%

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

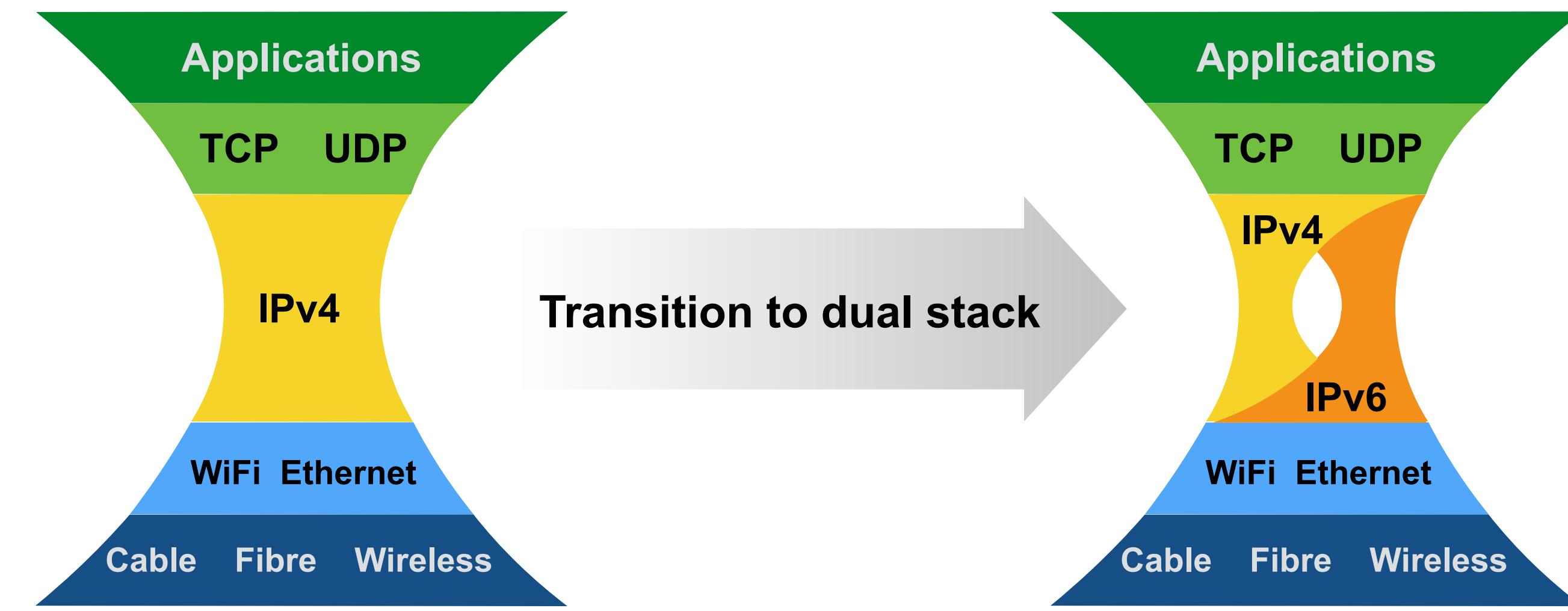
Assuming no impact due to congestion control

With additional RTT to model TLS 1.3 handshake

# 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
    - 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

# 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 block IPv4, some block IPv6, some block both types of traffic
- IPv6 network is **not** a subset of the IPv4 network; it's separate, but overlaps in places

# Happy Eyeballs

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

```
[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] >
```

# Happy Eyeballs

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

Internet Engineering Task Force (IETF)  
Request for Comments: 8305  
Obsoletes: 6555  
Category: Standards Track  
ISSN: 2070-1721

D. Schinazi  
T. Pauly  
Apple Inc.  
December 2017

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/>



# Summary: Connection Latency

- Two factors affect performance:
  - **Bandwidth** and **RTT**
  - In many cases, RTT dominates
- How to improve performance?
  - Overlap `connect()` calls for hosts with multiple addresses
  - Reduce number of TCP connections made
  - Reduce number of requests per connection
  - Overlap TCP and TLS handshakes
    - QUIC transport protocol → lecture 4
  - Reduce RTT
    - Increase speed of light or reduce queuing delay → lectures 5, 6

# Impact of TLS and IPv6 on Connection Establishment

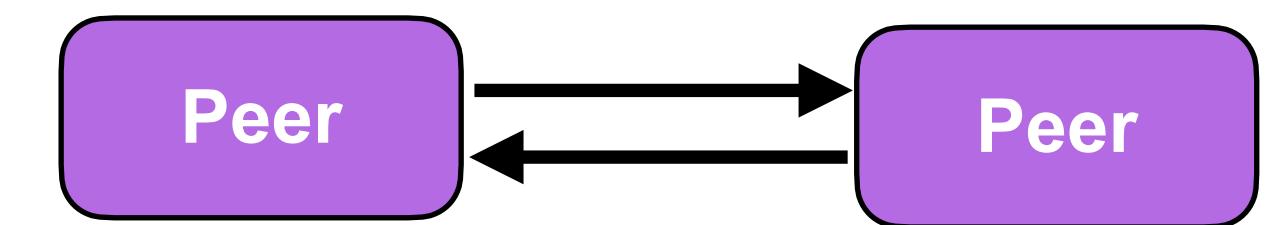
- Transport Layer Security (TLS) needs extra RTTs to negotiate security
- Dual-stack IPv4 and IPv6 hosts must race connections for good performance

# Peer-to-peer Connections

- Peer-to-peer connections and Network Address Translation

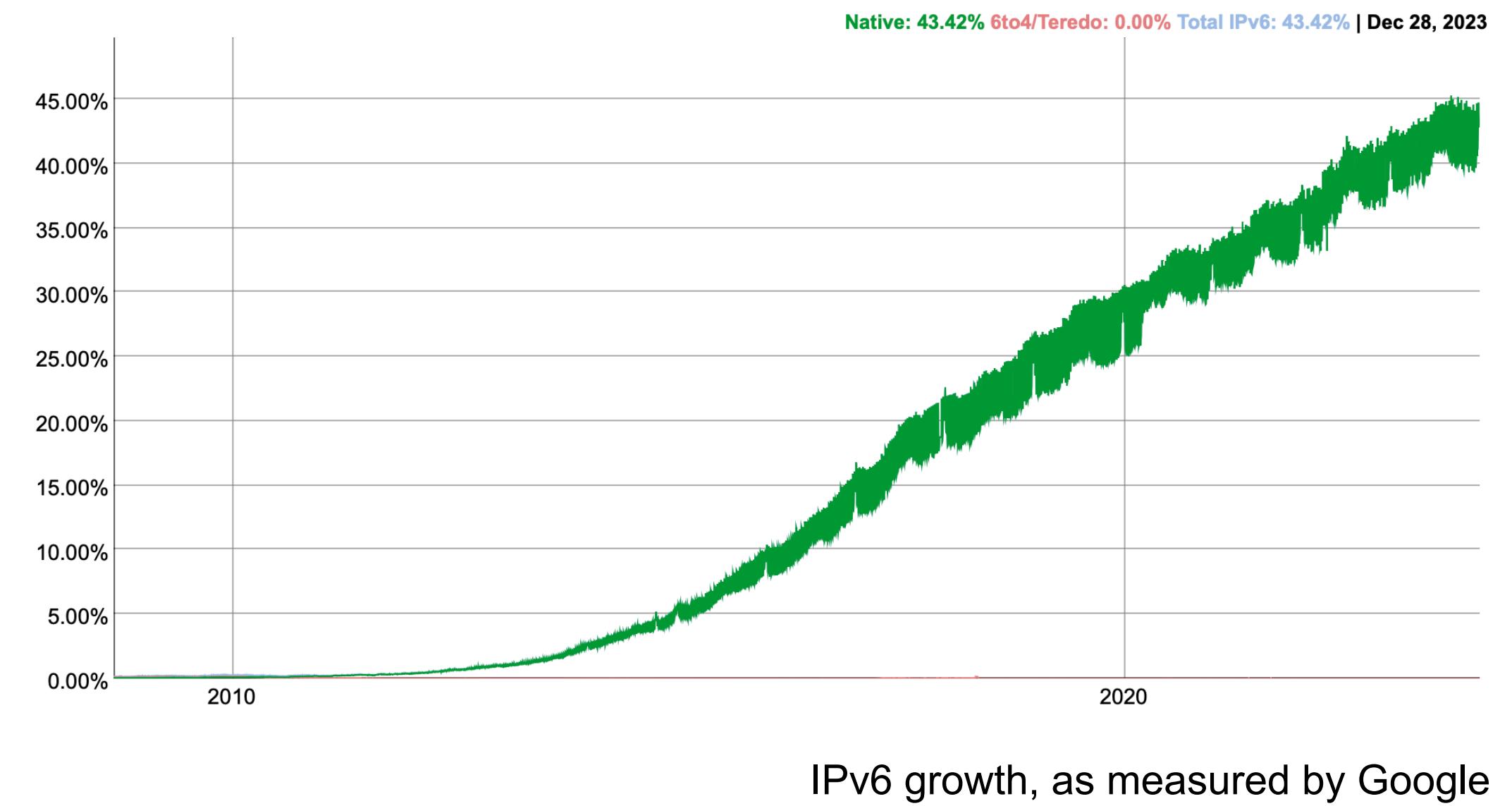
# Peer-to-peer Connection Establishment

- The Internet is conceptually a peer-to-peer network – any device can, in principle, talk to any other
- You should be able to run a TCP server on any device
- You should be able to run a TCP- or UDP-based peer-to-peer application
- In practise peer-to-peer connection establishment is difficult, due to **network address translation (NAT)**

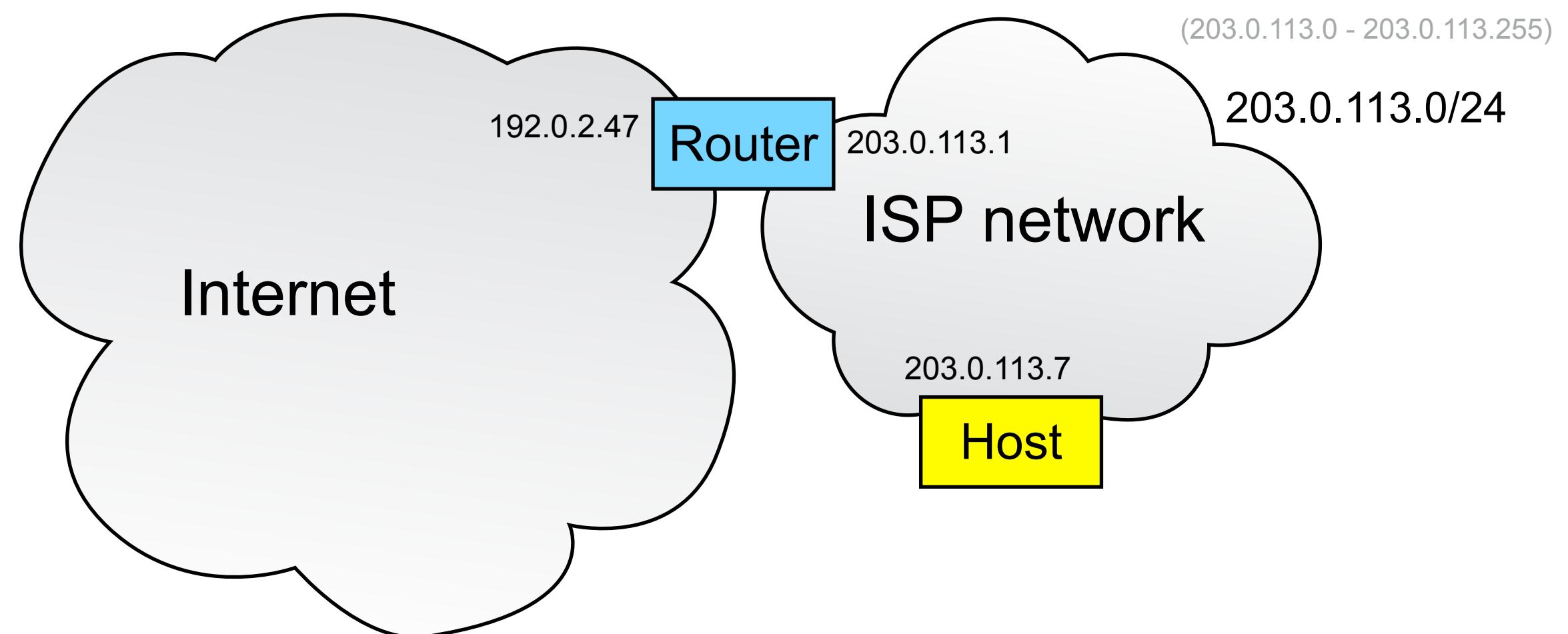


# Network Address Translation (NAT)

- IPv4 address space is exhausted
- IPv6 is the long-term solution, but the transition is taking many years
- Network address translation (NAT) is a work-around for the shortage of IPv4 addresses; it allowing several devices to share a single IP address

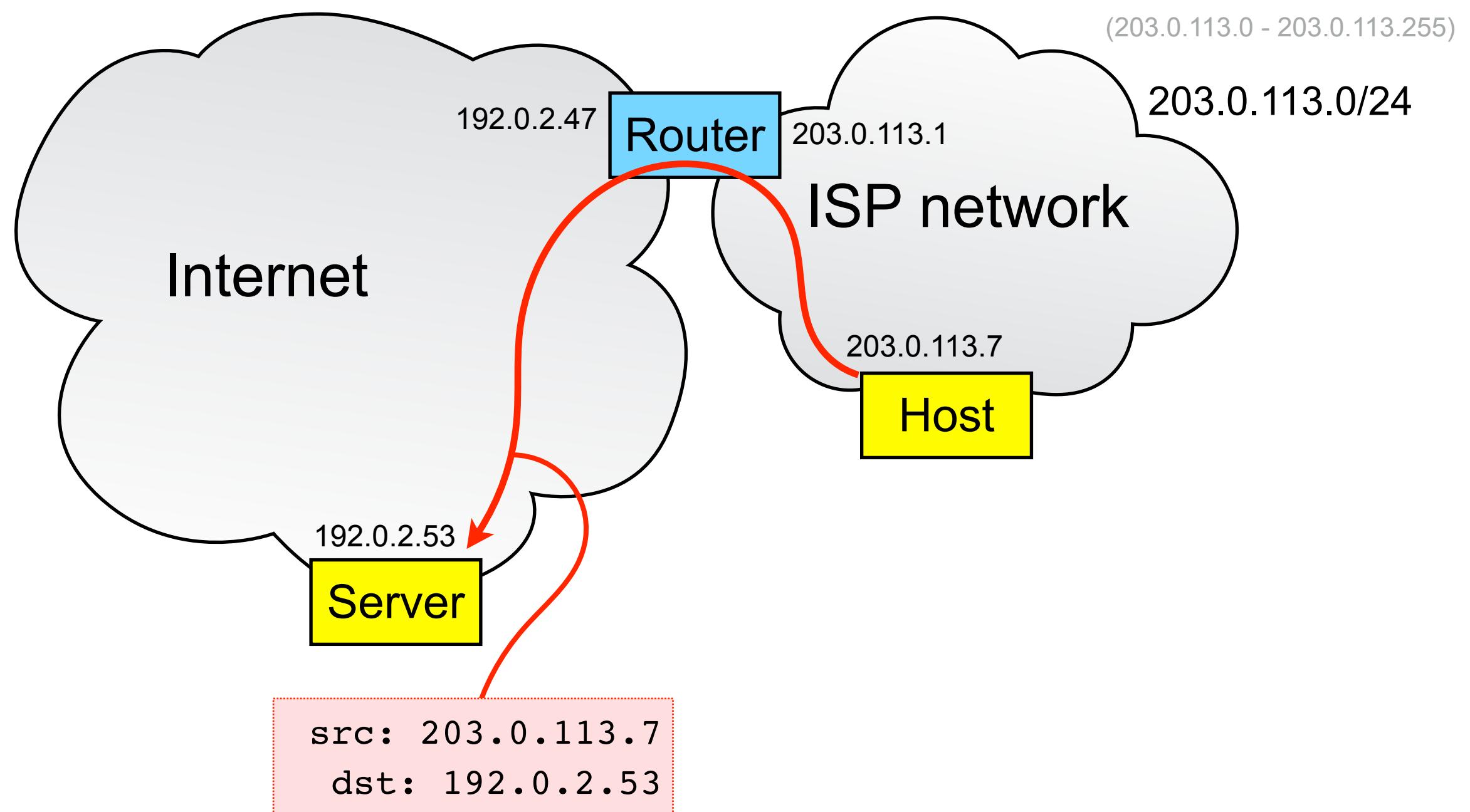


# Connecting a Single Host



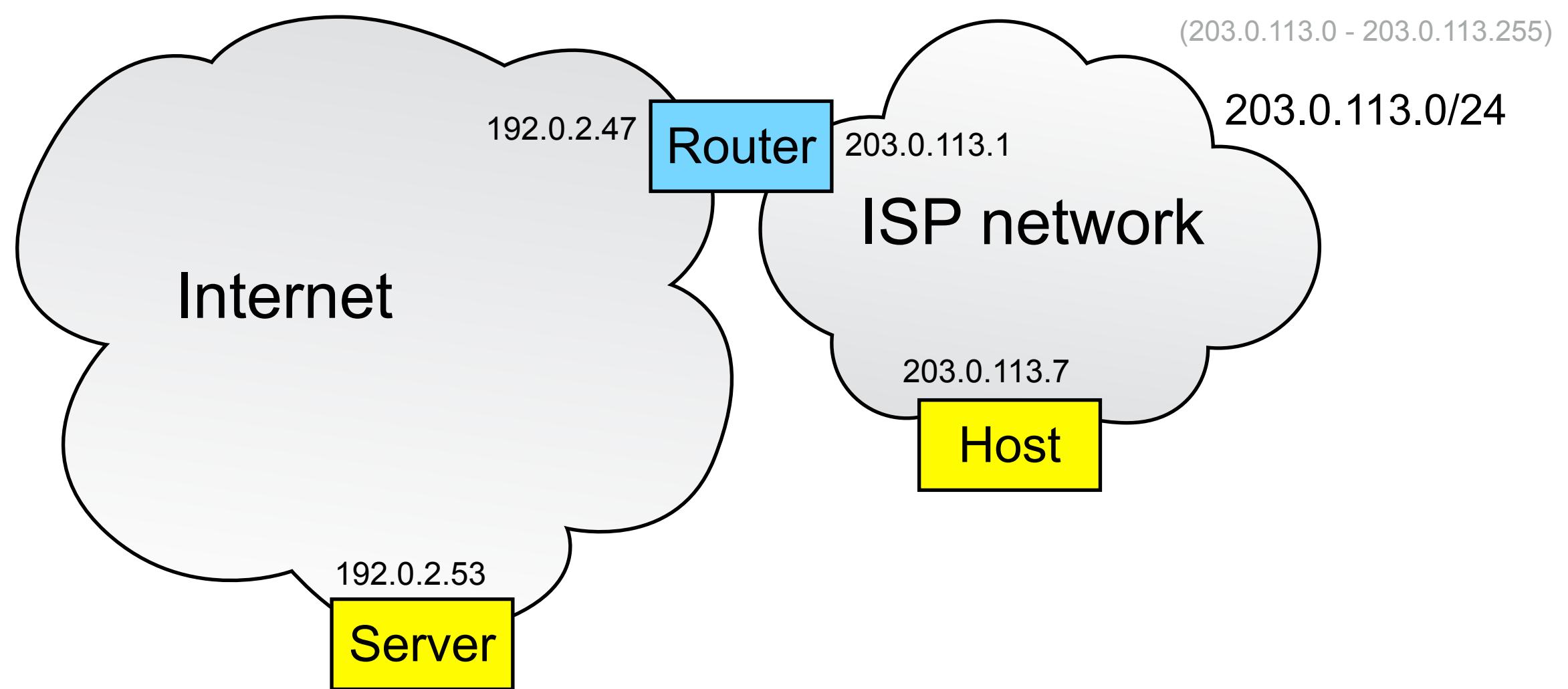
- An Internet service provider (ISP) owns an IP address prefix
- They assign a customer a single address for a single host

# Connecting a Single Host



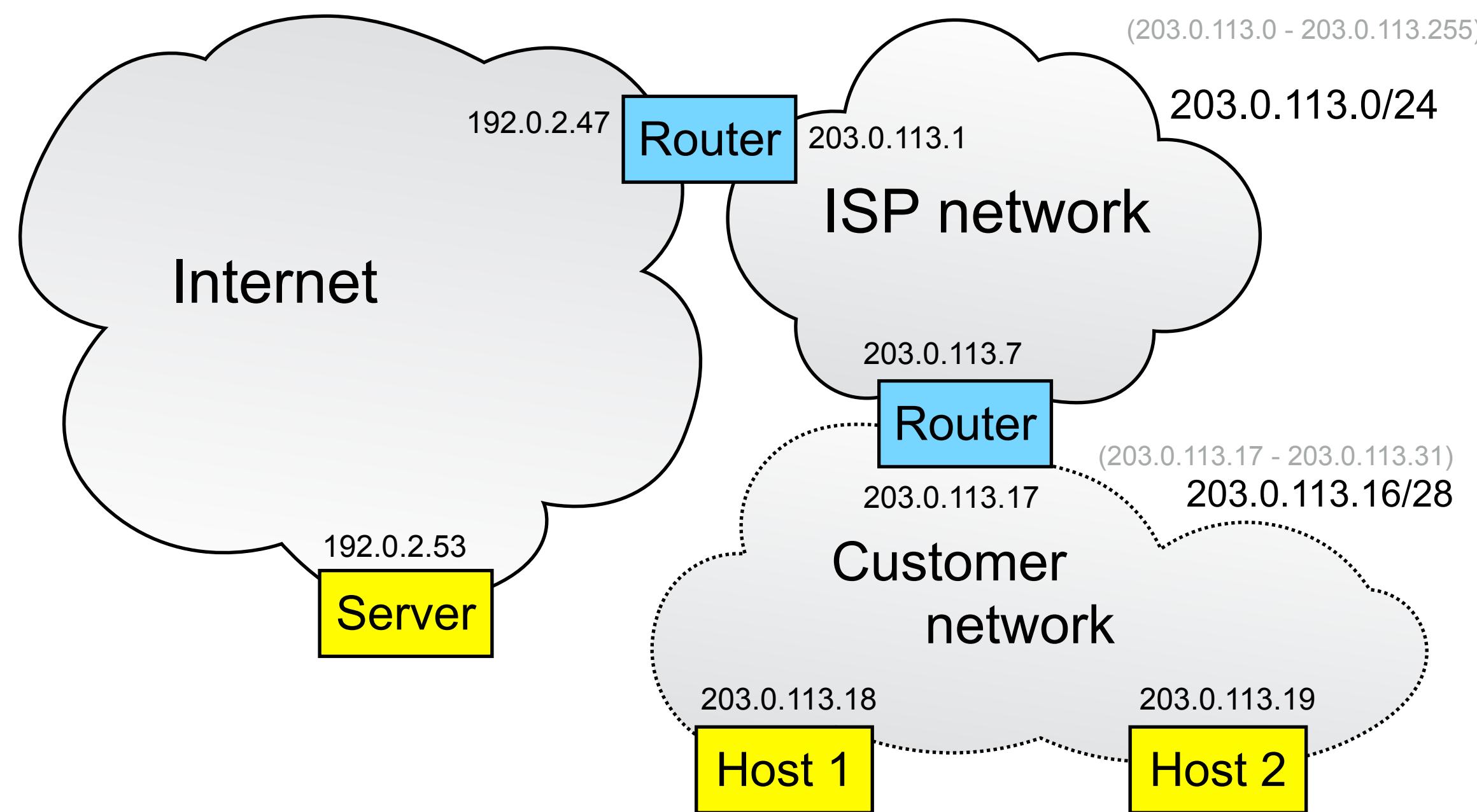
- An Internet service provider owns an IP address prefix
- They assign a customer a single address for a single host
- No address translation

# Connecting Multiple Hosts



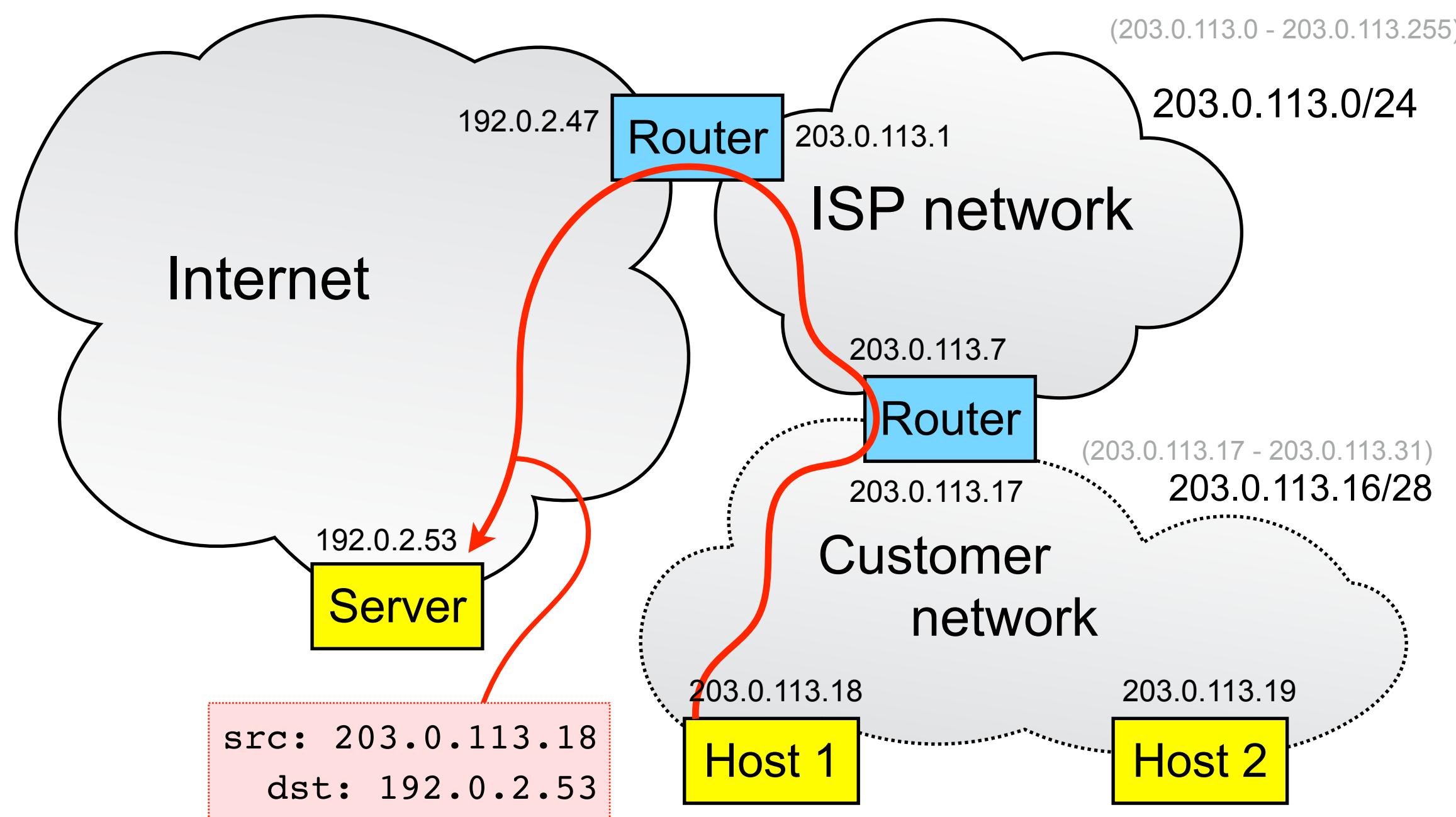
- The customer buys another host
- How does it connect?

# Connecting Multiple Hosts



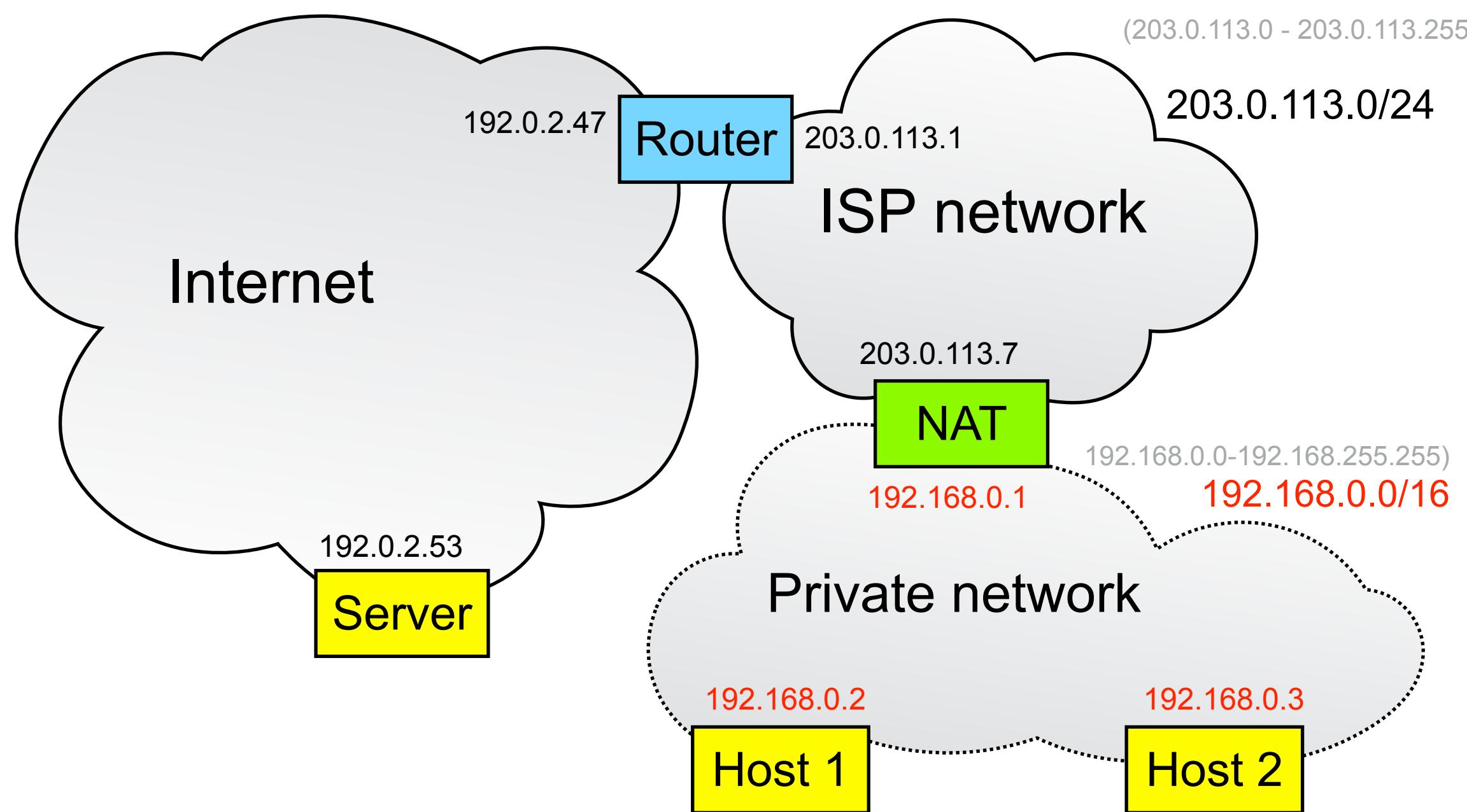
- The customer buys another host
- How does it connect?
- What's supposed to occur:
  - Customer acquires a router, which gets the customer's previous IP address
  - ISP assigns new range of IP addresses to customer (from the ISP's prefix)
  - Customer gives each host an address from that new range

# Connecting Multiple Hosts



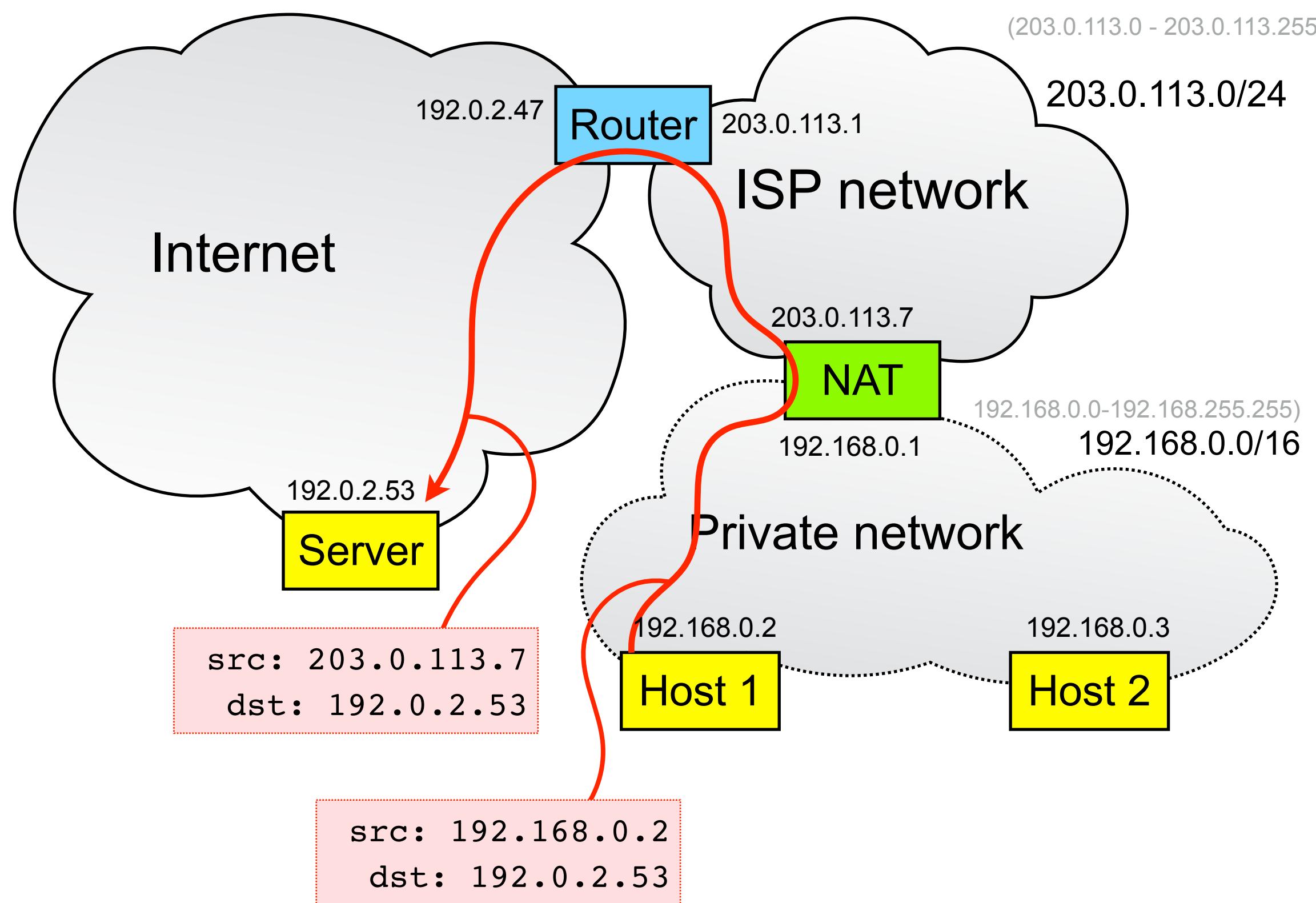
- The customer buys another host
- How does it connect?
- What's supposed to occur:
  - Customer acquires a router, which gets the customer's previous IP address
  - ISP assigns new range of IP addresses to customer's network (from the ISP's prefix)
  - Customer gives each host an address from that new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

# Network Address Translation



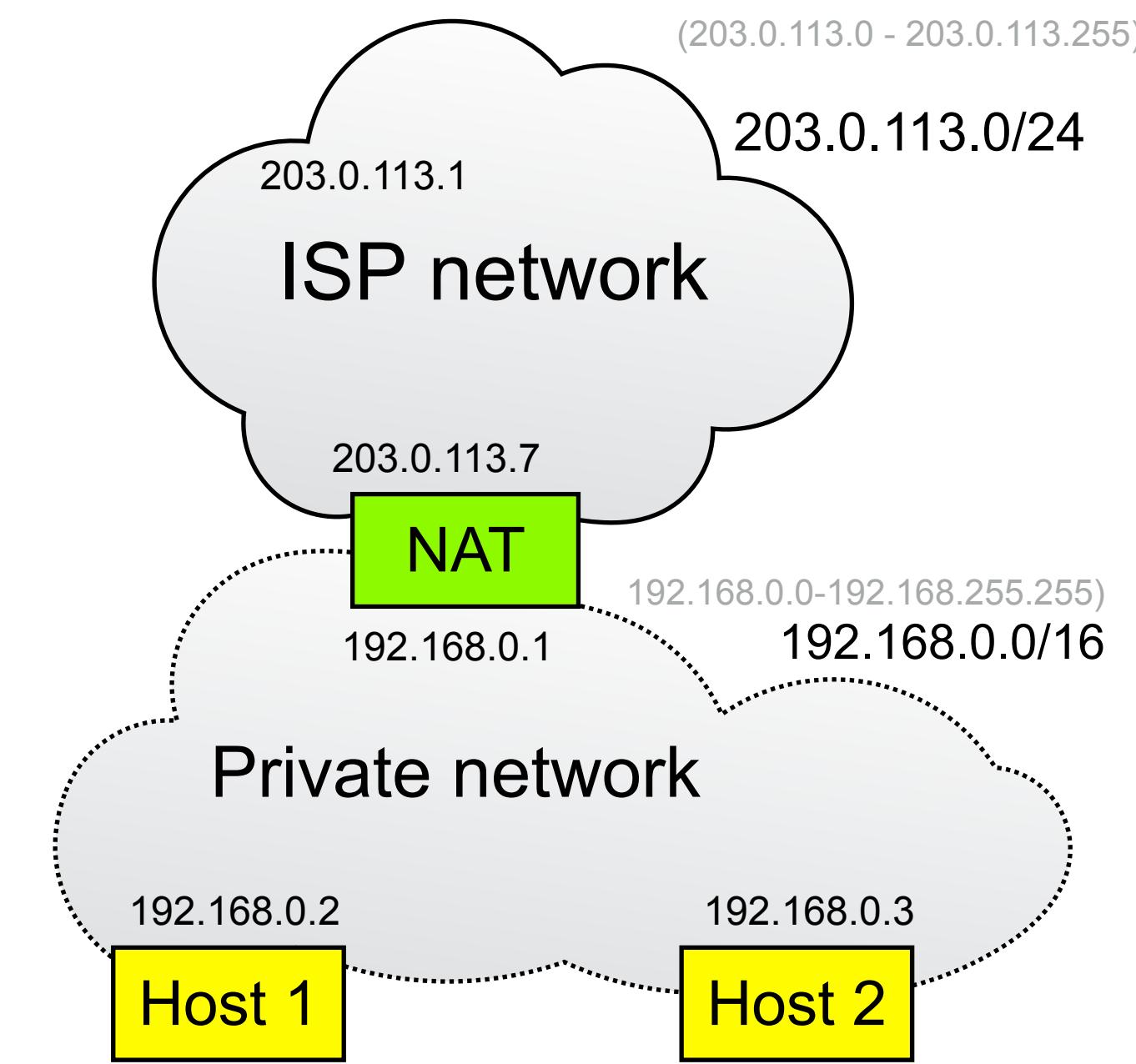
- The customer buys another host
- How does it connect?
- What actually happens:
  - Customer acquires a NAT router, which gets the customer's previous IP address
  - Customer gives each host on their network a private address

# 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

# NAT and Private Address Ranges



- The NAT hides a private network behind a single public IP address
  - The private IP network address ranges are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16
- Gives the illusion of more address space, by reusing IP addresses in different parts of the network
  - e.g., most home networks use 192.168.0.0/16

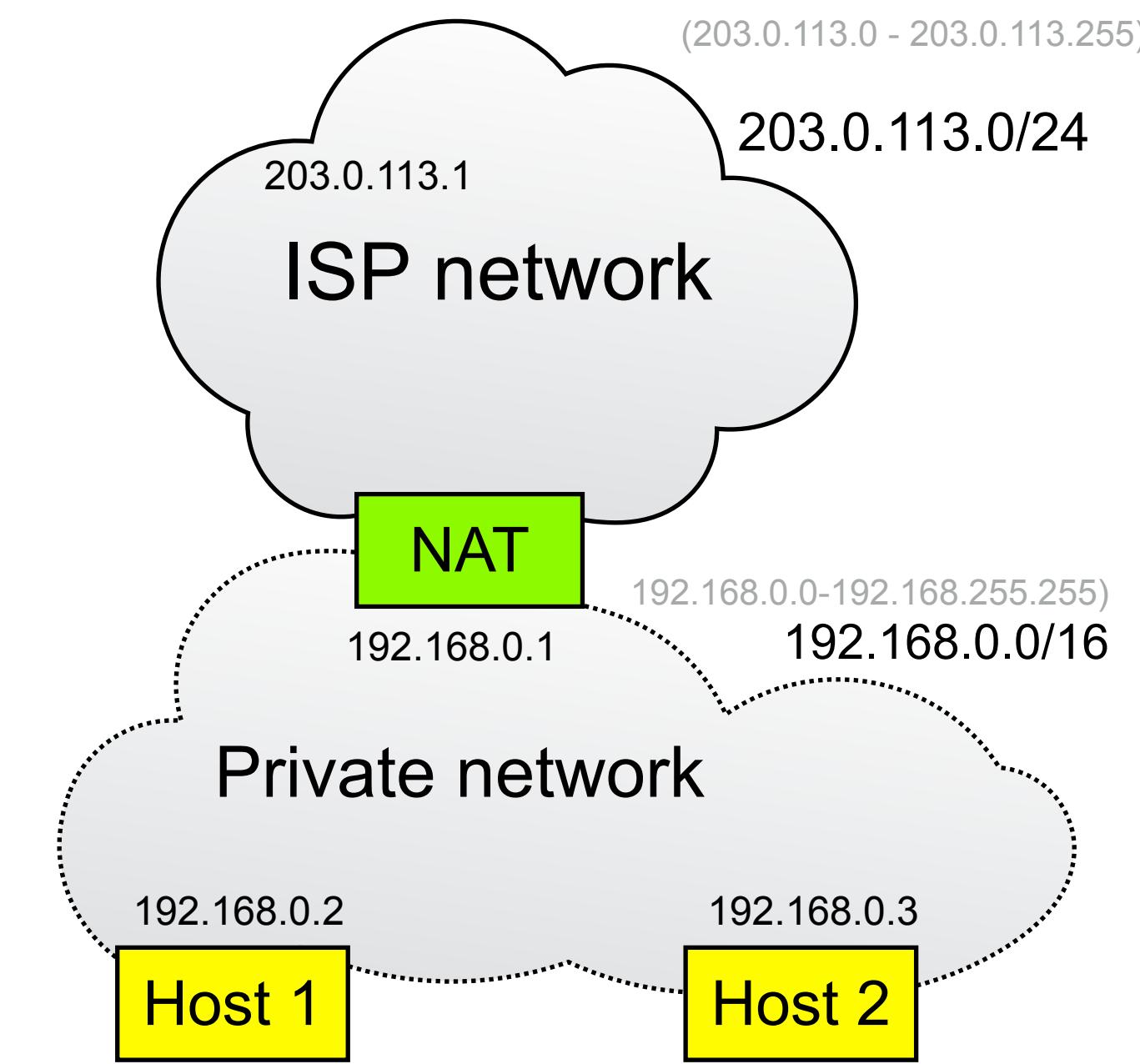
# Peer-to-peer Connections

- Peer-to-peer connections
- Network Address Translation
- Private IPv4 address ranges

# Problems due to Network Address Translation

- Problems due to NAT
- Why use NAT?
- Implications of NAT

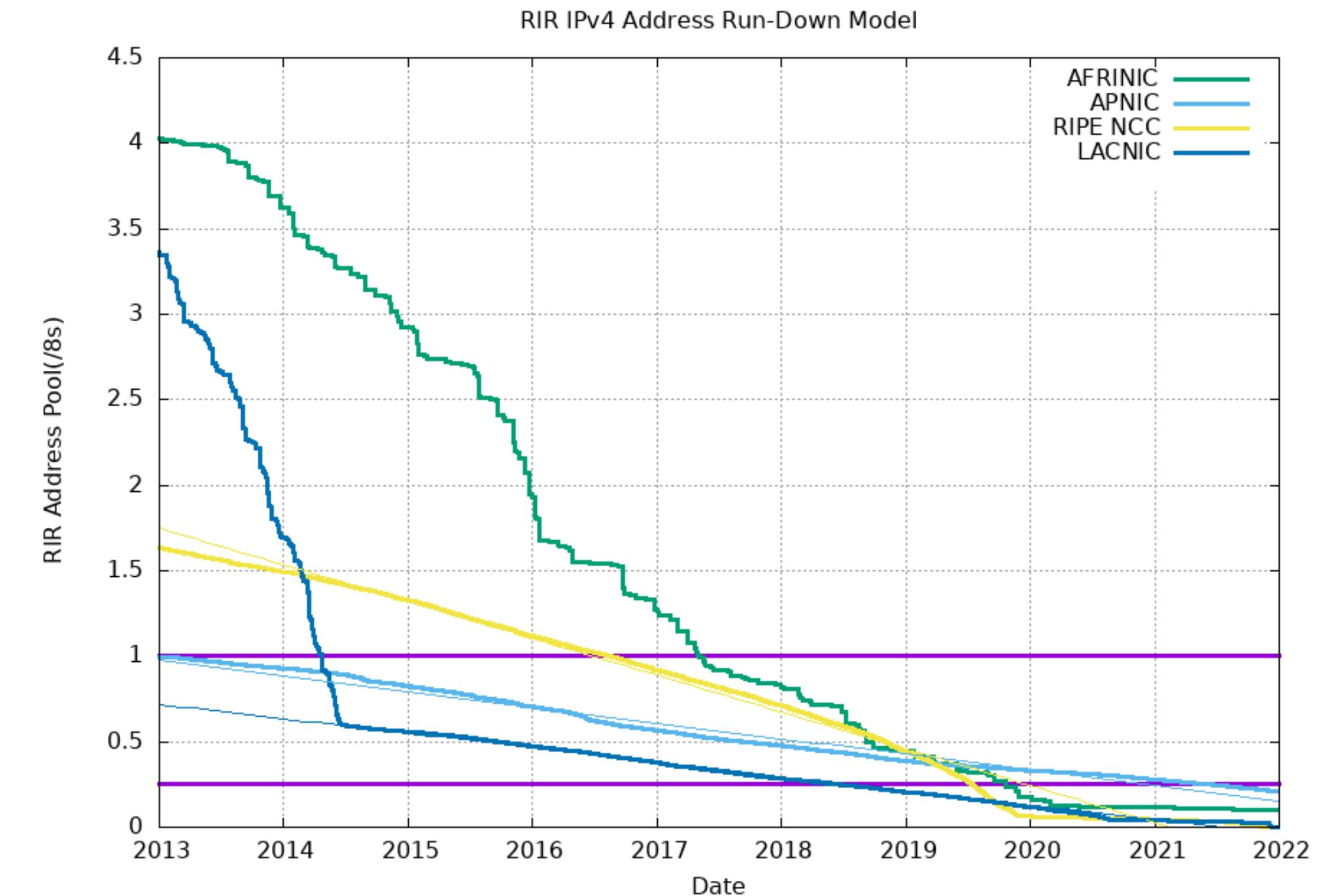
# NAT Routers Encourage Centralisation



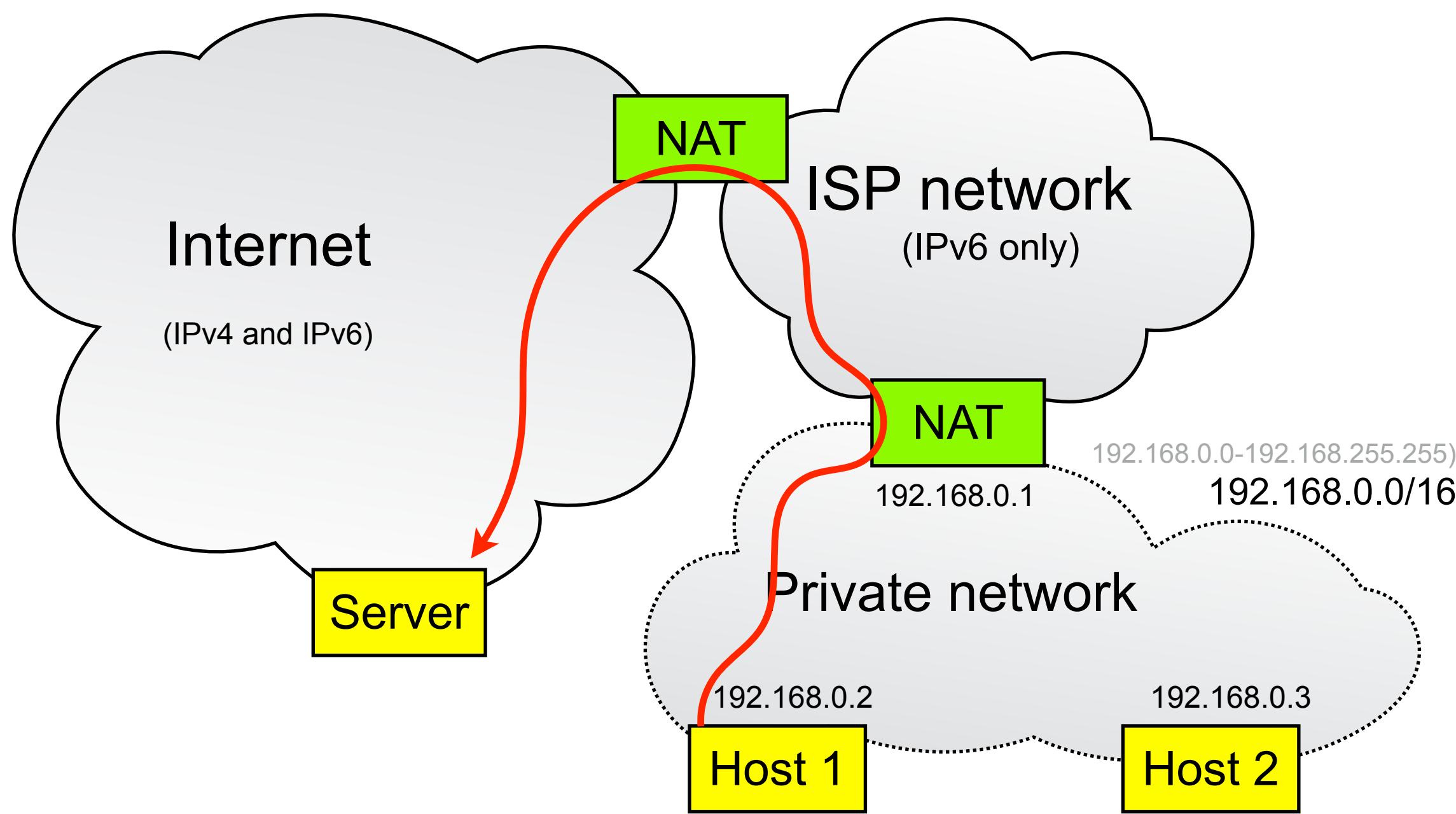
- Client-server applications with client behind NAT work without changes – web and email
- Client-server applications with server behind NAT fail – need explicit port forwarding
- Peer-to-peer applications fail – complex NAT traversal algorithm needed to connect
- Encourages centralisation of services

# 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



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



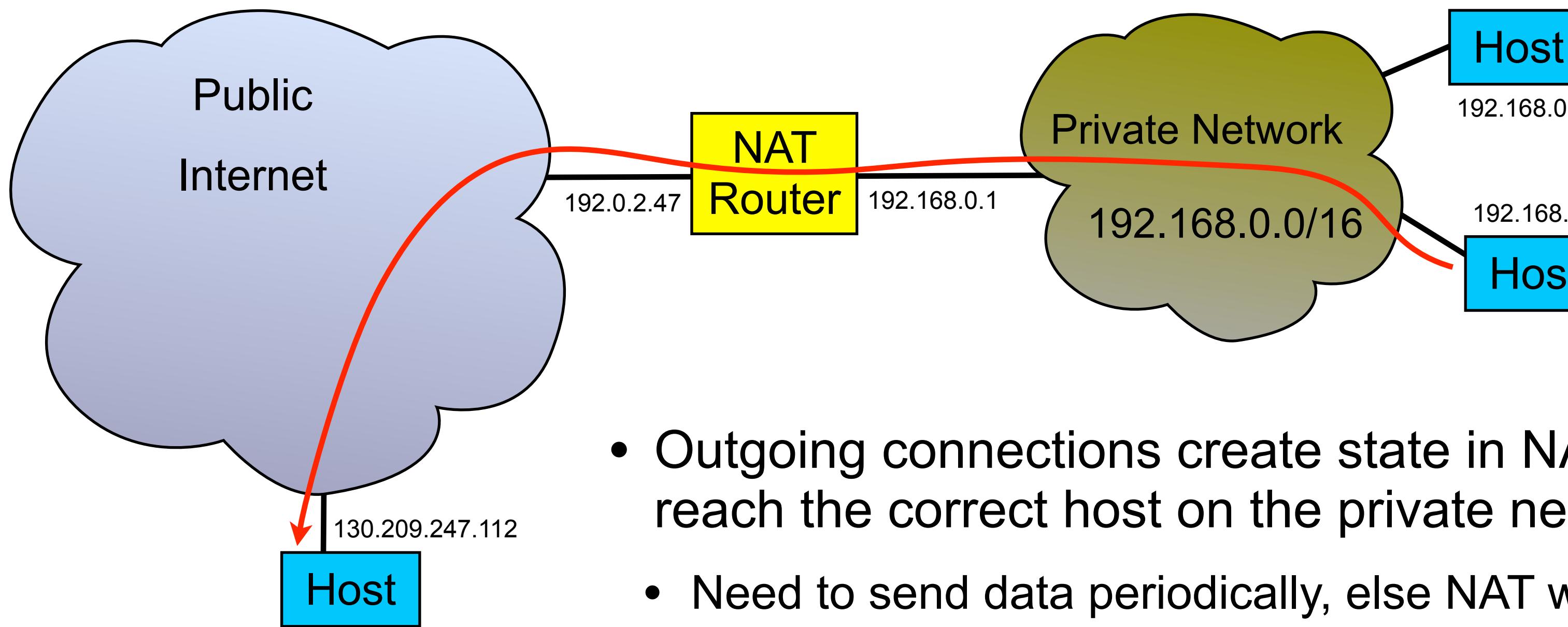
- To translate between IPv4 and IPv6 addresses
- ISP network runs IPv6 only; customers use public IPv6 addresses
- Translate IPv4-to-IPv6 as packets leave private network
  - If destined for IPv4 host on the Internet, translate back to IPv4 on leaving ISP network
  - If destined for IPv6 host on the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

# 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 to ease renumbering

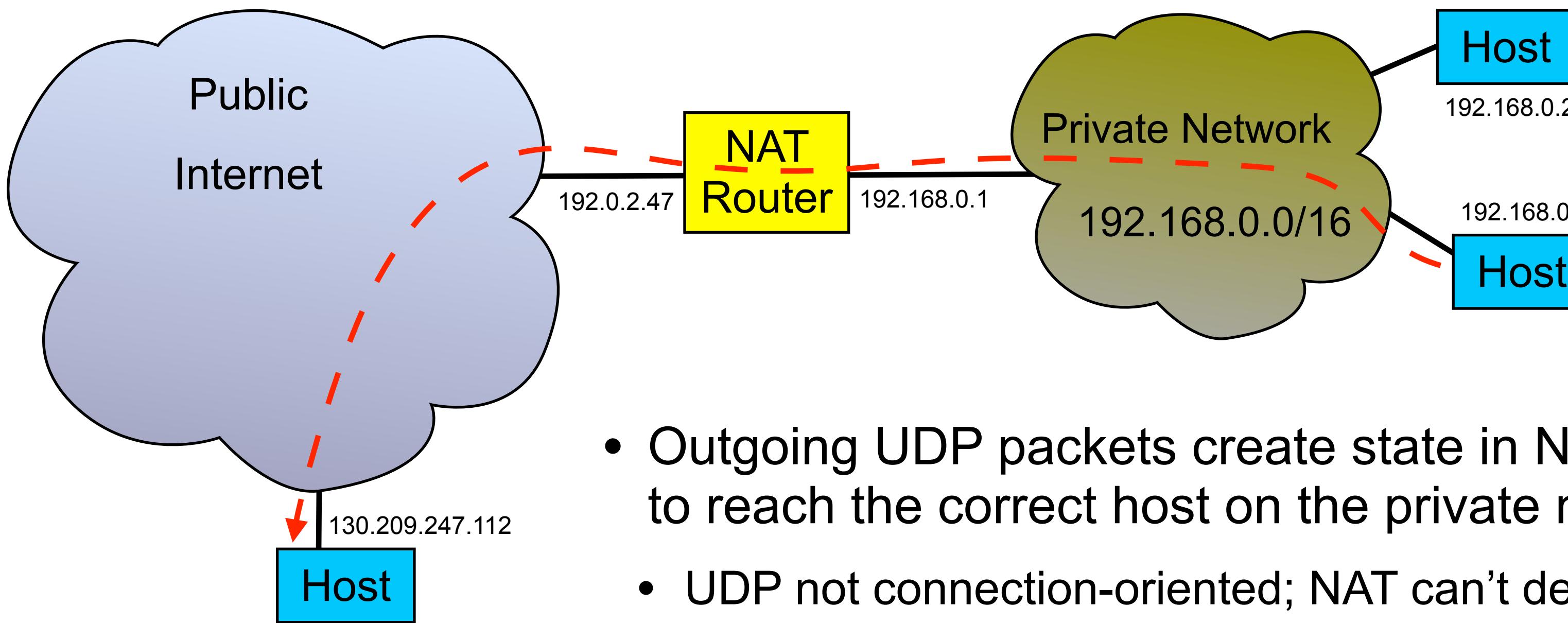
[RFC 6296]

# 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

# 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

# Problems due to Network Address Translation

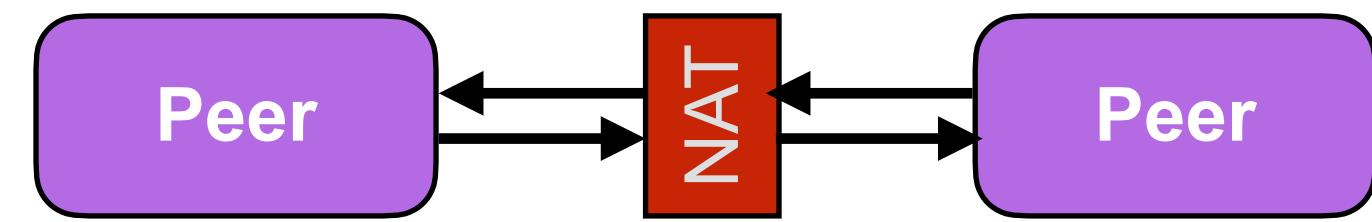
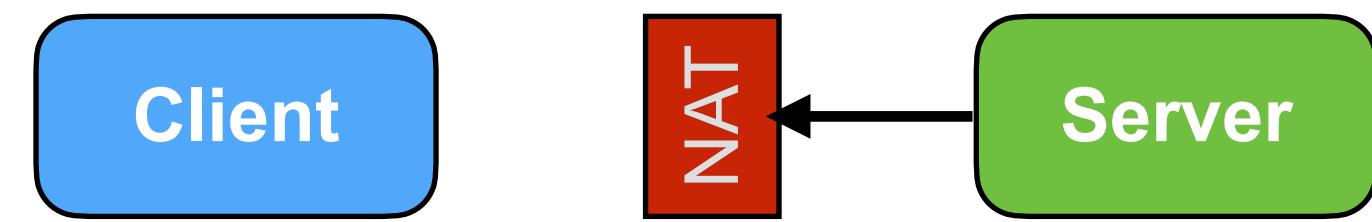
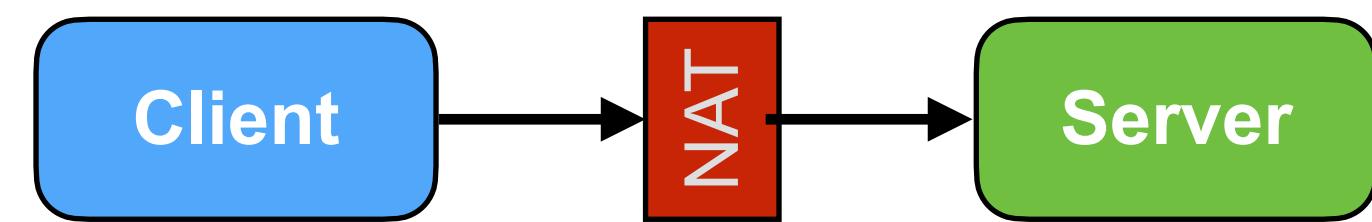
- Problems due to NAT
- Why use NAT?
- Implications of NAT

# NAT Traversal and Peer-to-Peer Connection Establishment

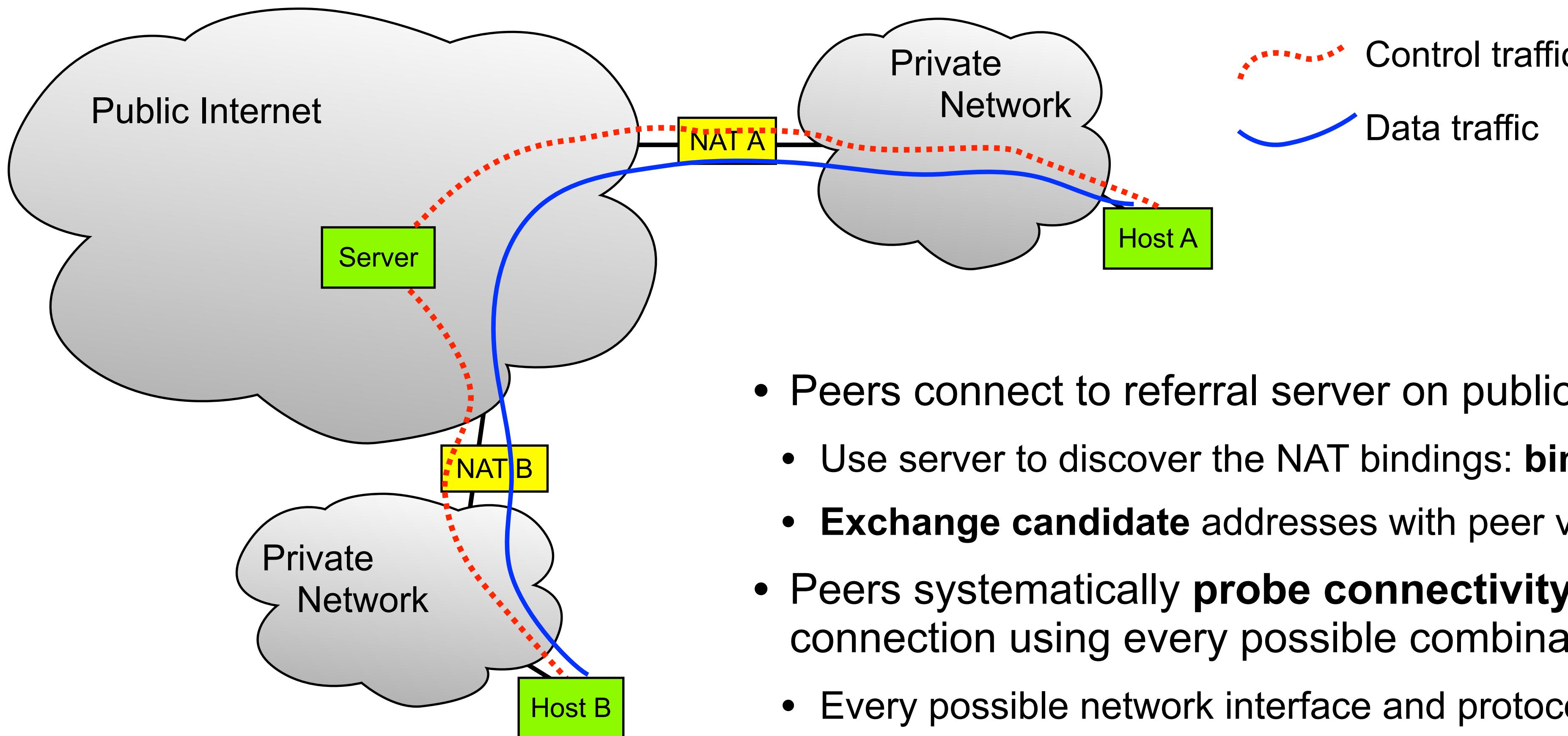
- NAT Traversal Concepts
  - Binding discovery
  - Candidate exchange
  - Probe for connectivity
  - Costs and Limitations

# 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

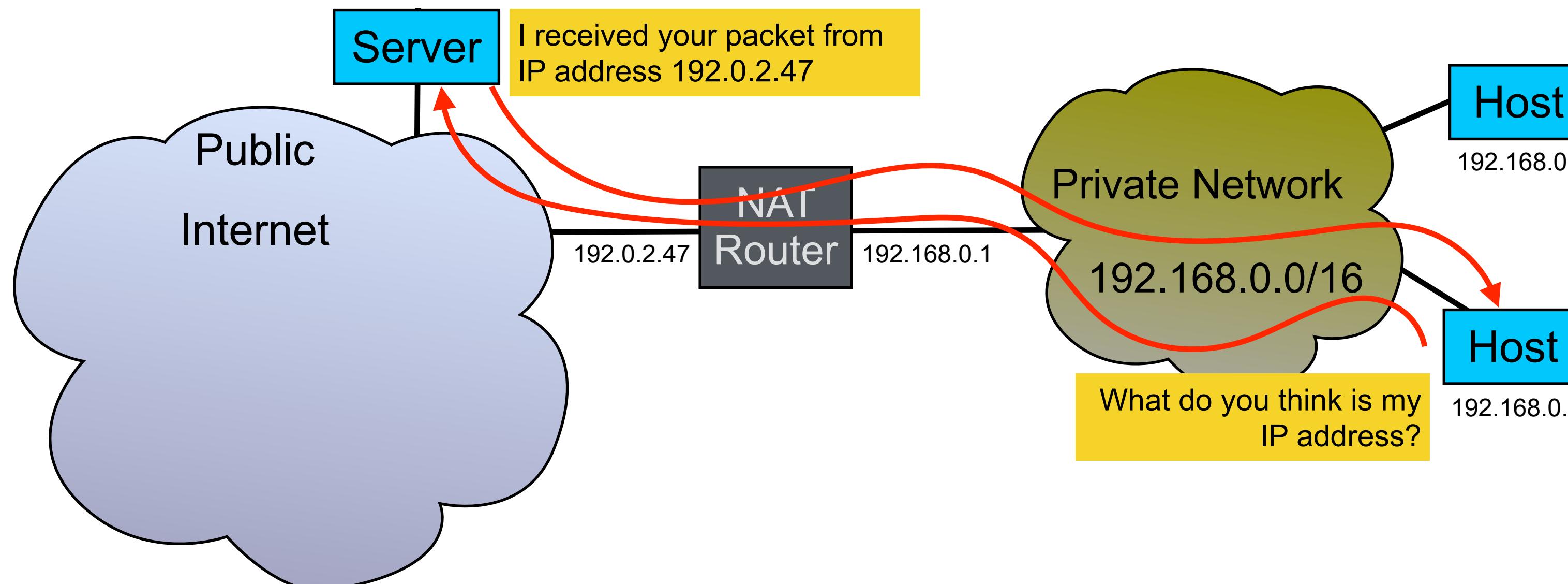


# Peer-to-peer NAT Traversal Concepts



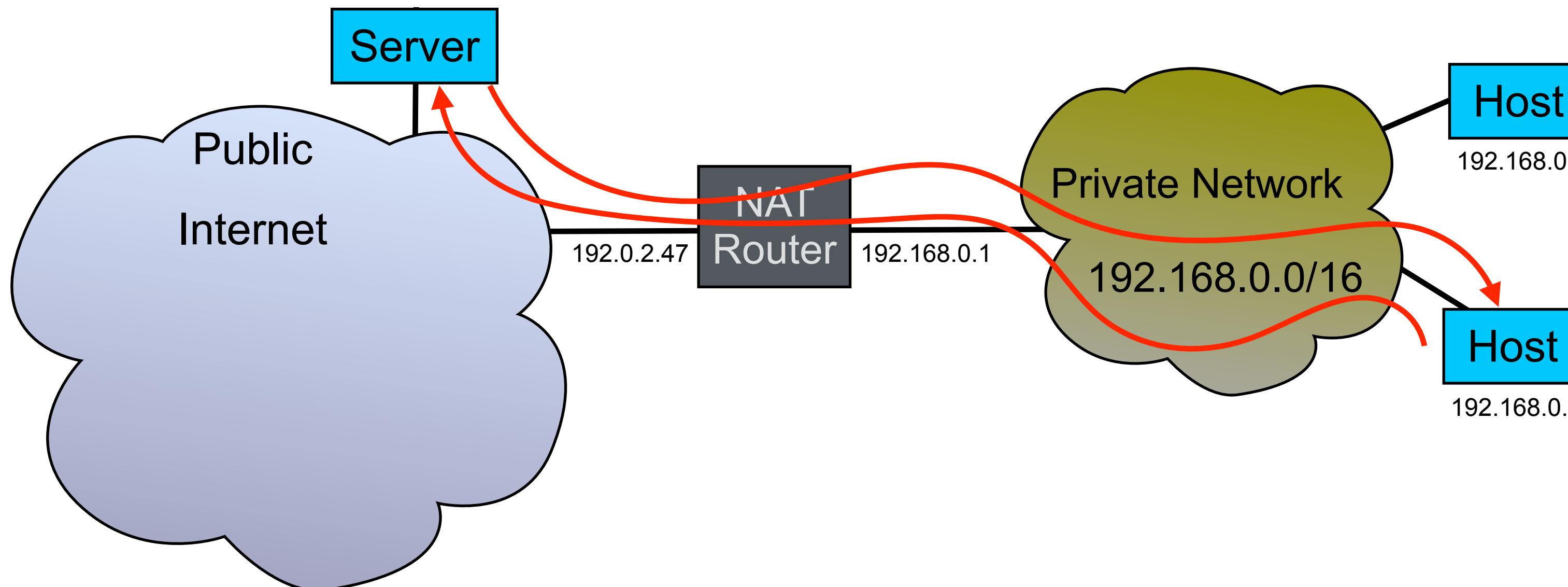
- Peers connect to referral server on public network
  - Use server to discover the NAT bindings: **binding discovery**
  - **Exchange candidate** addresses with peer via the referral server
  - Peers systematically **probe connectivity**, try to establish a connection using every possible combination of addresses
    - Every possible network interface and protocol, mapped and local
    - Complex and generates significant traffic overhead

# Binding Discovery (1/2)



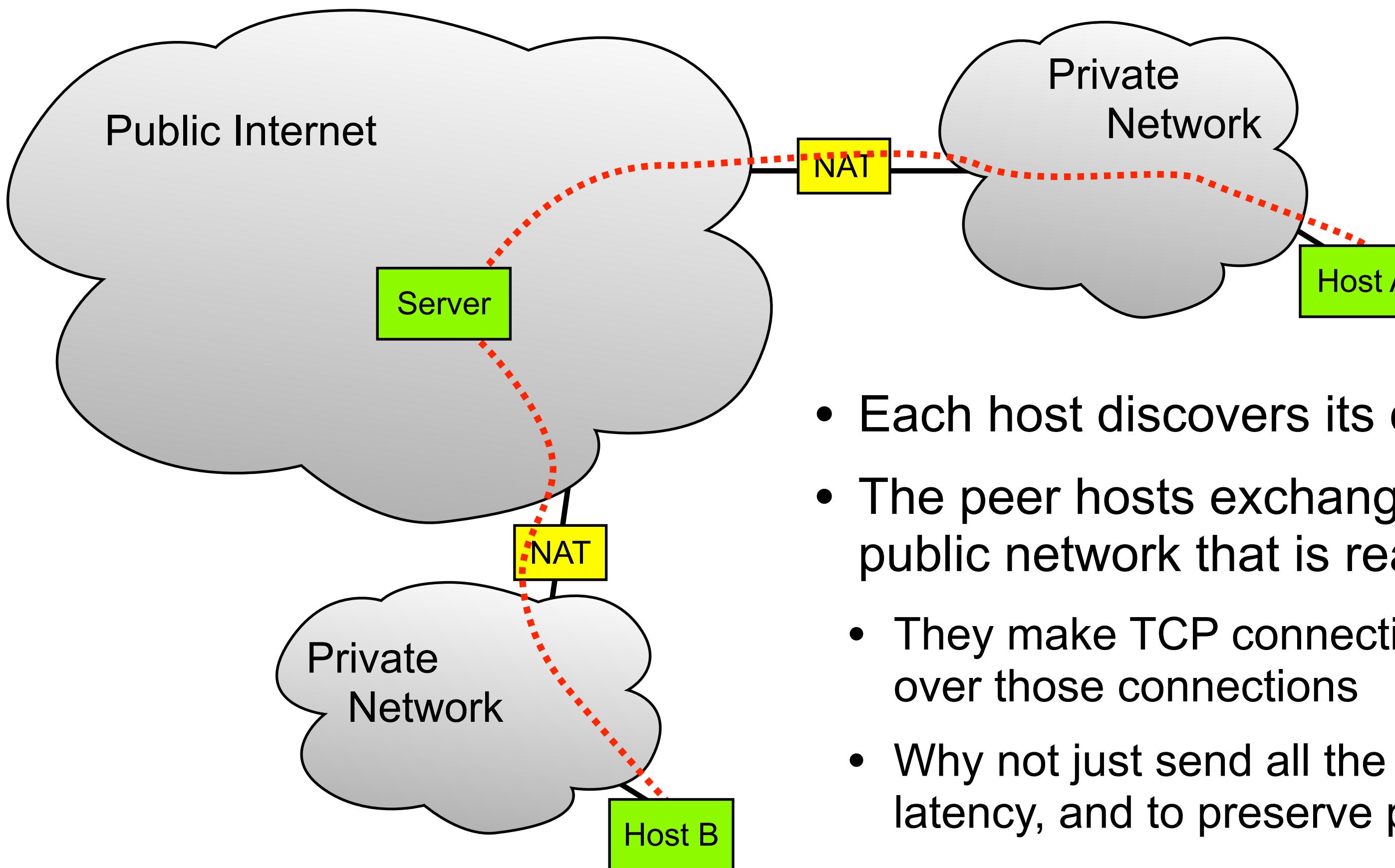
- Packets sent from a host on a private network to a server on the public network will have source IP address and port translated
- The server can see the translated address/port, and send a reply back to the host telling it the address its packets appear to come from – the server reflexive address
- Known as **NAT binding discovery**
- The **STUN Protocol** (Session Traversal Utilities for NAT) is a standard binding discovery mechanism <https://datatracker.ietf.org/doc/rfc8489/>

# Binding Discovery (2/2)



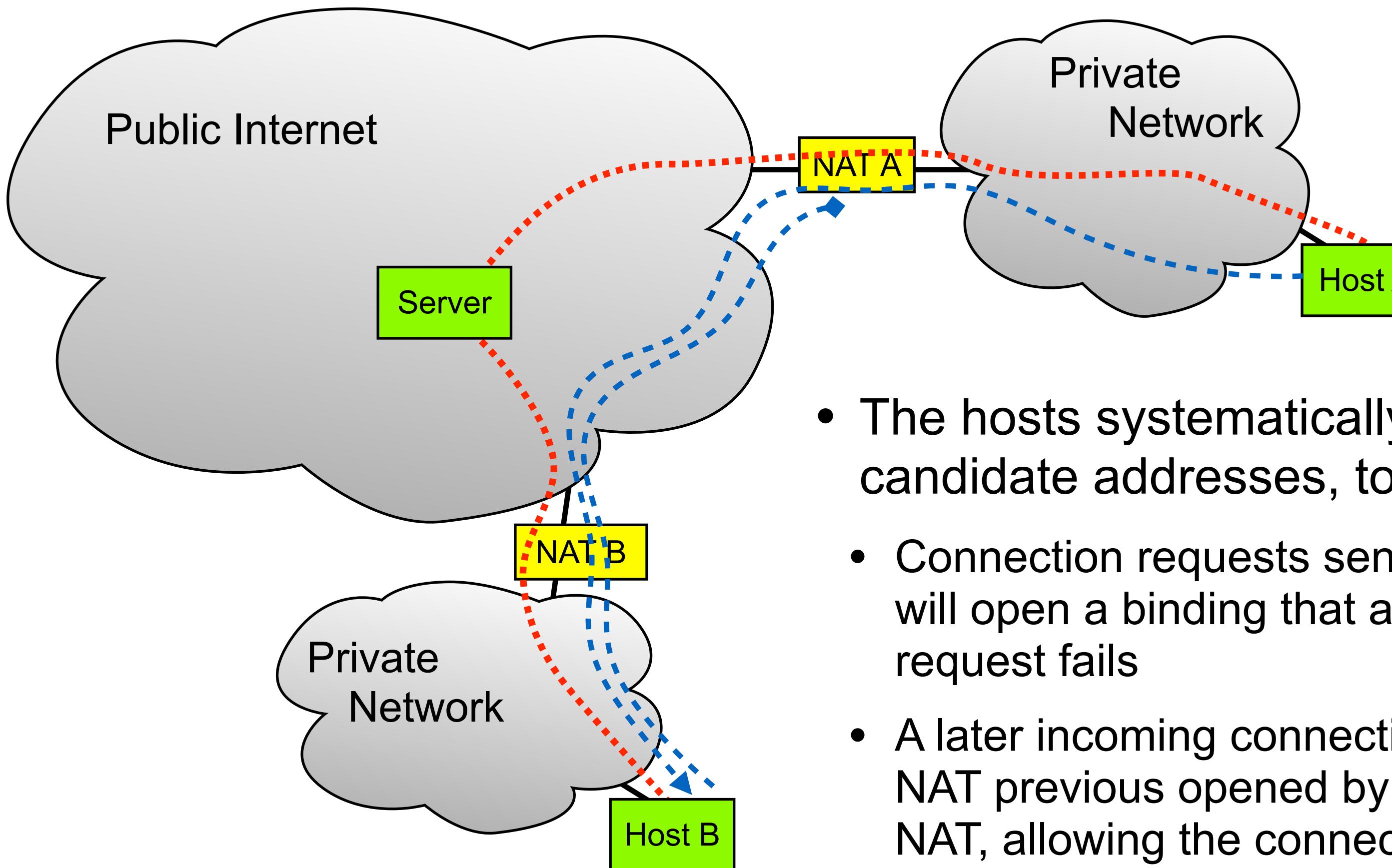
- Host attempts to find every possible **candidate** IP address, on which it might be reachable
  - The IPv4 and IPv6 addresses of each of its interfaces (Wi-Fi, Ethernet, 4G, ...)
  - For each address, any server reflexive addresses discovered using STUN from that address
  - Any relayed addresses (e.g.,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

# Candidate Exchange



- 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

# Probe for connectivity: 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 previously 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** describes how to do this [RFC 8445]

# 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

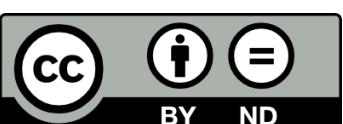
# Connection Establishment in a Fragmented Network

- Client-server connection establishment
- Impact of TLS and IPv6 on connection establishment
- Peer-to-peer connections, NATs, and NAT traversal



# Secure Communications

Networked Systems (H)  
Lecture 3



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Lecture Outline

- The Need for Secure Communications
- Principles of Secure Communication
- Transport Layer Security (TLS) v1.3
- Discussion

# The Need for Secure Communications

- Network Traffic Monitoring
- Protecting Privacy
- Protecting Message Integrity
- Preventing Protocol Ossification

# Numerous Organisations Monitor Internet traffic

- Governments, intelligence agencies, and law enforcement
  - Good reasons to monitor *some* traffic
  - Edward Snowden showed pervasive monitoring of *all* traffic – no reason to believe such monitoring has now stopped
- Business “Your call may be monitored for quality and training purposes”
  - Regulatory requirements to record some traffic – e.g., banking
  - To profile customers for advertising
- Network operators
  - To support network operations and trouble-shooting
  - To profile customers for advertising
- Criminals and malicious users
  - Steal data and user credentials; identity theft; active attacks



Edward Snowden

S. Farrell and H. Tschofenig, “Pervasive Monitoring is an Attack”, IETF, May 2014, RFC 7258  
<https://datatracker.ietf.org/doc/rfc7258/>

K. Moriarty and A. Morton, “Effect of Pervasive Encryption on Operators”, IETF, July 2018, RFC 8404  
<https://datatracker.ietf.org/doc/rfc8404/>

G. Fairhurst and C. Perkins, “Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols”, IETF, July 2021, RFC 9065, <https://datatracker.ietf.org/doc/rfc9065/>

# Protecting Privacy

- Mechanisms that protect privacy against malicious attackers will also prevent benign monitoring
- No known way to stop criminals and malicious attackers from accessing private data that doesn't also stop law enforcement



Edward Snowden

S. Farrell and H. Tschofenig, "Pervasive Monitoring is an Attack", IETF, May 2014, RFC 7258  
<https://datatracker.ietf.org/doc/rfc7258/>

K. Moriarty and A. Morton, "Effect of Pervasive Encryption on Operators", IETF, July 2018, RFC 8404  
<https://datatracker.ietf.org/doc/rfc8404/>

G. Fairhurst and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", IETF, July 2021, RFC 9065, <https://datatracker.ietf.org/doc/rfc9065/>

# Protecting Message Integrity

- Numerous organisations may want to change messages in transit
  - Governments, intelligence agencies, and law enforcement
    - Many governments require ISPs to censor or modify DNS responses
    - Many governments require ISPs to censor messages containing certain content
  - Businesses and network operators
    - To enforce legal restrictions on content, terms of service, or to prevent copyright infringement
  - Criminals and malicious users
    - Phishing scams and identity theft
- Mechanisms to protect messages from malicious attackers also prevent benign actors
  - e.g., web browsers deploy DNS-over-HTTPS (DoH) to ensure integrity of DNS responses → protects against phishing attacks that modify DNS replies → stops ISPs modifying DNS responses to prevent access to domains hosting illegal material
- No technical way to distinguish beneficial integrity violations from malicious attacks

# Preventing Protocol Ossification

- Network operators deploy **middleboxes** to monitor and/or modify traffic:
  - NATs, firewalls, monitors, traffic shaping, filtering, etc.
  - Some are necessary and beneficial, others are less so
- Middleboxes must understand the protocols used:
  - e.g., NAT has to know where to find IP addresses and ports in packets, if it is to translate them
  - e.g., traffic shapers, that limit TCP throughput for users exceeding their monthly bandwidth use cap, need to parse TCP packets and know how to change them to influence congestion control
- This can lead to **protocol ossification** – cannot change end-to-end protocols, because doing so interacts poorly with middleboxes that don't understand the changes
- The more of a protocol that is encrypted, the easier it is to change – since middleboxes cannot understand or modify the data – but the harder it is for middleboxes to provide useful services

Stream: Internet Architecture Board (IAB)  
RFC: 9170  
Category: Informational  
Published: December 2021  
ISSN: 2070-1721  
Authors: M. Thomson T. Pauly

**RFC 9170**  
**Long-Term Viability of Protocol Extension Mechanisms**

**Abstract**  
The ability to change protocols depends on exercising the extension and version-negotiation mechanisms that support change. This document explores how regular use of new protocol features can ensure that it remains possible to deploy changes to a protocol. Examples are given where lack of use caused changes to be more difficult or costly.

M. Thomson and T. Pauly, “Long-term Viability of Protocol Extension Mechanisms”, Internet Architecture Board, December 2021, RFC 9170.  
<https://datatracker.ietf.org/doc/rfc9170/>

# Security and Policy: Encryption and Key Escrow

- Strong reasons to protect privacy, prevent modification of data in transit, and prevent protocol ossification using end-to-end encryption and authentication
- But doing so can complicate the job of law enforcement and those performing legitimate network monitoring – numerous proposals for key escrow, targeted weakening of end-to-end encryption, and other forms of exceptional access
- Dialogue between engineers, protocol designers, operators, policy makers, and law enforcement crucial – to understand constraints and concerns

**Keys Under Doormats:**  
MANDATING INSECURITY BY REQUIRING GOVERNMENT ACCESS TO ALL DATA AND COMMUNICATIONS

Harold Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matthew Blaze, Whitfield Diffie, John Gilmore, Matthew Green, Peter G. Neumann, Susan Landau, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Michael Specter, Daniel J. Weitzner

**Abstract**

Twenty years ago, law enforcement organizations lobbied to require data and communication services to engineer their products to guarantee law enforcement access to all data. After lengthy debate and vigorous predictions of enforcement channels “going dark,” these attempts to regulate the emerging Internet were abandoned. In the intervening years, innovation on the Internet flourished, and law enforcement agencies found new and more effective means of accessing vastly larger quantities of data. Today we are again hearing calls for regulation to mandate the provision of exceptional access mechanisms. In this report, a group of computer scientists and security experts, many of whom participated in a 1997 study of these same topics, has convened to explore the likely effects of imposing extraordinary access mandates.

We have found that the damage that could be caused by law enforcement exceptional access requirements would be even greater today than it would have been 20 years ago. In the wake of the growing economic and social cost of the fundamental insecurity of today’s Internet environment, any proposals that alter the security dynamics online should be approached with caution. Exceptional access would force Internet system developers to reverse “forward secrecy” design practices that seek to minimize the impact on user privacy when systems are breached. The complexity of today’s Internet environment, with millions of apps and globally connected services, means that new law enforcement requirements are likely to introduce unanticipated, hard to detect security flaws. Beyond these and other technical vulnerabilities, the prospect of globally deployed exceptional access systems raises difficult problems about how such an environment would be governed and how to ensure that such systems would respect human rights and the rule of law.

H. Abelson, *et al.*, “Keys under doormats: Mandating insecurity by requiring government access to all data and communications”, MIT Computer Science and Artificial Intelligence Lab, technical report MIT-CSAIL-TR-2015-026, July 2015. <http://dspace.mit.edu/handle/1721.1/97690>

# Security and Policy: End-system Content Monitoring

- If data is encrypted end-to-end, protecting data in transit from malicious attacks, should law enforcement be able to monitor traffic at the end-points?
- Does the answer change if both end-points are personal devices vs. if one of the end-points is a data centre?
- How do the expectations for privacy, law enforcement access, and abuse protection differ between one-to-one conversations, group conversations, social networks, and other situations?

Bugs in our Pockets:  
The Risks of Client-Side Scanning

Hal Abelson    Ross Anderson    Steven M. Bellovin  
Josh Benaloh    Matt Blaze    Jon Callas    Whitfield Diffie  
Susan Landau    Peter G. Neumann    Ronald L. Rivest  
Jeffrey I. Schiller    Bruce Schneier    Vanessa Teague  
Carmela Troncoso

October 15, 2021

arXiv:2110.07450v1 [cs.CR] 14 Oct 2021

**Executive Summary**

Our increasing reliance on digital technology for personal, economic, and government affairs has made it essential to secure the communications and devices of private citizens, businesses, and governments. This has led to pervasive use of cryptography across society. Despite its evident advantages, law enforcement and national security agencies have argued that the spread of cryptography has hindered access to evidence and intelligence. Some in industry and government now advocate a new technology to access targeted data: *client-side scanning* (CSS). Instead of weakening encryption or providing law enforcement with backdoor keys to decrypt communications, CSS would enable on-device analysis of data in the clear. If targeted information were detected, its existence and, potentially, its source, would be revealed to the agencies; otherwise, little or no information would leave the client device. Its proponents claim that CSS is a solution to the encryption versus public safety debate: it offers privacy—in the sense of unimpeded end-to-end encryption—and the ability to successfully investigate serious crime.

In this report, we argue that CSS neither guarantees efficacious crime prevention nor prevents surveillance. Indeed, the effect is the opposite. CSS by its nature creates serious security and privacy risks for all society while the assistance it can provide for law enforcement is at best problematic. There are multiple ways in which client-side scanning can fail, can be evaded, and can be abused.

Its proponents want CSS to be installed on all devices, rather than installed covertly on the devices of suspects, or by court order on those of ex-offenders. But universal deployment threatens the security of law-abiding citizens as well as law-breakers. Technically, CSS allows end-to-end encryption, but this is moot if the

1

H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze,  
J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. Rivest,  
J. I. Schiller, B. Schneier, V. Teague, and C. Troncosco.  
“Bugs in our pockets: The risks of client-side scanning”.  
October 2021. <https://arxiv.org/abs/2110.07450>

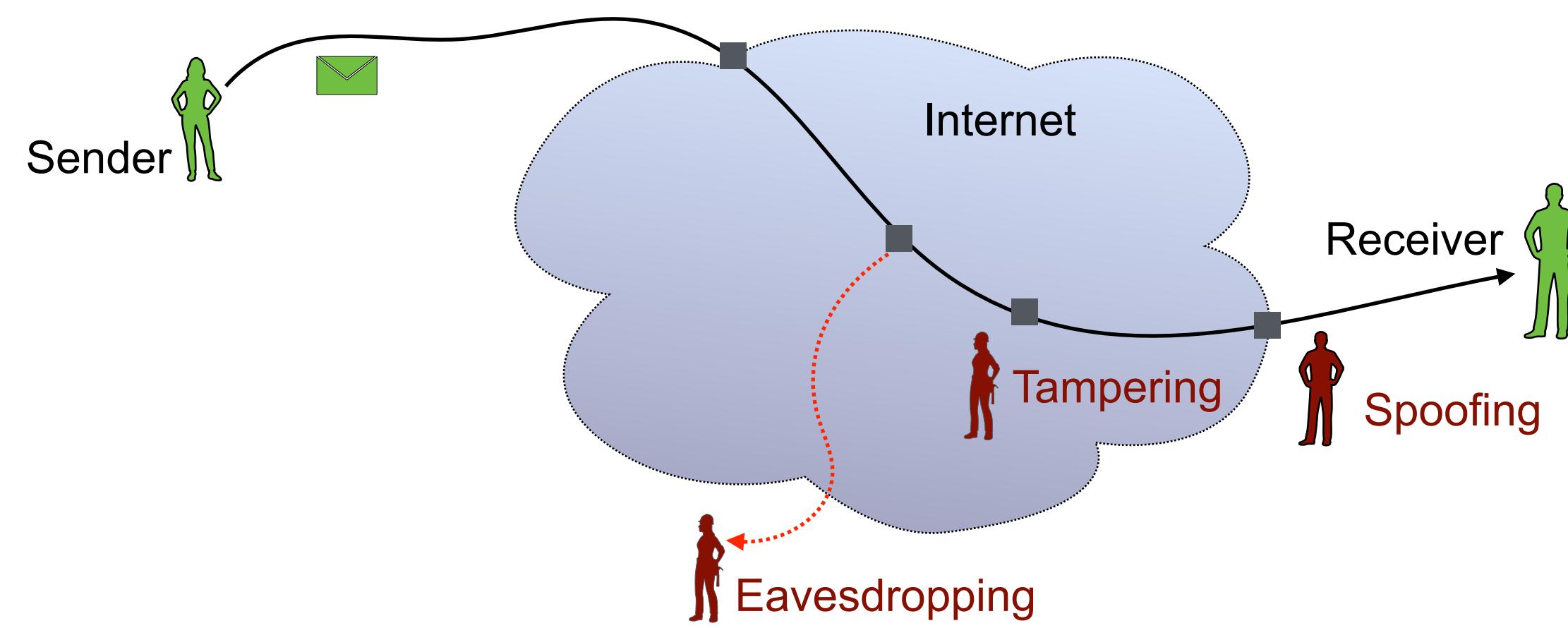
# The Need for Secure Communications

- Network Traffic Monitoring
- Protecting Privacy
- Protecting Message Integrity
- Preventing Protocol Ossification

# Principles of Secure Communication

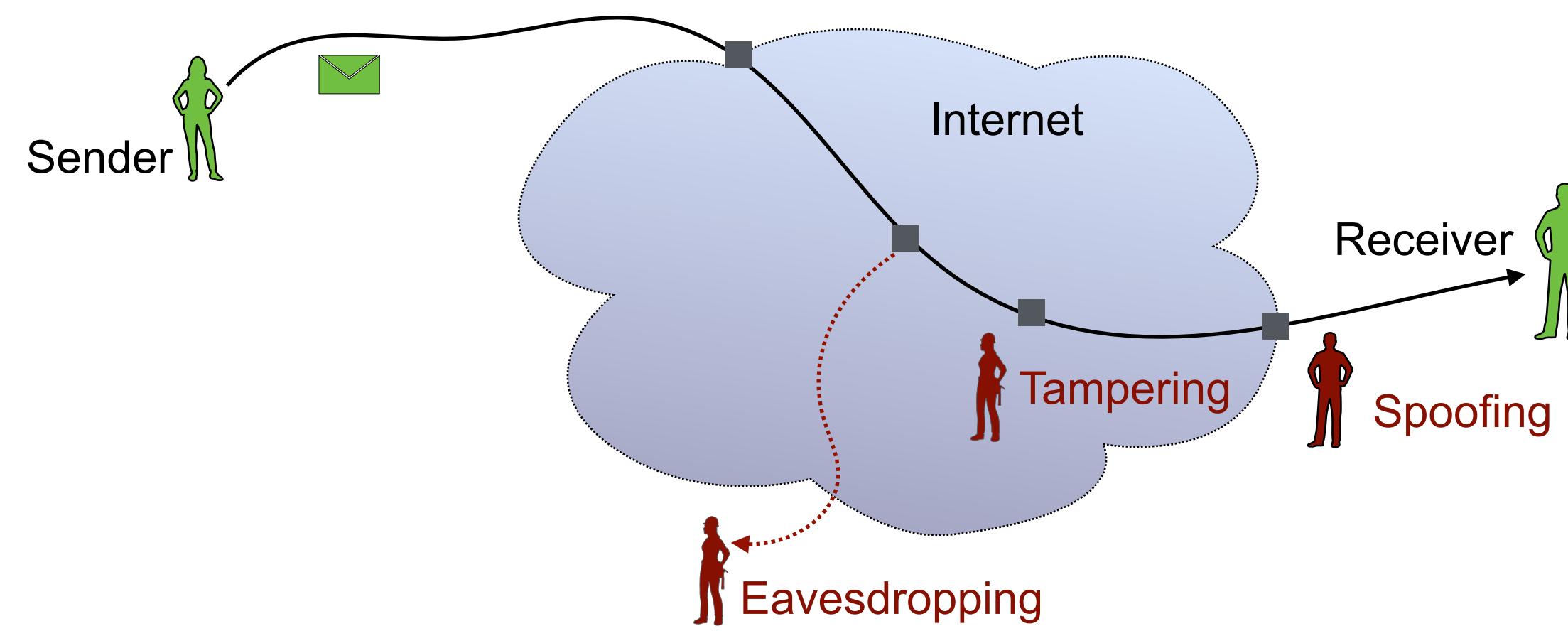
- Ensuring Confidentiality
- Authenticating Messages
- Validating Identity

# Goals of Secure Communication



- Goal: deliver message from sender to receiver
  - Avoid eavesdropping → encrypt to provide confidentiality
  - Avoid tampering → authenticate to ensure the message is not modified in transit
  - Avoid spoofing → validate identity of sender

# How to Provide Confidentiality? (1/2)



- Data traversing the network can be read by any device on the path
  - Can eavesdrop on packets as they traverse a link
  - Configure a switch or router to snoop on data as it's forwarded between links
  - The network operator can *always* do this; if their network has been compromised, maybe so can others

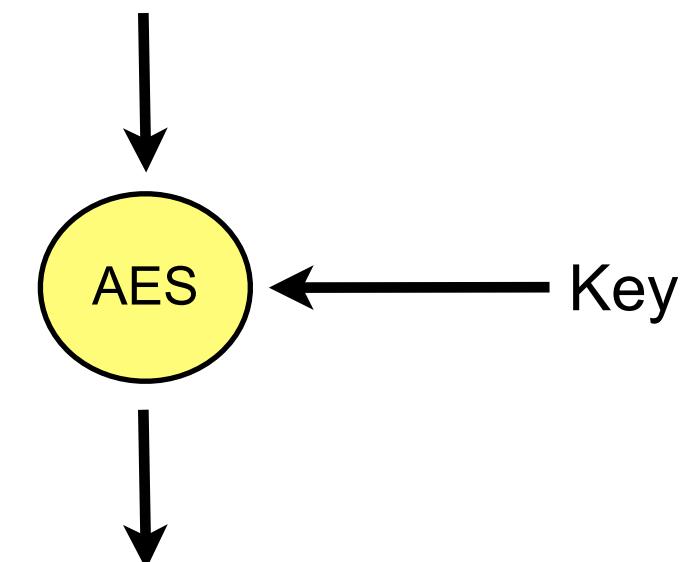
# How to Provide Confidentiality? (2/2)

- Data can always be read → use **encryption** to make it useless if intercepted
- Two basic approaches
  - Symmetric cryptography
    - Advanced Encryption Standard (AES)
  - Public key cryptography
    - The Diffie-Hellman algorithm
    - The Rivest-Shamir-Adleman (RSA) algorithm
    - Elliptic curve-based algorithms
  - Complex mathematics – will not attempt to describe

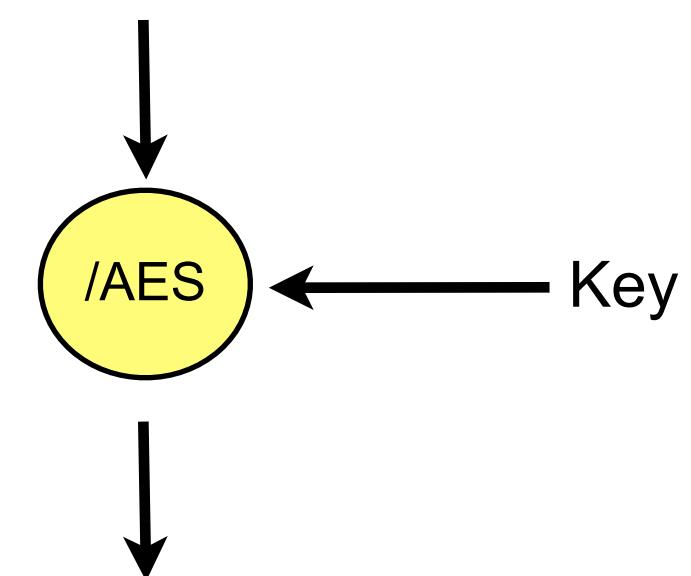
# Symmetric Cryptography

- Symmetric encryption converts plain text into cipher-text
  - A secret **key** control the encryption and decryption process
    - Same key used to encrypt as is used to decrypt
    - Provided the key is secret and only known to sender and receiver, the conversation is secure – problem: **how to securely distribute the key?**
  - Very fast – suitable for bulk encryption
  - Encryption and decryption algorithms are public
    - US Advanced Encryption Standard (AES) is widely used – based on Rijndael algorithm, developed by Vincent Rijmen and Joan Daemen  
<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

“It was a bright cold day in April, and the clocks were striking thirteen.”



rx27qrh1M/Pd5UnkpqTuXnJBZecF1  
bP5Xd8ouyAWgCLxZJUD951SaxusX5  
bj0O2P9XkVGGHmMoqByZxu2pU+cc1  
sERzuHKxc

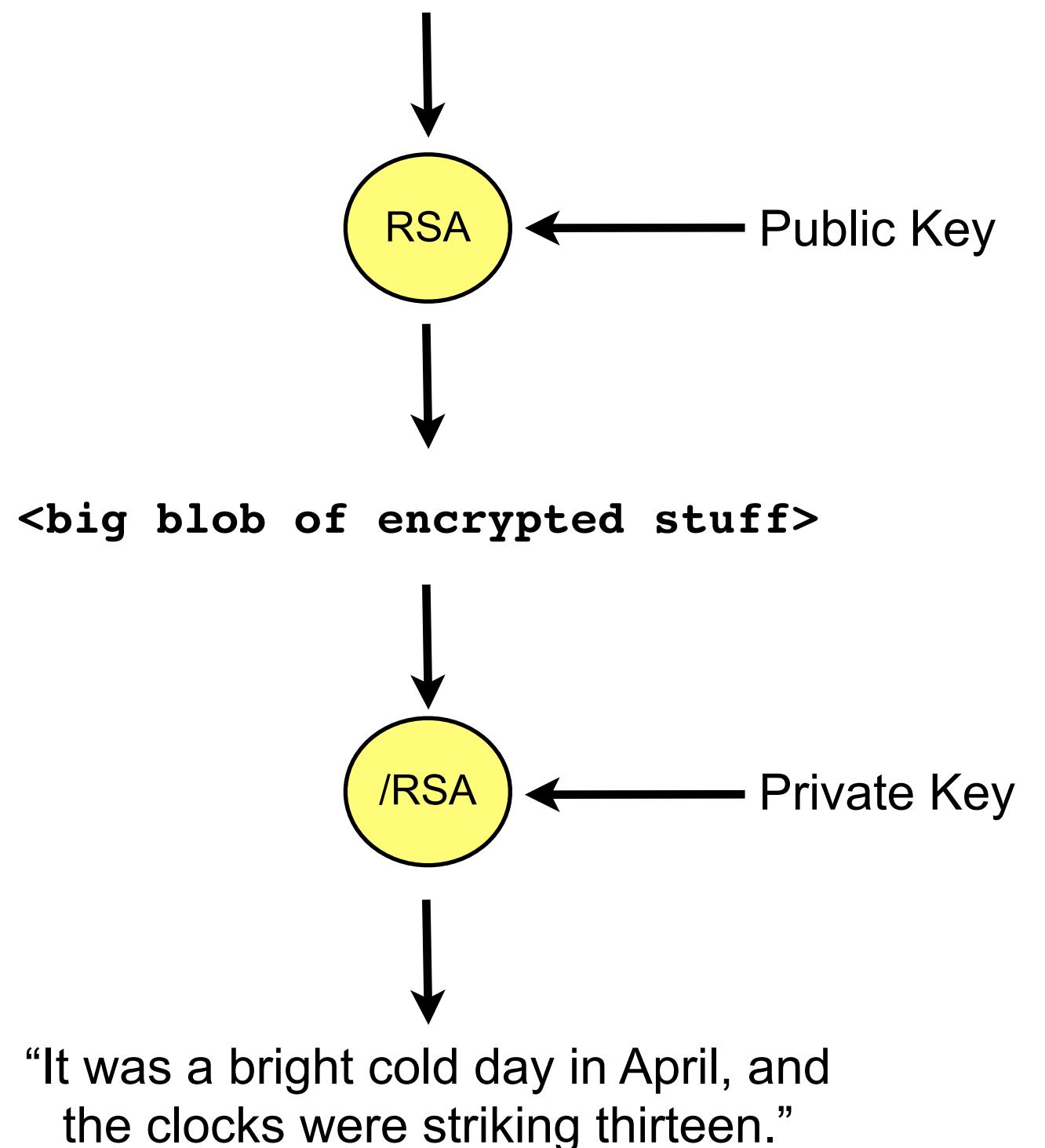


“It was a bright cold day in April, and the clocks were striking thirteen.”

# Public Key Cryptography

- Public key encryption also converts plain text into cipher-text
  - Again, the algorithms are public:
    - Diffie–Hellman
    - Rivest–Shamir–Adleman (RSA)
    - Ephemeral Elliptic Curve Diffie–Hellman
  - Public key algorithms use two related keys:
    - The **public key** for a user is widely distributed
    - **The corresponding private key must be kept secret**
    - If one key is used to encrypt, the other key is needed to decrypt the message
  - Solves key distribution problem
    - Look-up the public key of the receiver in a directory
    - Sender uses the public key to encrypt the message → can only be decrypted by the private key
    - If receiver is trusted to keep private key secret, only it can decrypt the message
  - Problem: **very slow** to encrypt and decrypt

“It was a bright cold day in April, and the clocks were striking thirteen.”



# Hybrid Cryptography

- Modern communications use a combination of **both** public-key and symmetric cryptography for security and speed
  - Sender chooses a random value,  $K_s$ , that can be used as key for the symmetric encryption algorithm
  - Sender looks up the receiver's public key,  $K_{pub}$ , uses it to encrypt  $K_s$ , and sends the result to the receiver; receiver uses the corresponding private key,  $K_{priv}$ , to decrypt the message and retrieve  $K_s$
  - Securely transfers  $K_s$  from sender to receiver
    - Public key encryption is very slow, but the key  $K_s$  is small, so this doesn't matter
  - Sender encrypts future messages using symmetric cryptography with key  $K_s$ , receiver also has  $k_s$ , which it uses to decrypt the messages
    - Symmetric cryptography is fast, but requires the key to be exchanged securely
    - The public key algorithm has been used to securely exchange the key
  - Ensures confidentiality of communication with good performance

# Authentication

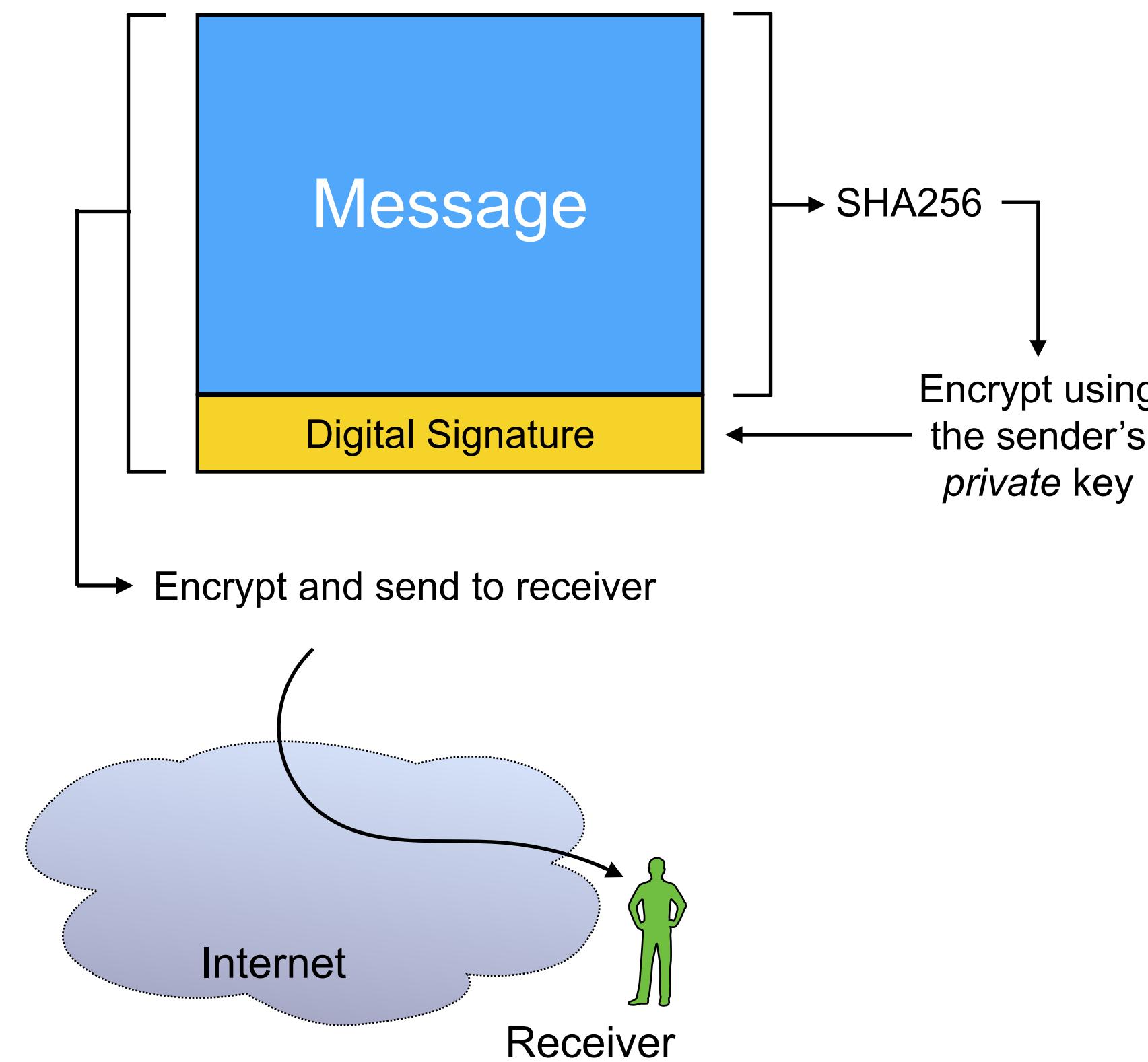
- Encryption can ensure confidentiality – but also need to verify identify of sender and ensure messages have not been modified in transit
  - Generate a **digital signature** to authenticate the message
  - Relies on public key cryptographic and a cryptographic hash

# Cryptographic Hash Functions



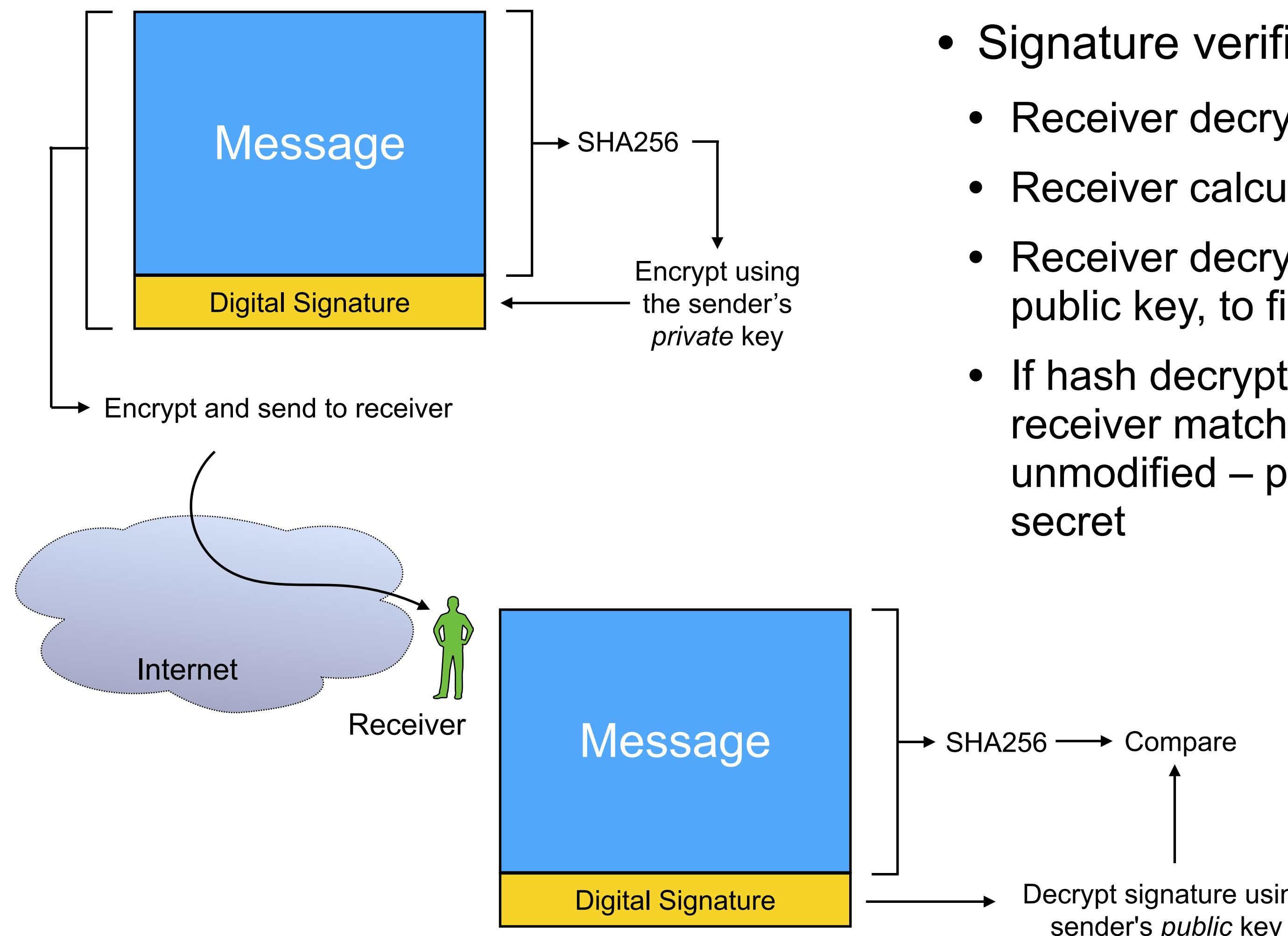
- **Cryptographic hash** takes arbitrary input and produces a fixed length output hash
  - Any change to input generates different output
  - Infeasible to find two inputs that give the same output
  - Calculating a cryptographic hash is fast
  - Reversing a hash, to find the input given only the output, is infeasible
- Many cryptographic hash algorithms exist:
  - Recommendation: SHA256 algorithm
    - <https://tools.ietf.org/html/rfc6234>
  - Older algorithms, e.g., MD5 and SHA1, have known security flaws

# Digital Signatures (1/2)



- Sender generates a digital signature
  - Sender calculates the cryptographic hash of the message
  - Sender encrypts the hash with their own *private key*
    - Anyone can use the sender's public key to decrypt this, but only the sender can have encrypted it (if trusted to keep their private key secret)
  - Attaches encrypted hash to the message
- Message and its digital signature are encrypted and sent to receiver using hybrid encryption

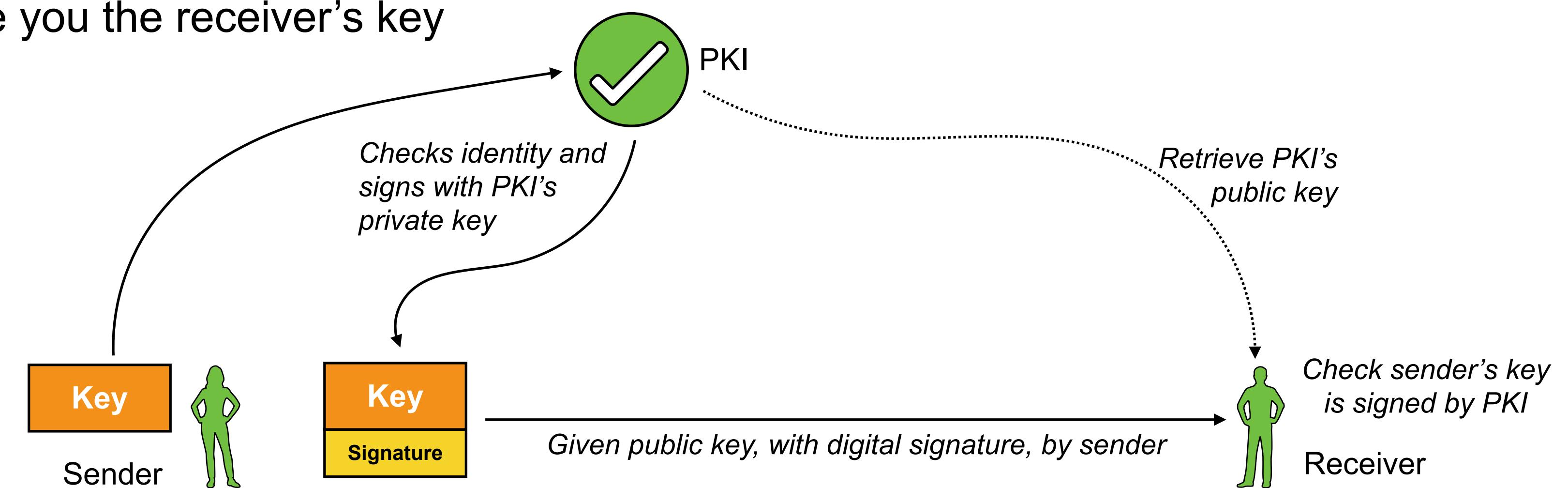
# Digital Signatures (2/2)



- Signature verification process:
  - Receiver decrypts the message
  - Receiver calculate cryptographic hash of the message
  - Receiver decrypts digital signature using sender's public key, to find the hash that the sender calculates
  - If hash decrypted hash and the hash calculated by the receiver match, then the message is authentic and unmodified – provided the sender kept its private key secret

# Trust and Public Key Infrastructure

- How to know what public key corresponds to a particular receiver?
  - The receiver gave you their key in person
  - The receiver sent you their key, authenticated by someone you trust
  - Someone you trust gave you the receiver's key



- A **public key infrastructure** can authenticate keys
  - PKI verifies sender's identity, then adds their digital signature to sender's public key
  - If receiver trusts PKI, can verify the digital signature to confirm identity of sender

# Principles of Secure Communication

- Ensuring Confidentiality
- Authenticating Messages
- Validating Identity

# Transport Layer Security (TLS) v1.3

- What is TLS?
- TLS handshake protocol
- TLS record protocol
- TLS 0-RTT mode
- Limitations

# Transport Layer Security (TLS) v1.3

- TCP is not secure – neither the TCP/IP headers nor the data are encrypted or authenticated
- The **Transport Layer Security** protocol (TLS v1.3) can be used to encrypt and authenticate data carried **within** a TCP connection
  - Official specification:
    - <https://tools.ietf.org/html/rfc8446> – The Transport Layer Security (TLS) Protocol Version 1.3
  - Useful blog posts:
    - <https://ietf.org/blog/tls13/> – Introduction to TLS 1.3 from IETF
    - <https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/> – A detailed look at what's new in TLS 1.3
    - <https://davidwong.fr/tls13/> – The TLS 1.3 specification, redrawn in a readable way with explanatory videos and comments
    - <https://tls13.ulfheim.net> – An annotated packet capture, showing details of a TLS Connection

# TLS v1.3 Goals

The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:

- **Authentication:** The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [[RSA](#)], the Elliptic Curve Digital Signature Algorithm (ECDSA) [[ECDSA](#)], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [[RFC8032](#)]) or a symmetric pre-shared key (PSK).
- **Confidentiality:** Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
- **Integrity:** Data sent over the channel after establishment cannot be modified by attackers without detection.

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, <https://tools.ietf.org/html/rfc8446>

# TLS v1.3 Overview

TLS consists of two primary components:

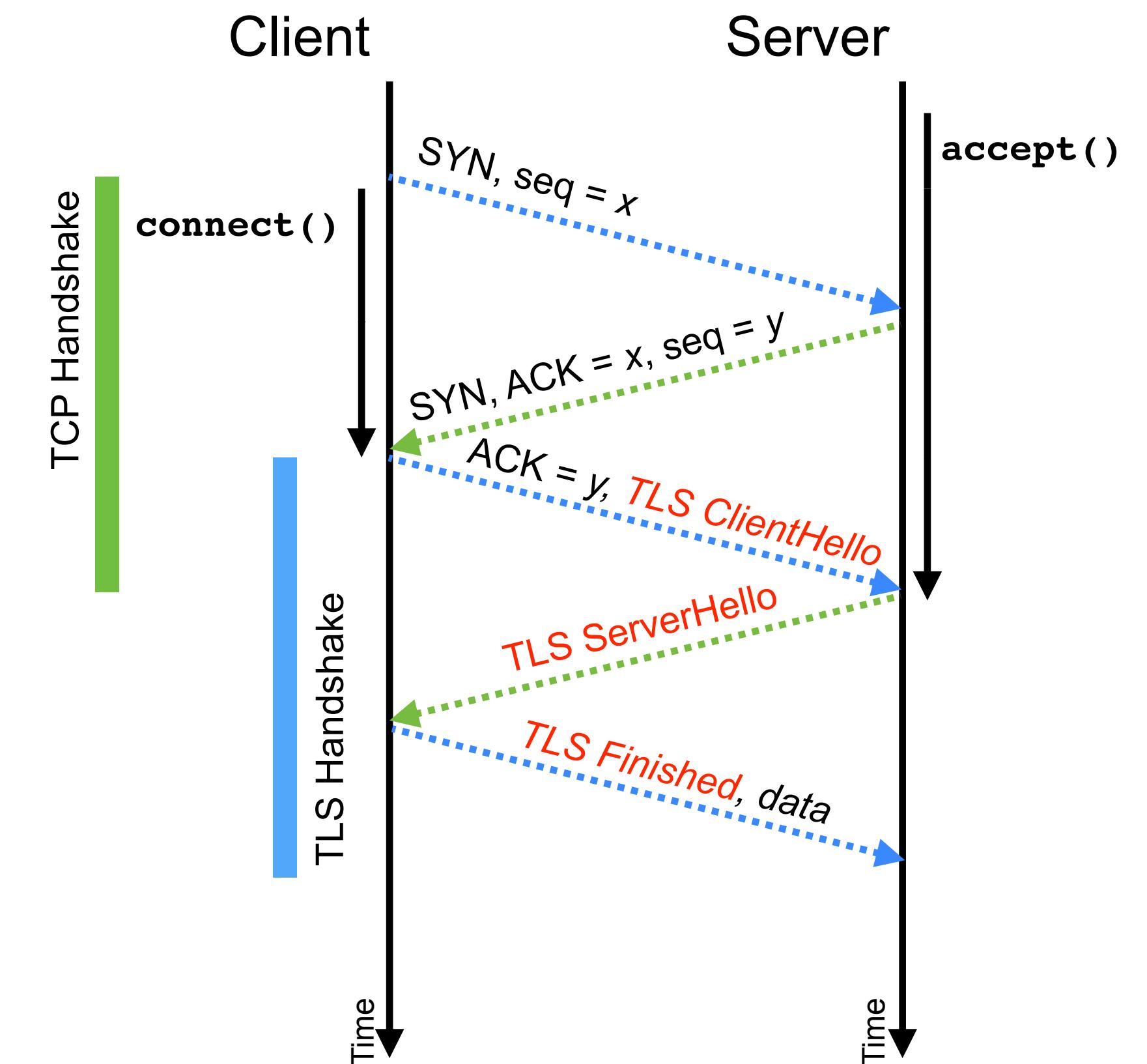
- A **handshake protocol** ([Section 4](#)) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering; an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.
- A **record protocol** ([Section 5](#)) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, <https://tools.ietf.org/html/rfc8446>

- A TCP connection is established
- TLS handshake protocol runs within that TCP connection
  - Authenticates endpoints and agrees on the encryption keys to use
- TLS record protocol then runs over the TCP connection, lets endpoints exchange authenticated and encrypted blocks of data
  - TLS turns the TCP byte stream into a series of records → provides framing

# TLS v1.3 Handshake Protocol

- TCP connection established as usual
  - **SYN** → **SYN+ACK** → **ACK**
- TLS handshake protocol immediately follows
  - TLS **ClientHello** sent with the **ACK**
  - TLS **ServerHello** sent in response
  - TLS **Finished** message concludes, and carries initial secure data record
- Adds 1-RTT to connection establishment



# TLS v1.3 ClientHello

- Key Exchange: Establish shared keying material and select the cryptographic parameters. Everything after this phase is encrypted.

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3",  
IETF, August 2018, RFC 8446, <https://tools.ietf.org/html/rfc8446>

- The **ClientHello** message:
  - Indicates that TLS v1.3 is to be used
    - Signals TLS **v1.2** in the main **ClientHello** message, with an extension header to say “I’m really TLS v1.3” because too many middleboxes break if the TLS version changes – *protocol ossification*
    - Provides the cryptographic algorithms the client supports
    - Provides the name of the server to which the client is connecting
      - If connecting to a web hosting server, it’s likely that more than one site is hosted on the same server, so need to specify which is intended
    - Does not contain any data

# TLS v1.3 ServerHello

- **Server Parameters:** Establish other handshake parameters (whether the client is authenticated, application-layer protocol support, etc.).

The server processes the ClientHello and determines the appropriate cryptographic parameters for the connection. It then responds with its own ServerHello (Section 4.1.3), which indicates the negotiated connection parameters. The combination of the ClientHello and the ServerHello determines the shared keys

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, <https://tools.ietf.org/html/rfc8446>

- The **ServerHello** message:
  - Indicates that TLS v1.3 is to be used
    - It signals TLS **v1.2** in the main **ServerHello** message, and includes an extension header to say “I’m really TLS v1.3” because too many middleboxes break if the TLS version changes – *protocol ossification*
    - Provides cryptographic algorithms selected by the server, from the set the client suggested
    - Provides the server’s public key and digital signature used to verify its identity
    - Does not contain any data

# TLS v1.3 Finished

- **Authentication: Authenticate the server (and, optionally, the client) and provide key confirmation and handshake integrity.**

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, <https://tools.ietf.org/html/rfc8446>

- The **Finished** message:
  - Provides the client's public key
  - Optionally, provides the certificate needed to authenticate the client to the server
  - May contain data sent from client to server

# TLS v1.3 Cryptographic Algorithms

- Client and server use the Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange algorithm
  - Client sends its public key to server in **ClientHello**
  - Server sends its public key to client in **ServerHello**
    - The public keys are *ephemeral* – the client and server both generate new, randomly chosen, private keys for each connection and derive the public keys from them
  - Client and server both combine the two public keys with their own private key to derive the key used for the symmetric cryptography – complex mathematics; will not describe
  - Server provides a certificate (digital signature) to allow the client to verify its identity in the **ServerHello** – client can optionally provide this in **Finished**
- Client proposes symmetric encryption algorithms in **ClientHello**, server picks from them and replies in **ServerHello**
  - Either AES or ChaCha20 symmetric encryption

# TLS v1.3 Record Protocol

- TLS v1.3 splits the data into **records**, each containing  $\leq 2^{14}$  bytes
  - Each record is both encrypted and authenticated; it has a sequence number
  - Can renegotiate encryption keys between records
  - TCP does not preserve record boundaries; TLS adds **framing** so that it does – reading from a TLS connection will block until a complete record is received
- Client and server exchange records – send and receive data – then close connection

# TLS v1.3 0-RTT Mode

- TLS v1.3 usually takes 1-RTT to establish a connection, after TCP connection setup
- If a client and server have previously communicated, they can re-use a key:
  - Server can send additional encryption keys as part of **ServerHello**, that client can use the *next* time it connects to that server
  - On next connection, **ClientHello** message can include data to be delivered to the server, encrypted with that pre-shared key; the **ServerHello** can contain a reply
  - This *0-RTT data* may be delivered more than once, if the **ClientHello** or **ServerHello** messages are duplicated – TLS doesn't stop this
    - Protection against duplicate messages is provided by sequence numbers in the record layer
    - Handshake messages do not have sequence numbers, so duplicated messages can deliver data more than once
  - **Be careful writing applications that use 0-RTT mode**

# Limitations of TLS

- TLS is secure, but has limitations:
  - Does not encrypt server name:
    - Exposes hostname of server to which connection is being made <https://datatracker.ietf.org/doc/draft-ietf-tls-esni/>
    - Encrypted SNI extension in development – difficult to deploy <https://blog.cloudflare.com/encrypted-sni/>
  - Operates within a TCP/IP connection:
    - IP addresses and TCP port numbers are not protected – exposes information about who is communicating and what application is being used
  - Relies on a PKI to validate public keys:
    - Significant concerns about trustworthiness of PKI
    - May deliver 0-RTT data more than once

# Transport Layer Security (TLS) v1.3

- What is TLS?
- TLS handshake protocol
- TLS record protocol
- TLS 0-RTT mode
- Limitations

# Discussion

- End-to-end security
- Robustness principle
- Validating input data
- Writing secure code

# End-to-end Security? (1/2)

- For communication to be secure, it must be **end-to-end**
- The two endpoints are the sender and the final recipient
  - With a data centre or CDN, what is the final recipient?
    - Is it the load balancer at the entrance to the data centre, or the server within the data centre that processes the request?
    - If the request is directed to a content distribution network (CDN), is the end the local cache that serves the request? If so, how are the encryption keys shared?
  - If the data is moving between two users, is it encrypted between the two users or between each user and the data centre?
    - i.e., can the data centre see user-to-user data flows?
    - (This is normal if you run TLS from user-to-data centre then from data centre-to-user)

# End-to-end Security? (2/2)

- Is there in-network processing? How much data is revealed to the in-network server?
  - e.g., video conference with privacy protection vs. without
  - Does the central server decrypt the speech data, mix into one stream and send to the receiver, or does it forward all active streams in encrypted form



- Trades-off security vs bandwidth
  - For audio the bandwidth is small enough this doesn't matter
  - For video conferencing, the combined bandwidth may be significant

# The Robustness Principle (Postel's Law)

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability:

"Be liberal in what you accept, and conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by un-envisioned mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

RFC1122

Balance interoperability with security – don't be *too* liberal in what you accept; a clear specification of how and when you will fail might be more appropriate

# The Robustness Principle (Postel's Law)

**"Be liberal in what you accept, and  
conservative in what you send"**

# The Robustness Principle (Postel's Law)

**"Be liberal in what you accept, and  
conservative in what you send"**

**"Postel lived on a network with all his friends.  
We live on a network with all our enemies.  
Postel was wrong for todays internet."**

*Poul-Henning Kamp*

# The Robustness Principle (Postel's Law)

Stream: Internet Architecture Board (IAB)  
RFC: [9413](#)  
Category: Informational  
Published: June 2023  
ISSN: 2070-1721  
Authors: M. Thomson D. Schinazi

## RFC 9413 Maintaining Robust Protocols

### Abstract

The main goal of the networking standards process is to enable the long-term interoperability of protocols. This document describes active protocol maintenance, a means to accomplish that goal. By evolving specifications and implementations, it is possible to reduce ambiguity over time and create a healthy ecosystem.

The robustness principle, often phrased as "be conservative in what you send, and liberal in what you accept", has long guided the design and implementation of Internet protocols. However, it has been interpreted in a variety of ways. While some interpretations help ensure the health of the Internet, others can negatively affect interoperability over time. When a protocol is actively maintained, protocol designers and implementers can avoid these pitfalls.

### Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Architecture Board (IAB) and represents information that the IAB has deemed valuable to provide for permanent record. It represents the consensus of the Internet Architecture Board (IAB). Documents approved for publication by the IAB are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc9413>.

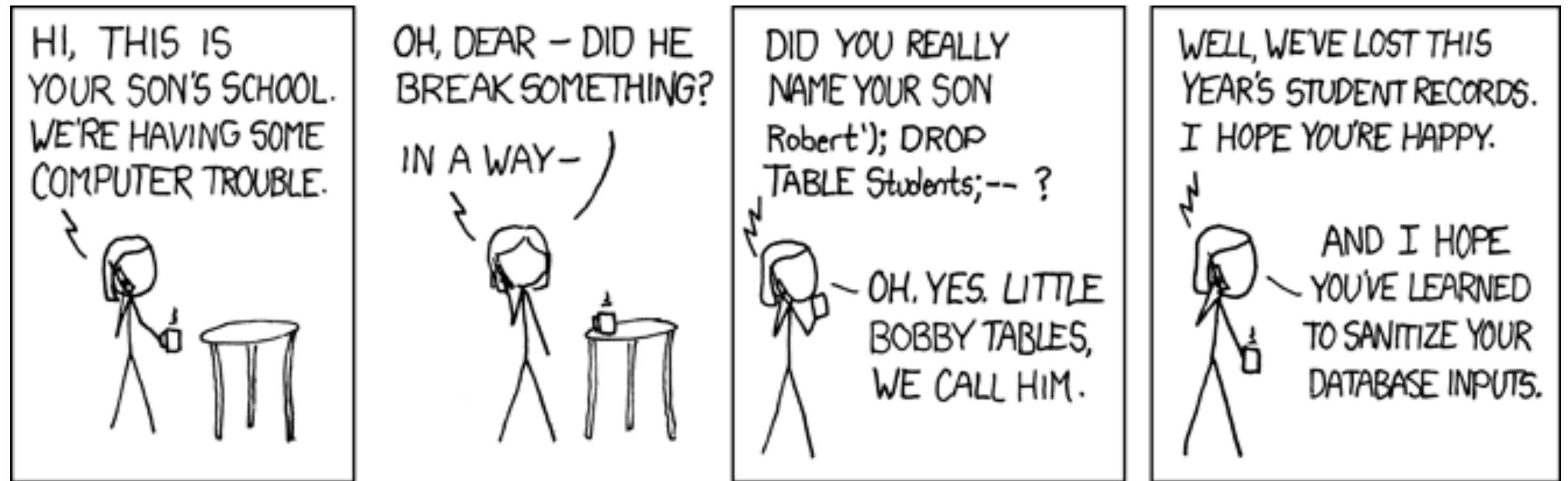
### Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

Thomson & Schinazi      Informational      Page 1

- Systems on the Internet are frequently subject to attack
- Systems must interoperate with other implementations that may be badly written
- Protocol implementations **must have**:
  - Robustness to software defects
  - Robustness to attacks
  - Robustness to the unexpected
- <https://datatracker.ietf.org/doc/rfc9413/>

# Validating Input Data



- Networked applications work with data supplied by un-trusted third parties
- Data read from the network may not conform to the protocol specification
- Due to ignorance, bugs, malice, or a desire to disrupt services
- **Must carefully validate all data before use**

# Writing Secure Code

- The network is hostile: any networked application is security critical
  - Must carefully specify behaviour with both correct and incorrect inputs
  - Must carefully validate inputs and handle errors
  - Must take additional care if using type- and memory-unsafe languages, such as C and C++, since these have additional failure modes
- **The best encryption doesn't help if the endpoints can be compromised**

# Secure Communications

- The need for secure communication
- Principles of secure communication
- TLS v1.3
- Discussion

# Improving Secure Connection Establishment

Networked Systems (H)  
Lecture 4



# Lecture Outline

- Limitations of TLS v1.3
  - Slow Connection Establishment
  - Metadata Leakage
  - Protocol Ossification
- QUIC Transport Protocol
  - Performance, security, and avoiding ossification
  - Unified protocol handshake
  - Reliable multi-streaming transport

# Limitations of TLS v1.3

- Slow Connection Establishment
- Metadata Leakage
- Protocol Ossification

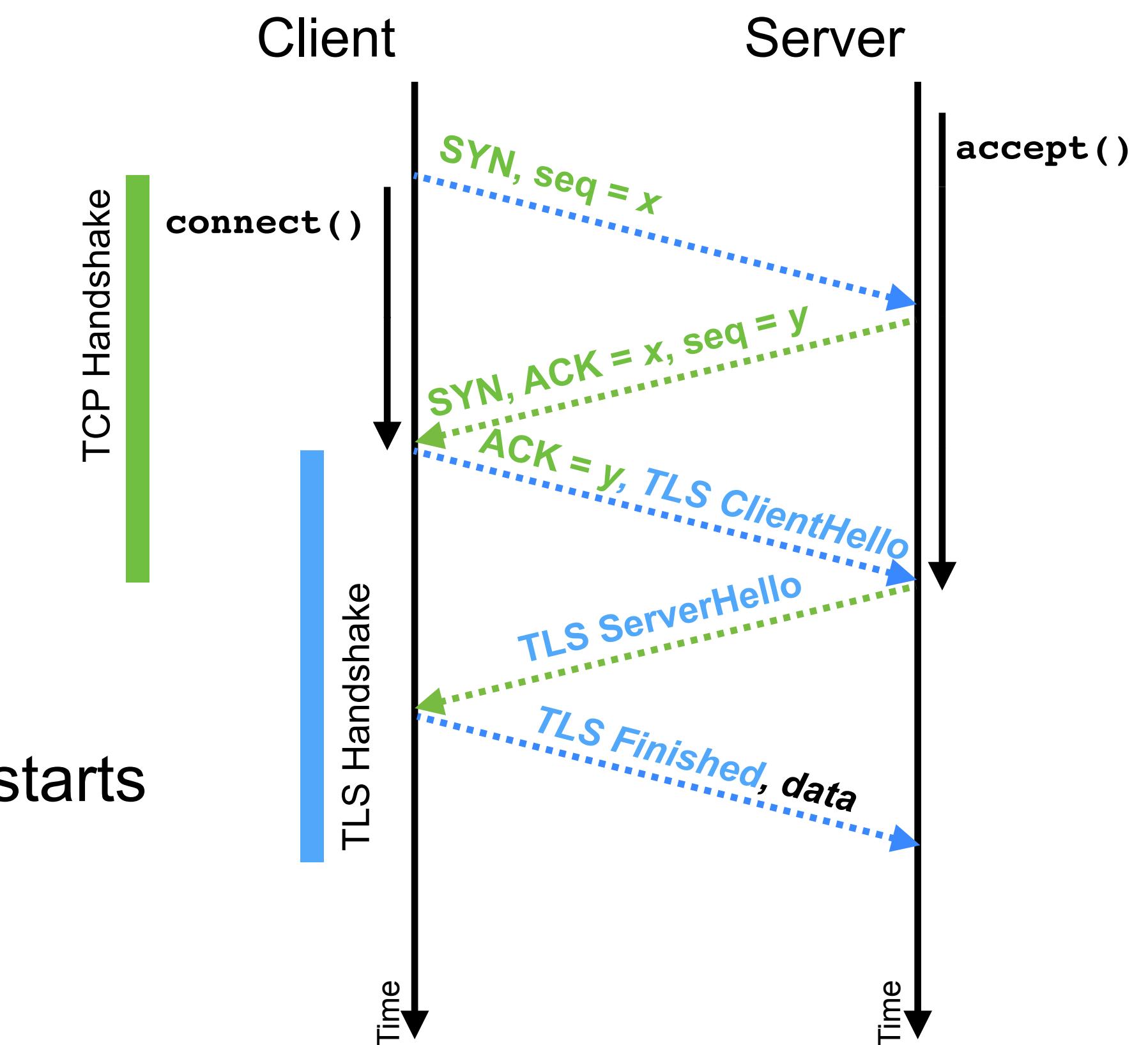
# Limitations of TLS v1.3

- TLS v1.3 is a tremendous success
  - **Significant security improvements** compared to TLS v1.2
    - Removed support for older and less secure encryption and key exchange algorithms
    - Removed support for secure algorithms that have proven difficult to implement correctly
  - Some **performance improvements** to the initial handshake and with 0-RTT mode
- Despite this, TLS v1.3 has some limitations that are hard to fix
  - Connection establishment is still relatively slow
  - Connection establishment leaks potentially sensitive metadata
  - The protocol is ossified due to middlebox interference



# TLS v1.3 Connection Establishment Performance (1/2)

- TCP connection established as usual:
  - **SYN** → **SYN+ACK** → **ACK**
- TLS handshake protocol runs inside TCP connection:
  - TLS **ClientHello** sent with final **ACK**
  - TLS **ServerHello** sent in response
  - TLS **Finished** message concludes, and carries initial secure data record
- First data sent 2x RTT after connection establishment starts
- Earliest response received 3x RTT after connection establishment starts



# TLS v1.3 Connection Establishment Performance (2/2)

- Average web page comprises 1.7 MB of data, fetched as 69 HTTP requests, using 15 TCP connections
- 83% of HTTP requests run over TLS  
<https://httparchive.org/reports/page-weight>
- **Enormous amount of time wasted, waiting for TCP and TLS connection establishment handshakes**

Destination	Average RTT (ms)
London	72.5
New York	153.3
Los Angeles	221.2
Sydney	381.2

RTT measurements (ping times) from residential site in Glasgow

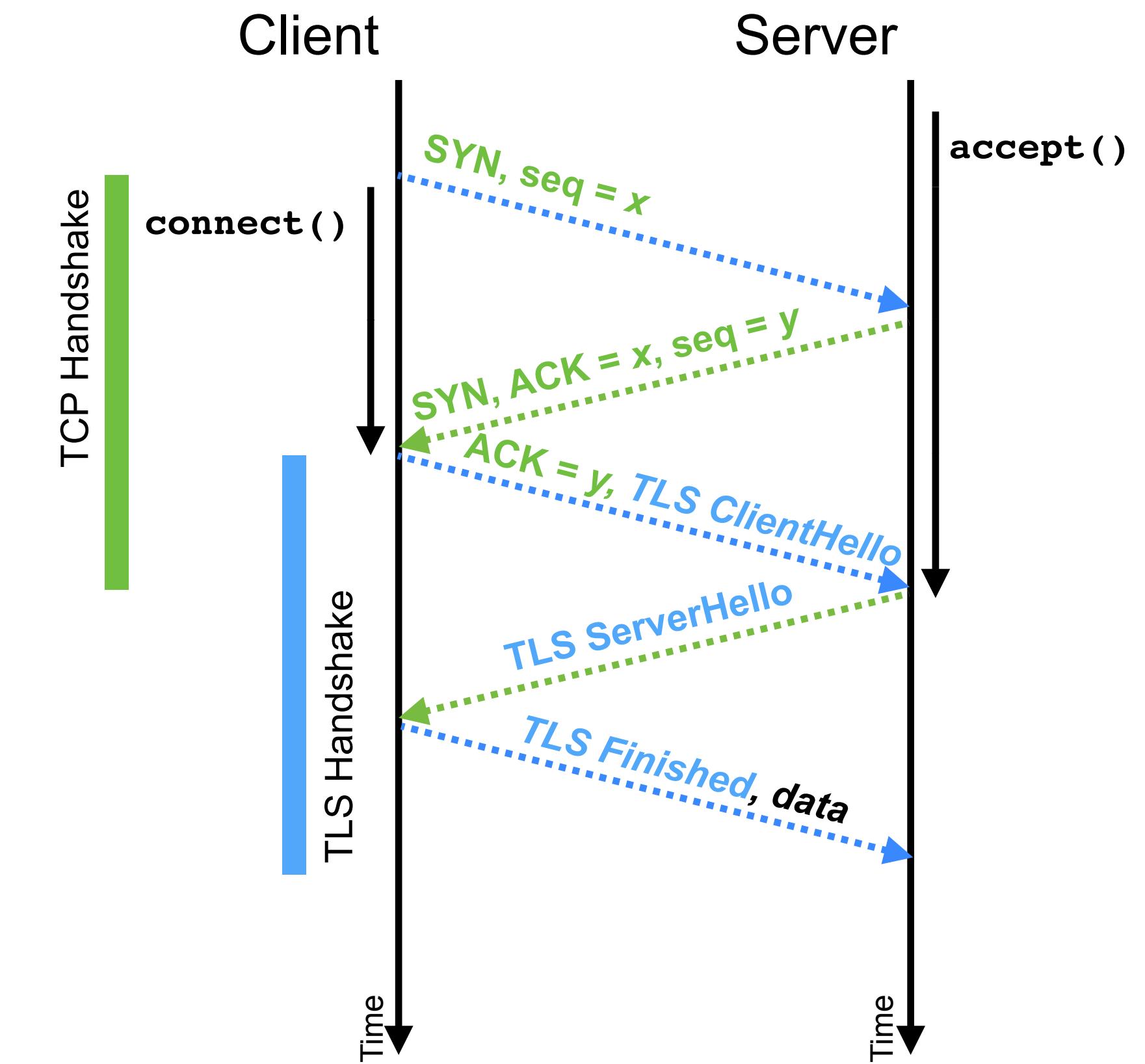
- Can we speed up TLS connection setup?
  - 0-RTT Connection Reestablishment – speed-up connections to known servers
  - Concurrent TCP and TLS handshake – speed-up connections to all servers

# 0-RTT Connection Reestablishment (1/4)

- Common to connect to a previously known TLS server – is it possible to shortcut the connection establishment in such cases?
- Need to understand:
  - What is the role of the TLS handshake?
  - How to encrypt initial data?
  - What are the potential risks?

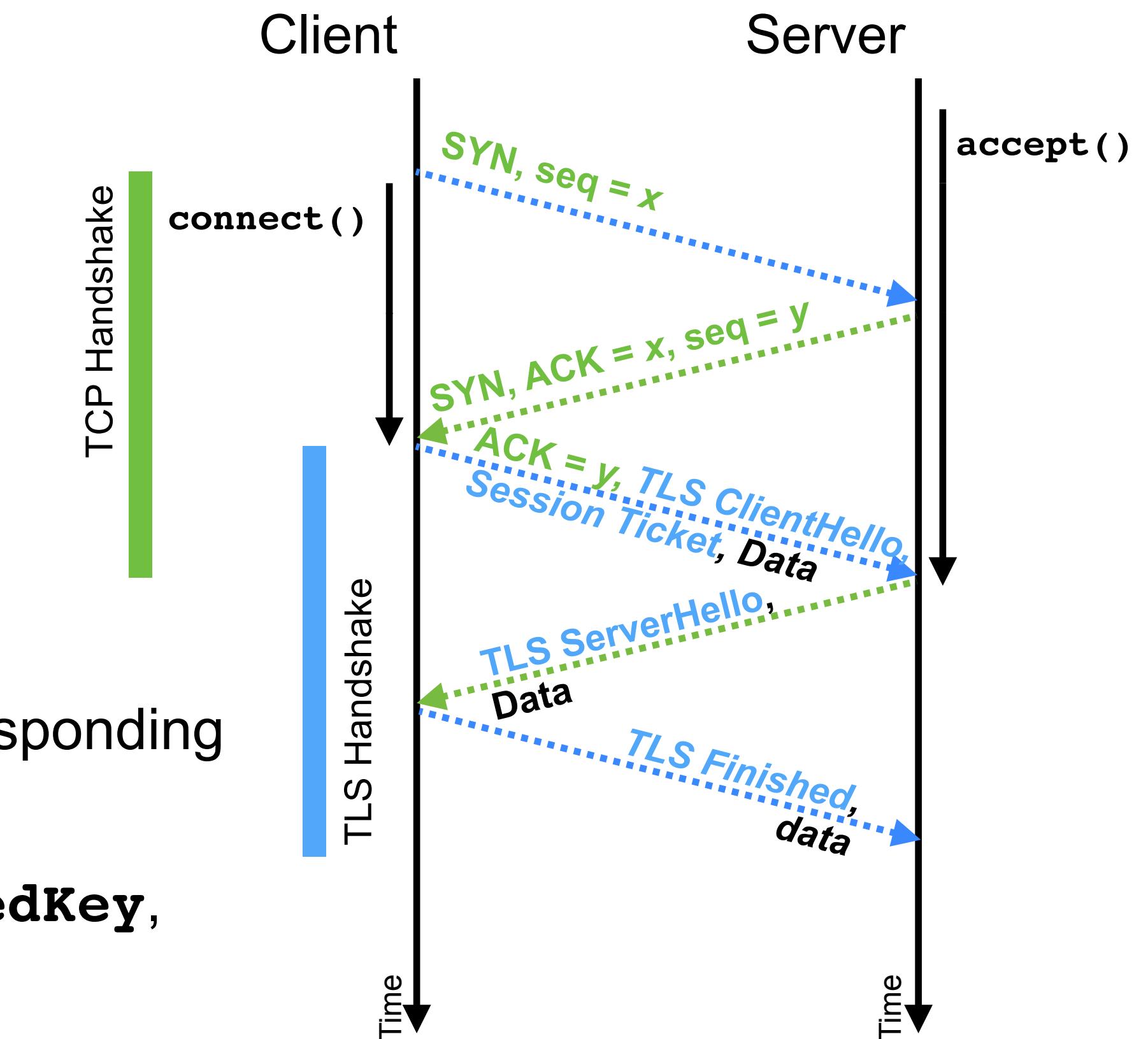
# 0-RTT Connection Reestablishment (2/4)

- What is the role of the TLS handshake?
  - Uses public key cryptographic techniques to establish an **ephemeral session key**, used to encrypt the data
  - The **ClientHello** and **ServerHello** are used to exchange material used to derive a session key – *using ECDHE key negotiation*
  - The session is ephemeral – different for each connection; derived from the public keys and a random value
  - The ephemeral session key provides **forward secrecy** – each connection has a unique key; if the encryption key for one session leaks, it doesn't help an attacker break other sessions
  - Retrieve the server's certificate, allowing the client to authenticate the server
  - The **ServerHello** contains the certificate



# 0-RTT Connection Reestablishment (3/4)

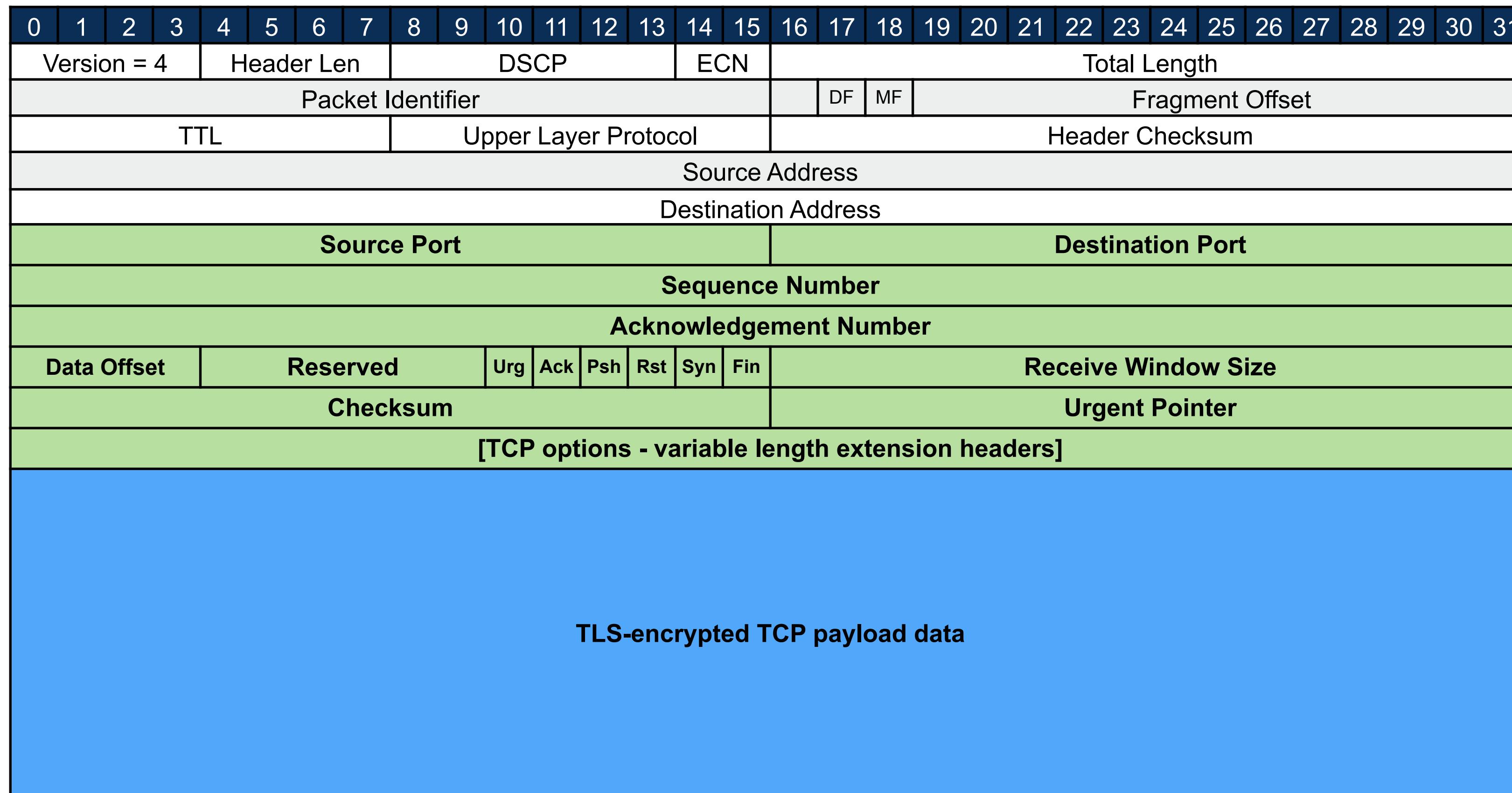
- How to encrypt initial data?
  - Cannot negotiate ephemeral session key for initial data  
→ relies on data exchanged in the handshake
  - Reuse a **pre-shared key** agreed in previous TLS session
  - In a previous TLS connection:
    - Server sends a **PreSharedKey** with a **SessionTicket** to identify the key
  - When reestablishing a connection:
    - Client sends **SessionTicket**, data encrypted using corresponding **PreSharedKey**, along with **ClientHello**
    - The server uses **SessionTicket** to find saved **PreSharedKey**, decrypt the data
    - **ClientHello** and **ServerHello** complete usual key exchange; data sent with **ServerHello** and later protected using ephemeral session key → no additional round-trips due to TLS



# 0-RTT Connection Reestablishment (4/4)

- What are the potential risks?
  - 0-RTT data sent with **ClientHello** using a **PreSharedKey** is not forward secret
    - Use of **PreSharedKey** links TLS connections – if session where **PreSharedKey** is distributed is compromised, 0-RTT data sent using that key in future connections will also be compromised
  - 0-RTT data sent with **ClientHello** using a **PreSharedKey** is subject to replay attack
    - The 0-RTT data is accepted during TLS connection establishment
    - If on-path attacker captures and replays the TCP segment with the **ClientHello**, **SessionTicket**, and data protected with the **PreSharedKey**, that data will be accepted by the server again
      - The server will respond to the replay, trying to complete the handshake – this might fail
      - But – by then, the data will have been accepted
    - Ensure 0-RTT data is **idempotent** to avoid this risk
- Be very careful using 0-RTT data in TLS v1.3 – trades performance for safety

# TLS v1.3 Metadata Leakage (1/2)



IP exposes addresses

TCP exposes port numbers  
and connection metadata

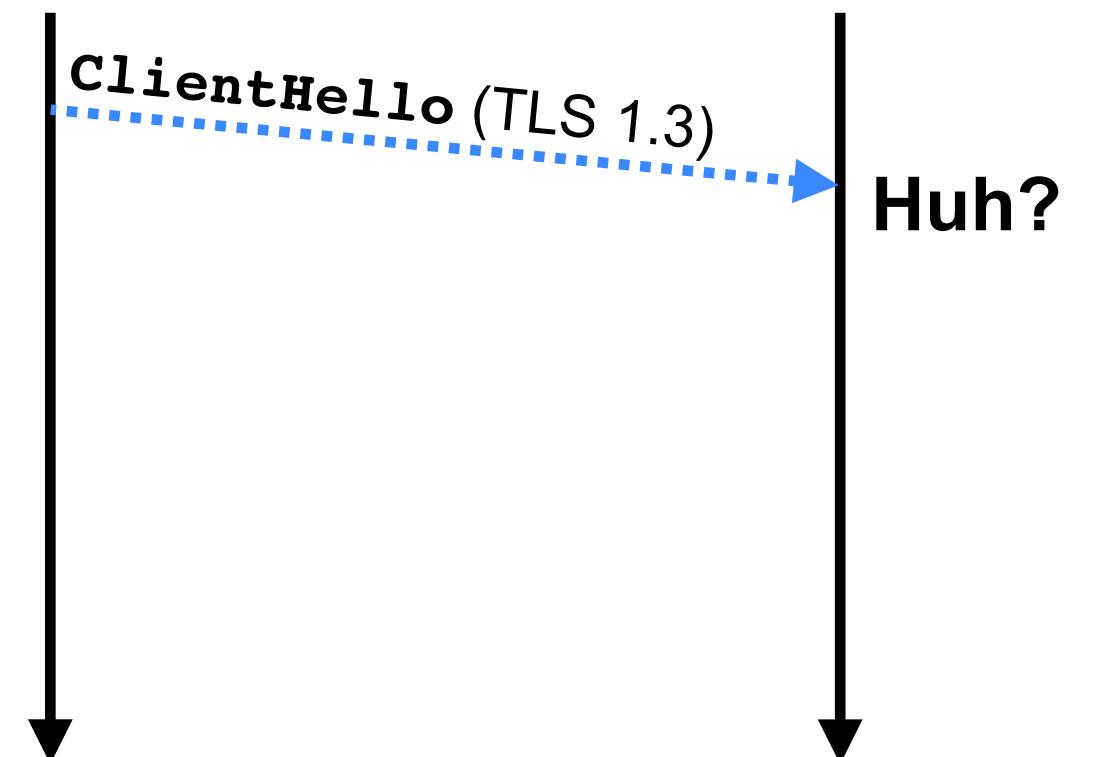
**Is there a privacy  
concern?**

# TLS v1.3 Metadata Leakage (2/2)

- When TLS is used with HTTPS, **ClientHello** includes the Server Name Indication (SNI) extension
  - Identifies requested site, so server knows what public key to use in **ServerHello**
    - Required to support shared hosting, with multiple websites on one server
    - Has to be unencrypted – sent before session keys are negotiated
    - Can't encrypt with **PreSharedKey**, since that's provided by server, and goal is to select the server
  - **A privacy concern with TLS v1.3**
    - See <https://datatracker.ietf.org/doc/draft-ietf-tls-esni/> for work-in-progress attempt to resolve

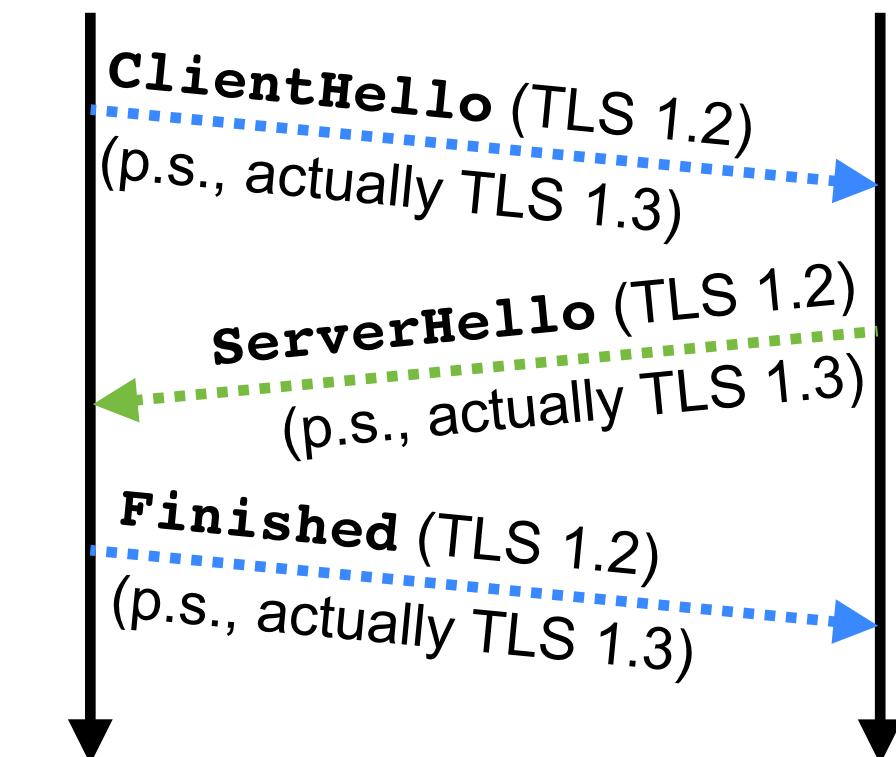
# TLS v1.3 Protocol Ossification (1/3)

- TLS is widely implemented, but many poor quality implementations:
  - Some TLS servers fail if **clientHello** uses unexpected version number, rather than try to negotiate older version
  - Some firewalls block connections if **ClientHello** is structured differently to that used by TLS 1.2 and earlier, even if TLS 1.3 is signalled
- Original design of TLS 1.3 changed **clientHello**
  - Updated the version number (1.2 → 1.3)
  - Removed some now unused header fields
- Measurements showed this caused ~8% of TLS 1.3 connections to fail



# TLS v1.3 Protocol Ossification (2/3)

- Later versions of TLS 1.3 changed the design to work around these bugs
  - Version number in **ClientHello** says **TLS 1.2**; unused header fields present with dummy values; extension header to **ClientHello** signals actual version
  - (Similar changes in **ServerHello**)
  - When TLS 1.3 client talks to TLS 1.3 server, version negotiated in extensions
  - When TLS 1.3 client talks to TLS 1.2 server, extension ignored and TLS 1.2 is negotiated
- **Protocol ossification is a significant concern**
  - TLS is not the only protocol to include such workarounds
  - Widely deployed faulty implementations constrain design of most protocols



# How to Avoid Protocol Ossification?

- Ossification happens when extension mechanisms, or allowed flexibility, are not used
  - TLS 1.3 was released ten years after TLS 1.2
  - Allowed products to be built and deployed that didn't do version negotiation correctly, since no new versions to negotiate
  - Allowed products to be built that relied on the presence and order of fields in `ClientHello`, since all implementations included the same fields in the same order
- **Generate Random Extensions And Sustain Extensibility (GREASE)**
  - If the protocol allows extensions, send extensions
  - If the protocol allows different versions, negotiate different versions
  - **Do this even if you don't need to → “use it or lose it”**
    - Send meaningless dummy extensions that are ignored
    - Change the version number to prove you can

# Limitations of TLS v1.3

- TLS v1.3 is a significant improvement on prior versions: faster and more secure
- TLS v1.3 runs within a TCP connection:
  - Must wait for TCP connection establishment
  - Some metadata leakage
- Implementations of TLS are ossified and hard to extend

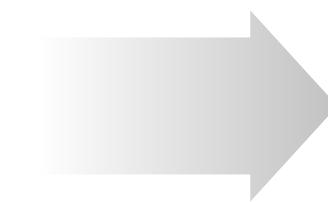
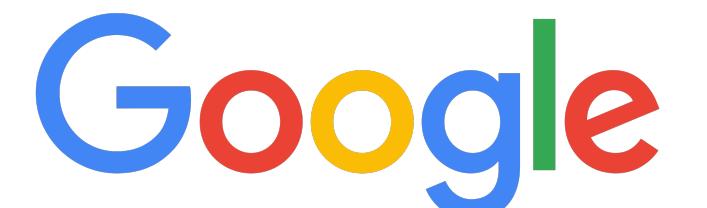
# QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

# QUIC: Performance, Security, Avoiding Ossification

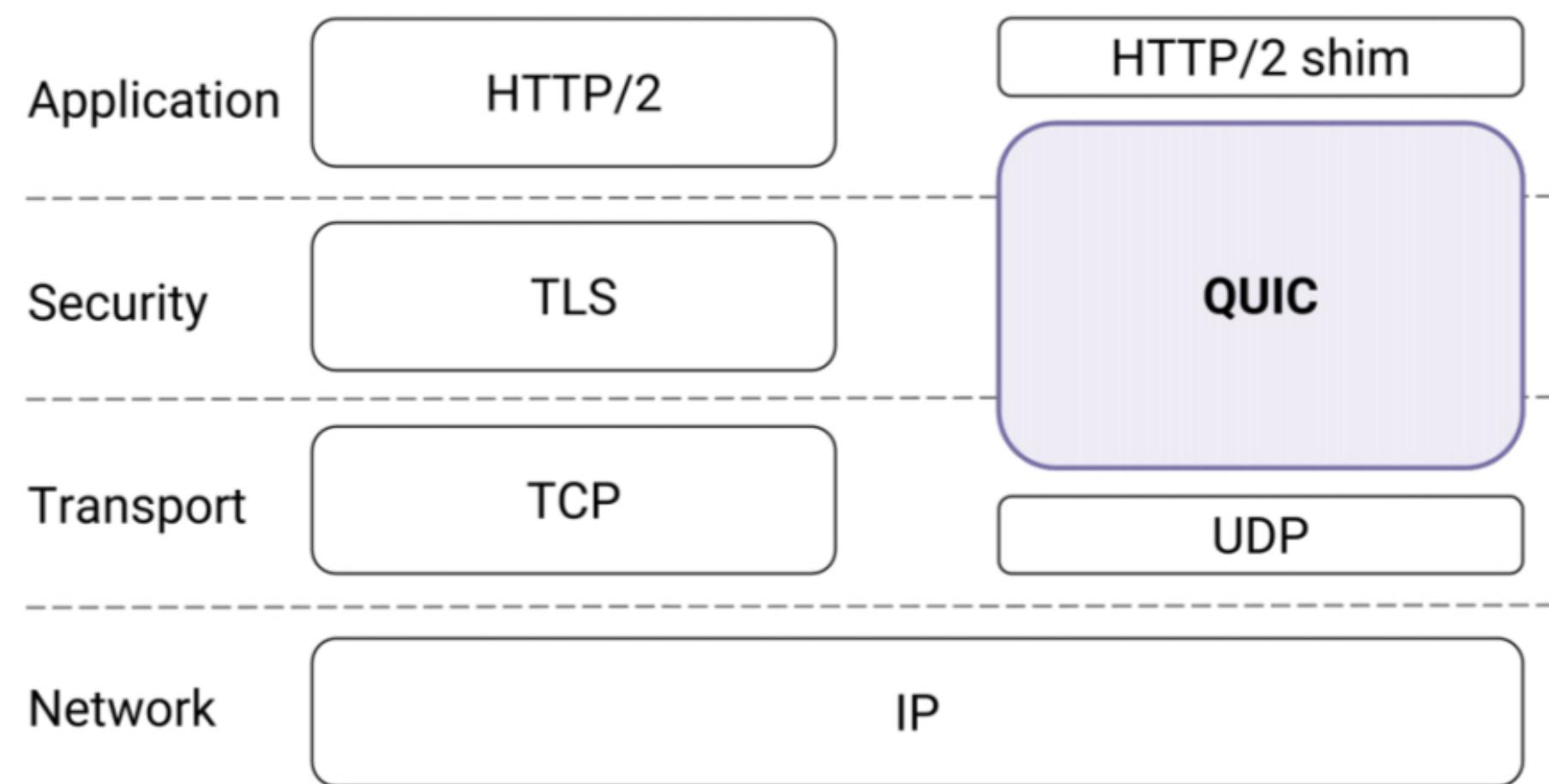
- What's wrong with TLS v1.3 over TCP?
  - Slow to connect – due to sequential TCP and TLS handshakes
  - Leaks some metadata
  - Ossified and hard to extend
- QUIC aims to replace TLS v1.3 and TCP with a single secure transport protocol
  - Reduce latency by overlapping TLS and transport handshake
  - Avoid metadata leakage via pervasive encryption
  - Avoid ossification via systematic application of GREASE and encryption

# History of QUIC



- 2012-2015: Initial proposal and experimental deployment by Google
- 2016-2021: IETF QUIC working group
  - Specifications went through 34 drafts in IETF before RFCs were published – standardisation led to **significant differences and improvements compared to initial Google proposal**
    - RFC 8999 “Version-Independent Properties of QUIC” <https://datatracker.ietf.org/doc/rfc8999/>
    - RFC 9000 “QUIC: A UDP-Based Multiplexed and Secure Transport” <https://datatracker.ietf.org/doc/rfc9000/>
    - RFC 9001 “Using TLS to Secure QUIC” <https://datatracker.ietf.org/doc/rfc9001/>
    - RFC 9002 “QUIC Loss Detection and Congestion Control” <https://datatracker.ietf.org/doc/rfc9002/>
  - GitHub repo for the specification has ~2200 issues and ~2500 pull requests

# QUIC Overview (1/2)



**Figure 1: QUIC in the traditional HTTPS stack.**

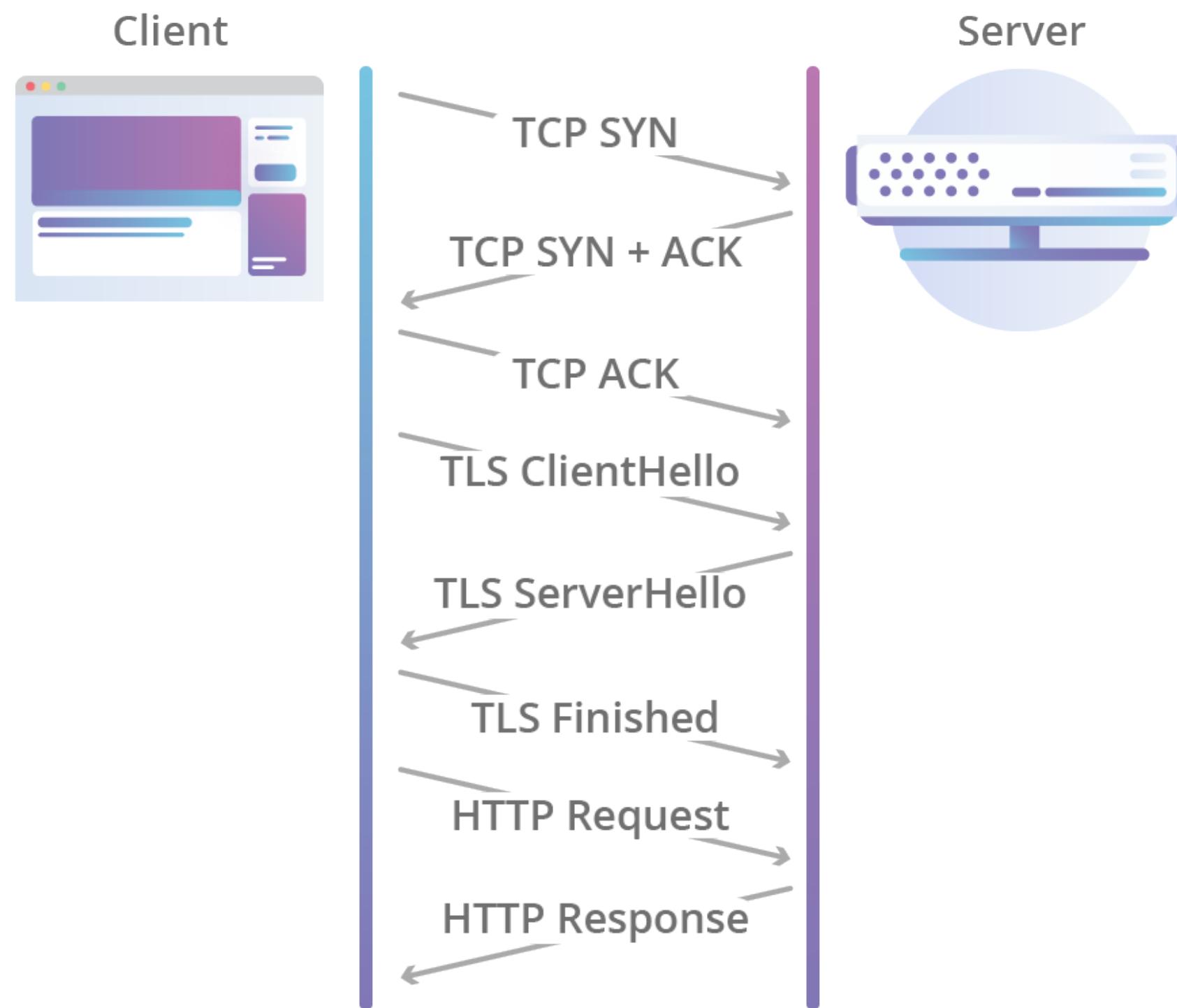
- QUIC replaces TCP, TLS, and parts of HTTP
  - HTTP → stream multiplexing
  - TLS → security
  - TCP → reliability, ordering, congestion control
  - Runs on UDP for ease of deployment
- QUIC is a general purpose client-server transport
  - Designed to run HTTP effectively



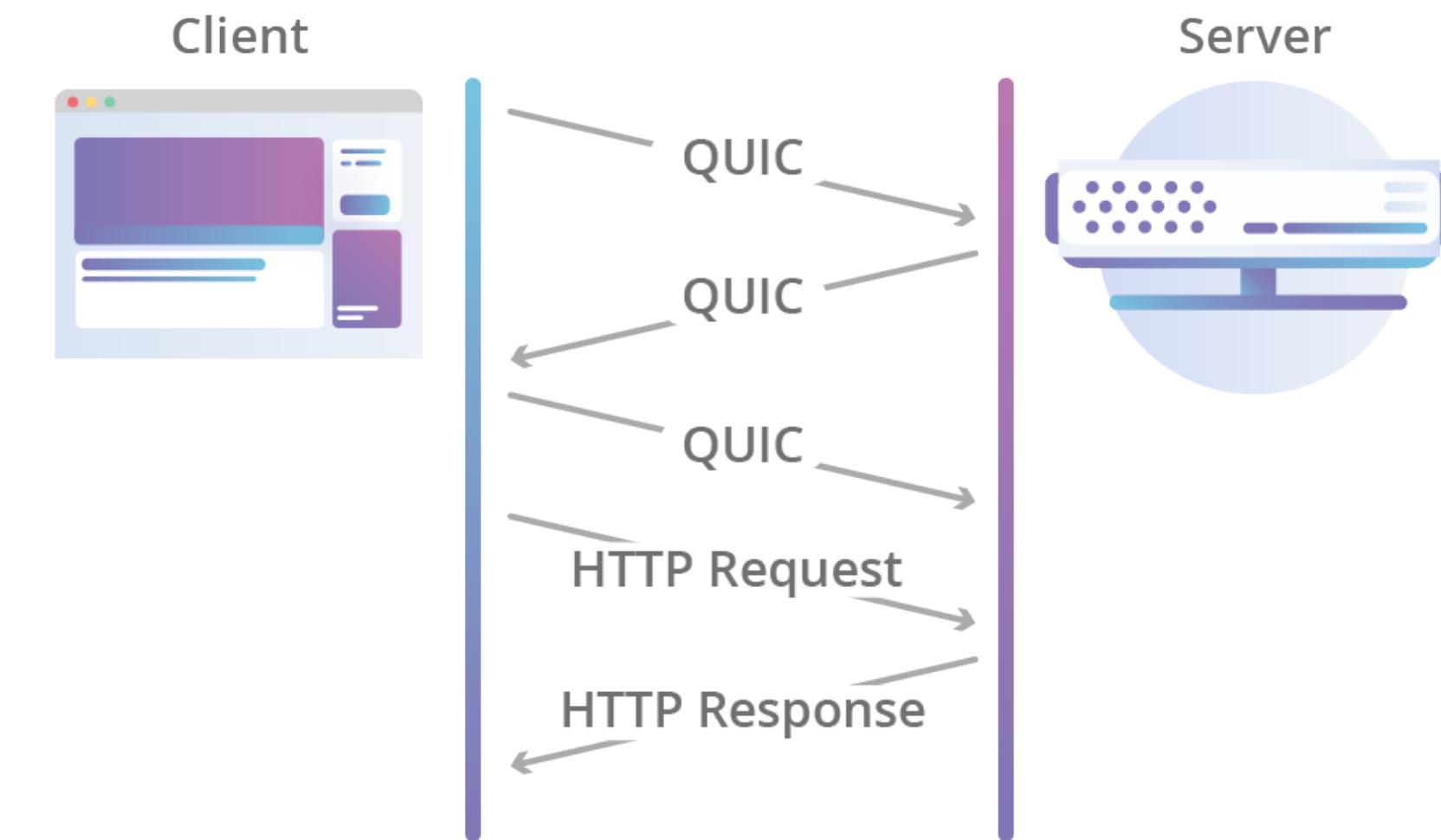
Figure from: A. Langley *et al.*, “The QUIC transport protocol: Design and Internet-scale deployment”, Proceedings of the ACM SIGCOMM Conference, Los Angeles, August 2017. <https://dl.acm.org/authorize.cfm?key=N33907>

# QUIC Overview (2/2)

HTTP Request Over TCP + TLS

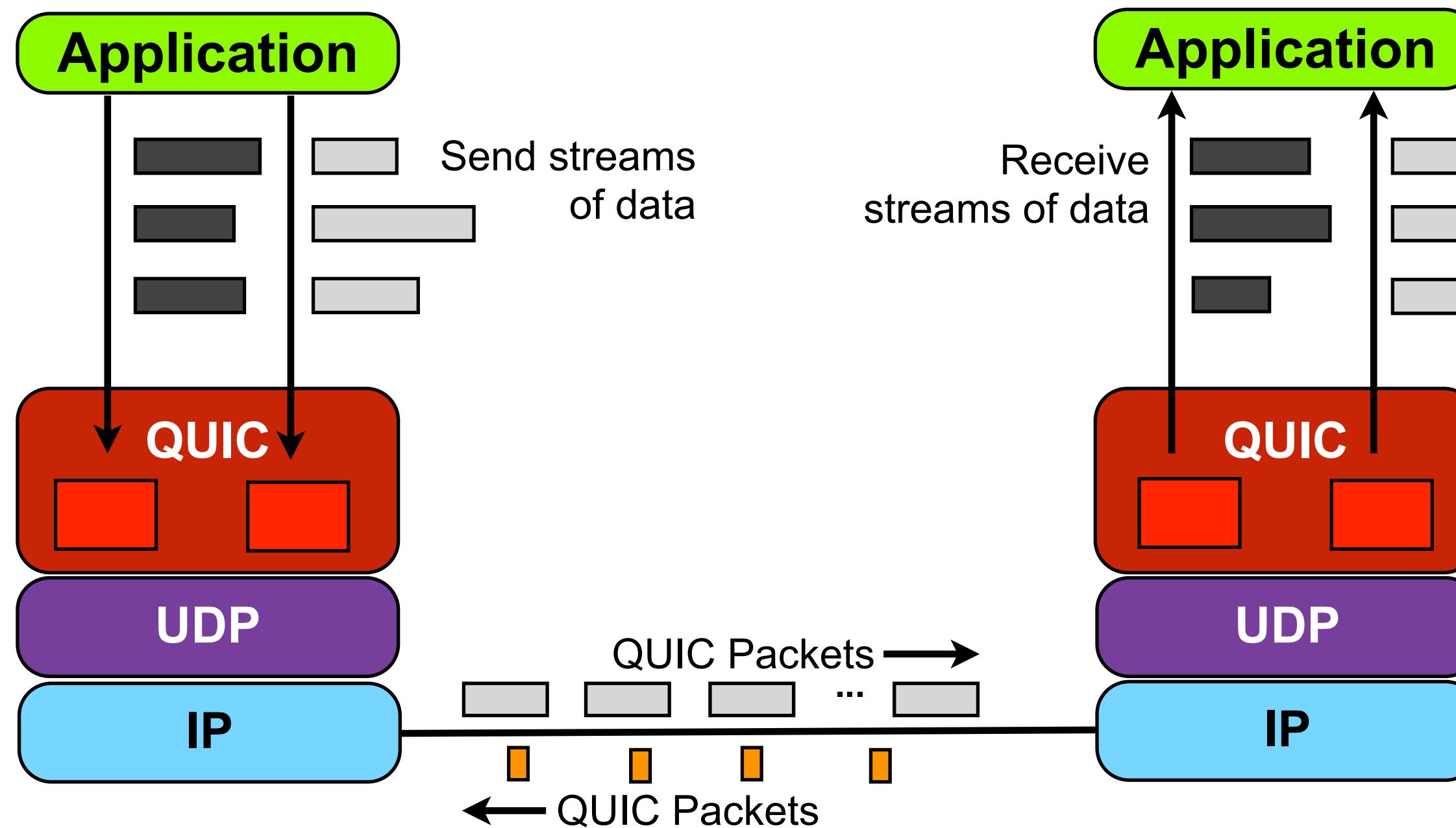


HTTP Request Over QUIC



Figures from <https://blog.cloudflare.com/the-road-to-quic/>

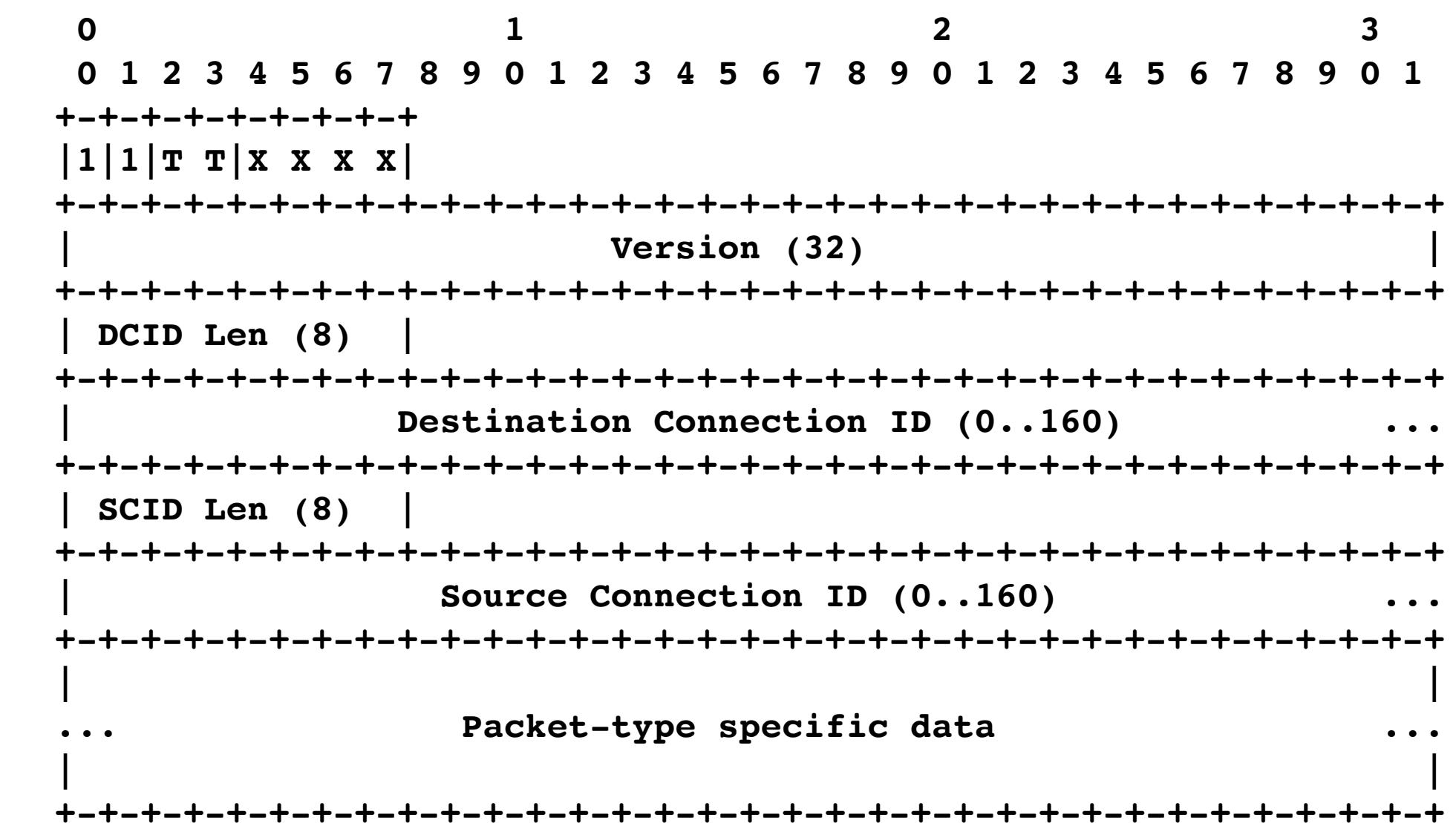
# QUIC Packets, Headers, and Frames



- QUIC sends and receives streams of data within a connection
  - Up to  $2^{62}$  different streams in each direction in a single QUIC connection
- A connection comprises QUIC packets sent within UDP datagrams
- Each QUIC packet starts with a header, and contains one or more frames
  - Frames contain data for streams, or acknowledgements, or other control messages

# QUIC Headers (1/2)

- QUIC packets can be long header packets or short header packets
- **Long header packets** are used to establish QUIC connections
  - They all start with a common header, followed by packet-type specific data
  - Four different long-header packet types, denoted by the TT field in the header:
    - **Initial** – initiates connection, starts TLS handshake
    - **0-RTT** – idempotent data sent with initial handshake, when resuming a session
    - **Handshake** – completes connection establishment
    - **Retry** – used to force address validation
  - Several **Initial**, **0-RTT**, and **handshake** packets can be included in one UDP datagram, one after the other, followed by a short header packet



# QUIC Headers (2/2)

- QUIC packets can be long header packets or short header packets
  - There is one **short header packet** defined in QUIC:
    - **1-RTT** – Used for all packets sent after the TLS handshake is complete
    - The short header is followed by QUIC frames in the enclosing UDP packet
    - Compared to long header, omits information that can be inferred from context

```
0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+---+---+---+
|0|1|S|R|R|K|P P|
+-+-+-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Destination Connection ID (0..160)      ...
+-+-+-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Packet Number (8/16/24/32)        ...
+-+-+-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Protected Payload (*)       ...
+-+-+-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

# QUIC Frames

- QUIC packets contain an encrypted sequence of **frames**
  - CRYPTO frames are used to carry TLS messages such as the **ClientHello**, **ServerHello** etc.
  - STREAM and ACK frames send data and acknowledgements
  - Migration between two network interfaces is supported by **PATH\_CHALLENGE** and **PATH\_RESPONSE** frames
  - Other frames control progress of a QUIC connection
- Compared to TCP, QUIC headers are limited in scope
  - Functionality provided by frames instead → more extensible
  - e.g., TCP sends sequence numbers and acknowledgements in the header; QUIC sends this information in STREAM and ACK frames

Type	Value	Frame Type	Name
	0x00		PADDING
	0x01		PING
	0x02 - 0x03		ACK
	0x04		RESET_STREAM
	0x05		STOP_SENDING
	0x06		CRYPTO
	0x07		NEW_TOKEN
	0x08 - 0x0f		STREAM
	0x10		MAX_DATA
	0x11		MAX_STREAM_DATA
	0x12 - 0x13		MAX_STREAMS
	0x14		DATA_BLOCKED
	0x15		STREAM_DATA_BLOCKED
	0x16 - 0x17		STREAMS_BLOCKED
	0x18		NEW_CONNECTION_ID
	0x19		RETIRE_CONNECTION_ID
	0x1a		PATH_CHALLENGE
	0x1b		PATH_RESPONSE
	0x1c - 0x1d		CONNECTION_CLOSE
	0x1e		HANDSHAKE_DONE

# QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

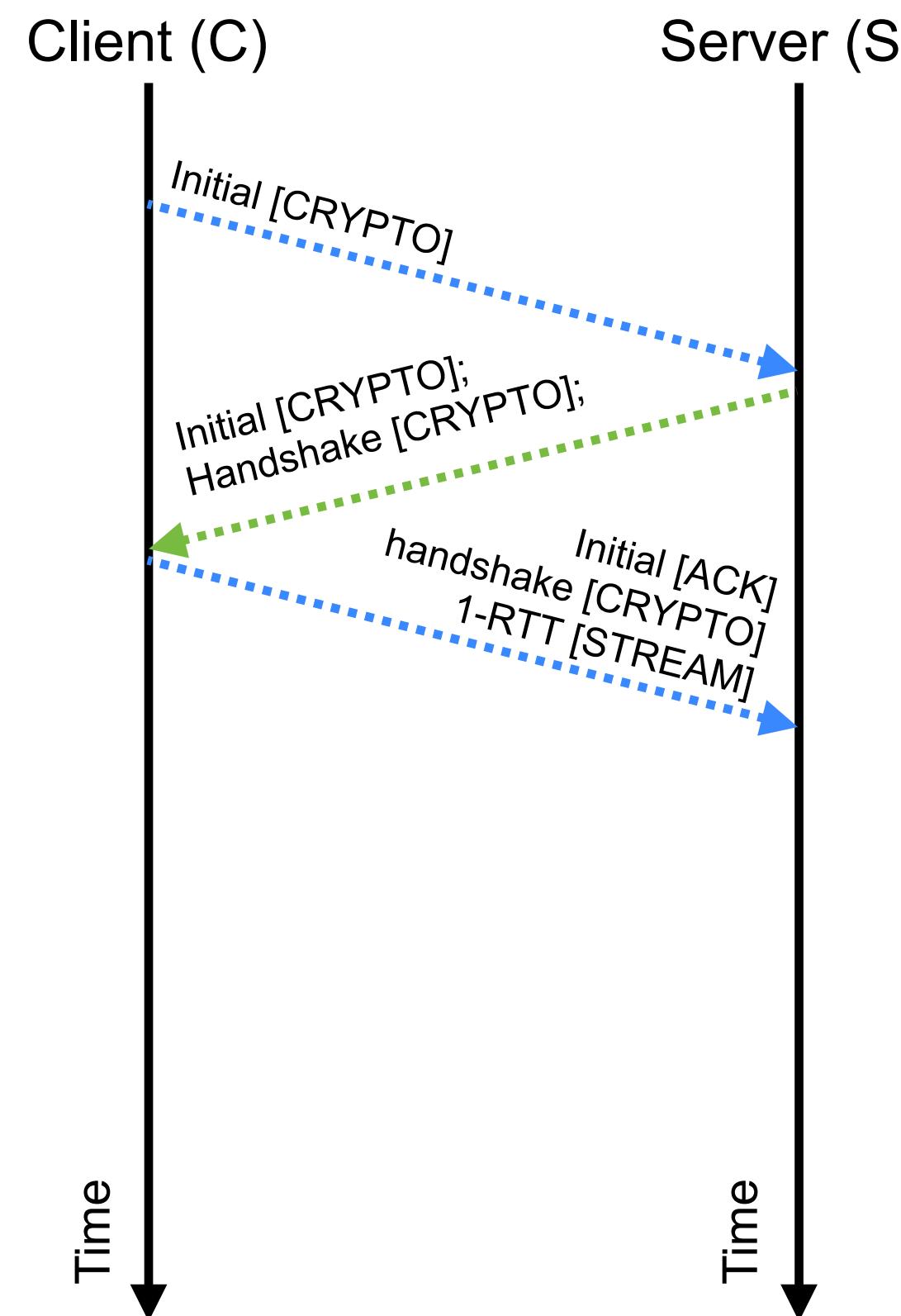
# QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

# QUIC Connection Establishment and Data Transfer

- A QUIC connection proceeds in two phases: handshake and data transfer
  - The **handshake** uses long header packets – establishes the connection, negotiates encryption keys, authenticates the server
  - The **data transfer** phase uses short header packets – sends data and acknowledgements after the connection is established

# QUIC Connection Establishment (1/5)



- QUIC combines connection establishment and TLS handshake into one round-trip
  - C → S: QUIC **Initial** packet
    - Initial packet contains a CRYPTO frame that contains TLS **clientHello**
  - S → C: QUIC **Initial** and **Handshake** packets
    - Both QUIC packets can be sent in a single UDP datagram
    - Initial packet contains CRYPTO frame that contains TLS **ServerHello**
    - Handshake packet contains other connection setup information
  - C → S: QUIC **Initial**, **Handshake**, and **1-RTT** packets
    - All three QUIC packets can be sent in a single UDP datagram
    - QUIC **Initial** packet contains ACK frame, acknowledging the server's Initial packet
    - QUIC **Handshake** packet contains CRYPTO frame that contains TLS **Finished**
    - QUIC **1-RTT** (short header) packet contains STREAM frame with initial data sent from client to server

# QUIC Connection Establishment (2/5)

```
+---+---+---+---+  
|1|1| 0 |R R|P P|  
+---+---+---+---+  
|                         Version (32)          |  
+---+---+---+---+  
| DCID Len (8) |  
+---+---+---+---+  
|             Destination Connection ID (0..160) ...  
+---+---+---+---+  
| SCID Len (8) |  
+---+---+---+---+  
|             Source Connection ID (0..160) ...  
+---+---+---+---+  
|     Token Length (i) ...  
+---+---+---+---+  
|     Token (*) ...  
+---+---+---+---+  
|     Length (i) ...  
+---+---+---+---+  
|     Packet Number (8..32) ...  
+---+---+---+---+  
|     Payload (8..) ...  
+---+---+---+---+
```

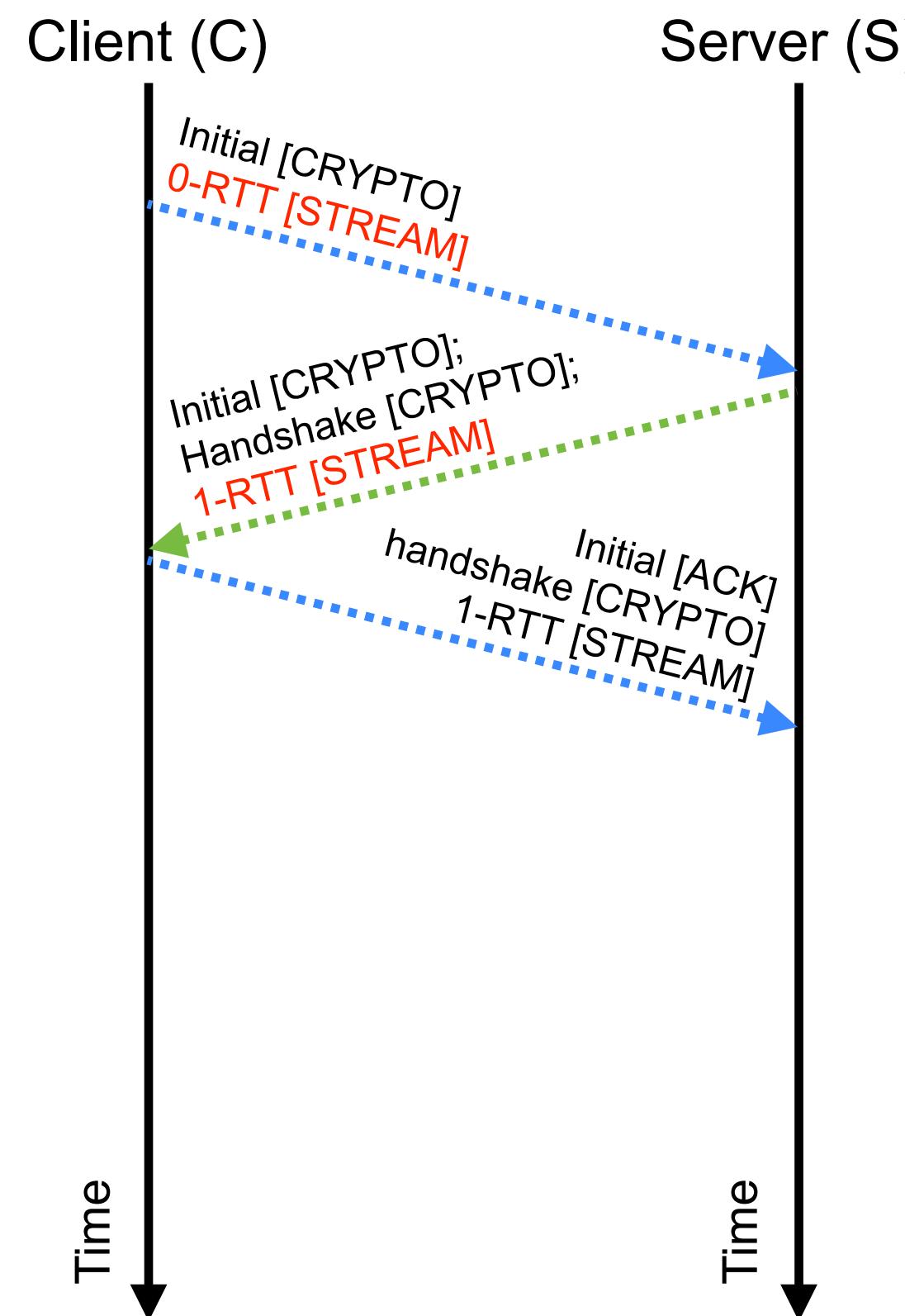
- QUIC **Initial** packets play two main roles:
  - Synchronise client and server state – like TCP's **SYN** and **SYN+ACK** packets
  - Contains CRYPTO frame, with TLS **clientHello** or **Finished**, and may also contain ACK frame
- QUIC **Initial** packets also carry optional **Token**
  - Server can refuse the initial connection attempt, and send a **Retry** packet containing a **Token**.
  - Client must then retry the connection, providing the **Token** in its **Initial** packet
  - Can be used to prevent connection spoofing

# QUIC Connection Establishment (3/5)

```
+---+---+---+---+  
|1|1| 2 |R R|P P|  
+---+---+---+---+  
|                         Version (32)          |  
+---+---+---+---+  
| DCID Len (8)   |  
+---+---+---+---+  
|             Destination Connection ID (0..160) ...  
+---+---+---+---+  
| SCID Len (8)   |  
+---+---+---+---+  
|             Source Connection ID (0..160) ...  
+---+---+---+---+  
|             Length (i) ...  
+---+---+---+---+  
| Packet Number (8..32) ...  
+---+---+---+---+  
|             Payload (8...) ...  
+---+---+---+---+
```

- QUIC **Handshake** packets complete the TLS 1.3 exchange
  - The TLS **ServerHello** and **Finished** messages
  - TLS can also renegotiate new keys part-way through a connection – this is done in **Handshake** packets

# QUIC Connection Establishment (4/5)

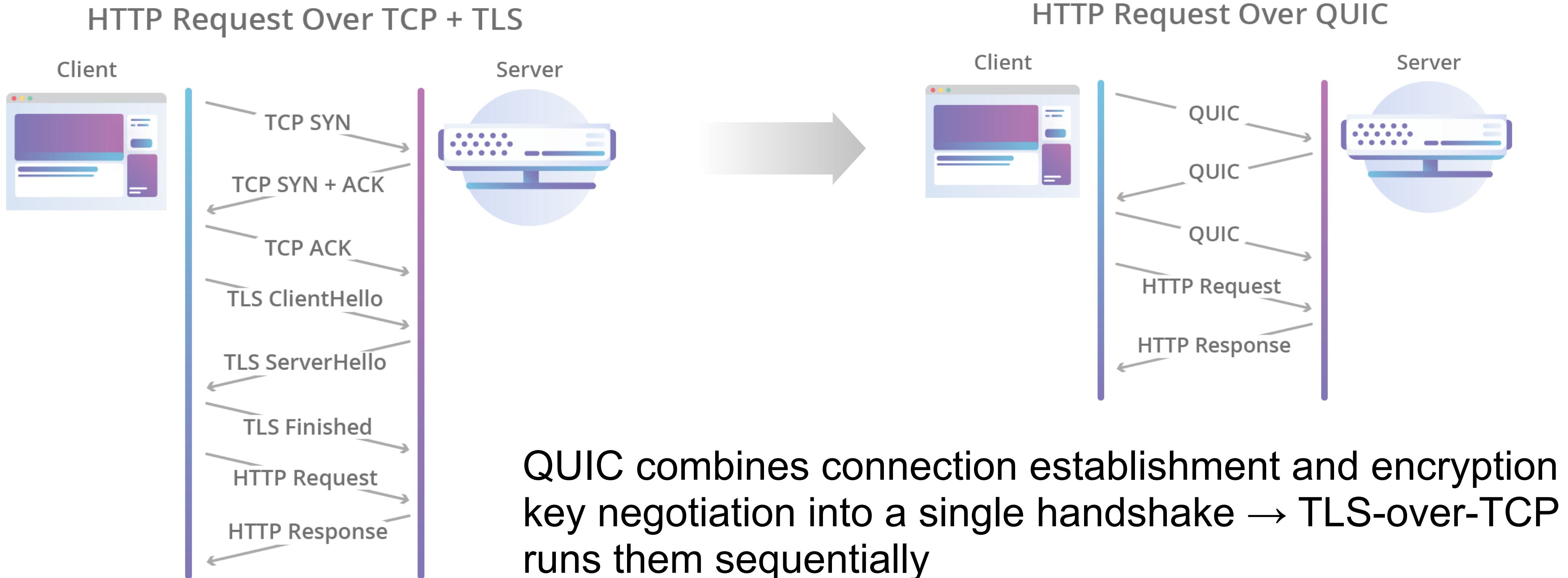


- QUIC supports TLS 0-RTT session re-establishment:
  - QUIC **Initial** packet contains CRYPTO frame with a TLS **ClientHello** and a **SessionTicket**
  - QUIC **0-RTT** packet included in the same UDP datagram contains a STREAM frame carrying idempotent 0-RTT data:

```
+-----+  
| 1 | 1 | R R | P P |  
+-----+  
|                                Version (32)          |  
+-----+  
| DCID Len (8) |  
+-----+  
|             Destination Connection ID (0..160) ...  
+-----+  
| SCID Len (8) |  
+-----+  
|             Source Connection ID (0..160) ...  
+-----+  
| Length (i) | ...  
+-----+  
| Packet Number (8..32) | ...  
+-----+  
| Payload (8...) | ...  
+-----+
```

- Server responds with data in 1-RTT (short header) packet, along with its **Initial** and **handshake** packets

# QUIC Connection Establishment (5/5)



# Data Transfer, Streams, and Reliability (1/3)

- After handshake has finished, QUIC switches to sending short header packets
    - The short header contains a **Packet Number** field
      - Packet numbers **increase by one** for each packet sent; ACK frames indicate received packet numbers
        - QUIC packet numbers count packets sent; TCP sequence numbers count bytes of data sent
      - QUIC **never** retransmits packets – retransmits frames sent in lost packets in new packets, with new packet numbers
        - TCP retransmits lost packet with original sequence number
    - Protected payload section of short header packets contains encrypted QUIC frames
      - STREAM frames contain data
      - ACK frames contain acknowledgements
    - Performs congestion control as in TCP → Lecture 6

The diagram illustrates the IEEE 802.11 frame structure, showing a 160-bit frame divided into four 40-bit segments (0, 1, 2, 3) at the top. The first segment (0) contains the Destination Connection ID (0..160) and the Packet Number (8..32). The second segment (1) contains the Protected Payload (8..).

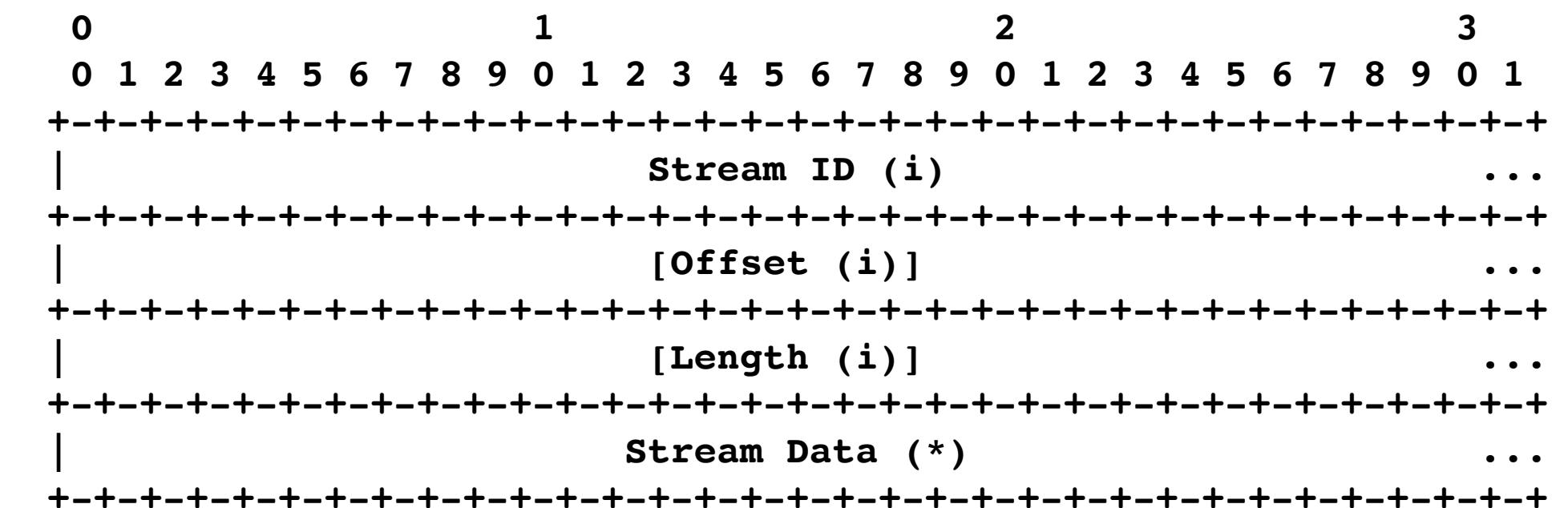
# Data Transfer, Streams, and Reliability (2/3)

- QUIC sends acknowledgements of received packets in ACK frames
  - Sent **inside** a long- or short-header packets; unlike TCP, not part of headers
  - Indicate sequence numbers of QUIC **packets** that were received, not frames

```
0      1      2      3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Largest Acknowledged (i) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               ACK Delay (i) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               ACK Range Count (i) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               First ACK Range (i) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               ACK Ranges (*) ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               [ECN Counts] ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

# Data Transfer, Streams, and Reliability (3/3)

- Data is sent within STREAM frames, sent within QUIC packets
  - Contain a stream identifier, offset of the data within the stream, data length, and data
  - QUIC provides **multiple** reliable byte streams within a single connection
    - Data for each stream is delivered reliably and in-order
    - Order is not preserved between streams
    - Avoids head-of-line blocking between streams → Lecture 5
  - Can view QUIC streams as multiple unframed byte streams sent within a single connection; alternatively can view each stream as framing a message, and the connection as a series of messages



# QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

# QUIC Transport Protocol

- Development and basic features
- Connection establishment; data transfer
- Avoiding ossification; limitations

# QUIC over UDP

- Why run QUIC over UDP rather than over IP?
- To ease end-system deployment in user-space applications
  - User-space applications, running over UDP, are easy to build
    - BSD sockets; portable → same API works everywhere
    - Widely understood programming model
    - No need for privileged access
  - No portable, unprivileged, interface to build applications that run directly over IP
    - Implementations have to run within the operating system kernel
    - Deploying kernel updates is difficult
    - Deploying application updates is straightforward
- To work around protocol ossification due to middleboxes
  - Firewalls block anything other than TCP and UDP

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2626780, I Communications Surveys & Tutorials

1

**De-ossifying the Internet Transport Layer: A Survey and Future Perspectives**

Giorgos Papastergiou, Gorry Fairhurst, David Ros, Anna Brunstrom, Karl-Johan Grinnemo, Per Hurnig, Naeem Khademi, Michael Tüxen, Michael Welzl, Dragana Damjanovic and Simone Mangiante

**Abstract**—It is widely recognized that the Internet transport layer has become ossified, where further evolution has become hard or impossible. This is a consequence of the ubiquitous deployment of transport protocols that hamper the adoption of new transports, aggravated further by the limited flexibility of the Application Programming Interface (API) typically presented to applications. To tackle this problem, a wide range of solutions have been proposed in the literature, each aiming to address specific issues. Yet, no single solution has emerged that is able to tackle all aspects of the transport layer. In this work, after an overview of the main issues and reasons for transport-layer ossification, we survey proposed solutions and discuss their potential and limitations. The survey is divided into five parts, each covering a set of possible solutions for a specific facet of the problem: 1) designing middlebox-proof transports; 2) realizing air interface facilities within the transport layer; 3) enhancing the API between the applications and the transport layer; 4) discovering and exploiting end-to-end capabilities; and 5) enabling user-space protocol stacks. Based on this analysis, we then identify further development needs towards an overall solution. We argue that the development of a comprehensive transport layer framework, able to facilitate the integration and cooperation of specialized solutions in an application-independent and flexible way, is a necessary step toward making the Internet transport architecture truly evolvable. To this end, we identify the requirements for such a framework and provide insights for its development.

**Index Terms**—Transport protocols, protocol-stack ossification, API, middleboxes, user-space networking stacks.

**I. INTRODUCTION AND BACKGROUND**

Networks can and do vary significantly in the set of functions they offer and their ability to move data between endpoints. The transport layer operates across the network and is responsible for efficient and robust end-to-end communication between network endpoints. The term end-to-end is often associated with a principle, called the end-to-end argument [1]. This suggests that “functions placed at low levels of a system

G. Papastergiou and D. Ros are with Simula Research Laboratory, P.O. Box 134, 1325 Lysaker, Norway. E-mail: {gpapaste,dros}@simula.no.  
G. Fairhurst is with University of Aberdeen, AB24 3UE Aberdeen, United Kingdom. E-mail: gorry@ab.ac.uk.  
A. Brunstrom, K.-J. Grinnemo and P. Hurnig are with Karlstad University, Universitetsgatan 2, 651 88 Karlstad, Sweden. E-mail: {anna.brunstrom,kar-johan.grinnemo,per.hurnig}@kit.edu.  
M. Welzl is with University of Oslo, P.O. Box 1072 Blinderen, 0316 Oslo, Norway. E-mail: {naeemk,mchawie}@ifi.uio.no.  
M. Tüxen is with Münster University of Applied Sciences, Bismarckstraße 11, 48337 Münster, Germany. E-mail: tuxen@fh-muenster.de.  
D. Damjanovic is with EMC Corporation, Ovens, Co. Cork, Ireland. E-mail: janovic@mozilla.com.  
S. Mangiante is with EMC Corporation, Ovens, Co. Cork, Ireland. E-mail: simone.mangiante@emc.com.

**A. Transport-layer ossification: overview of issues**

Why do developers and users not adopt more modern protocols? It is not because new transports do not meet a real need. The following paragraphs examine the main reasons for this *ossification* of the transport layer.

1553-877X (c) 2016 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

G. Papastergiou *et al*, “De-ossifying the Internet transport layer: A survey and future perspectives”, IEEE Communications Surveys and Tutorials (Nov. 2016).  
<https://dx.doi.org/10.1109/COMST.2016.2626780/>

# Ossification (1/2)

- Deployment experience → if a field in a protocol is visible to the network, someone will implement a middlebox that relies on its presence
- Once a protocol has been widely deployed, very hard to change

## Is it Still Possible to Extend TCP?

Michio Honda\*, Yoshifumi Nishida\*, Costin Raiciu†, Adam Greenhalgh\*,  
Mark Handley\*, Hideyuki Tokuda\*  
Keio University\*, Universitatea Politehnica Bucuresti†, University College London\*  
{micchii,nishida}@stc.wide.ad.jp, costin.raiciu@cs.pub.ro  
{a.greenhalgh,m.handley}@cs.ucl.ac.uk, hxt@ht.stc.keio.ac.jp

### ABSTRACT

We've known for a while that the Internet has ossified as a result of the race to optimize existing applications or enhance security. NATs, performance-enhancing-proxies, firewalls and traffic normalizers are only a few of the middleboxes that are deployed in the network and look beyond the IP header to do their job. IP itself can't be extended because "IP options are not an option" [10]. Is the same true for TCP?

In this paper we develop a measurement methodology for evaluating middlebox behavior relating to TCP extensions and present the results of measurements conducted from multiple vantage points. The short answer is that we can still extend TCP, but extensions' design is very constrained as it needs to take into account prevalent middlebox behaviors. For instance, adaptive sequence numbers cannot be embedded in options, as middleboxes can rewrite ISN and preserve undefined options. Sequence numbering also must be consistent for a TCP connection, because many middleboxes only allow through contiguous flows.

We used these findings to analyze three proposed extensions to TCP. We find that MPTCP is likely to work correctly in the Internet or fallback to regular TCP. TcpCodel seems ready to be deployed, however it is fragile if resegmentation does happen—for instance with hardware offload. Finally, TCP extended options in its current form is not safe to deploy.

### Categories and Subject Descriptors

C.2.2 [Computer-communication Networks]: Network Protocols;  
C.2.6 [Computer-communication Networks]: Internetworking

### General Terms

Measurement, Design, Experimentation, Standardization

### Keywords

Middleboxes, Measurements, TCP, Protocol design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
*IMC '11*, November 2–4, 2011, Berlin, Germany.  
Copyright 2011 ACM 978-1-4503-1013-0/11/11 ...\$10.00.

### 1. INTRODUCTION

The Internet was designed to be extensible: routers only care about IP headers, not what the packets contain, and protocols such as IP and TCP were designed with options fields that could be used to add additional functionality. The great virtue of the Internet was always that it was *stupid*: it did no task especially well, but it was extremely flexible and general, allowing a proliferation of protocols and applications that the original designers could never have foreseen.

Unfortunately the Internet, as it is deployed, is no longer the Internet as it was designed. IP options have been unusable for twenty years [10] as they cause routers to process packets on their slow path. Above IP, the Internet has benefited (or suffered, depending on your viewpoint) from decades of optimizations and security enhancements. To improve performance [2, 7, 18, 3], reduce security exposure [15, 29], enhance control, and work around address space shortages [22], the Internet has experienced an invasion of middleboxes that *do* care about what the packets contain, and perform processing at layer 4 or higher *within* the network.

The problem now faced by designers of new protocols is that there is no longer a well defined or understood way to extend network functionality, short of implementing everything over HTTP [25]. Recently, we have been working on adding both multipath support [11] and native encryption [5] to TCP. The obvious way to do this, in both cases, is to use TCP options. In the case of multipath, we would also like to stripe data across more than one path. At the end systems, the protocol design issues were mostly conventional. However, it became increasingly clear that no one, not the IETF, not the network operators, and not the OS vendors, knew what will and what will not pass through all the middleboxes as they are currently deployed and configured. Will TCP options pass unchanged? If the sequence space has holes, what happens? If a retransmission has different data than the original, which arrives? Are TCP segments coalesced or split? These and many more questions are crucial to answer if protocol designers are to extend TCP in a deployable way. Or have we already lost the ability to extend TCP just like we did two decades ago for IP?

In this paper we present the results from a measurement study conducted from 142 networks in 24 countries, including cellular, WiFi and wired networks, public and private networks, residential, commercial and academic networks. We actively probe the network to elicit middlebox responses that violate the end-to-end transparency of the original Internet architecture. We focus on TCP, not only because it is by far the most widely used transport protocol, but also because while it is known that many middleboxes modify TCP behavior [6], it is not known how prevalent such middleboxes are, nor precisely what the emergent behavior is with TCP extensions that were unforeseen by the middlebox designers.

181

M. Honda *et al.*, “Is it still possible to extend TCP?”, In Proc. ACM Internet Measurement Conference, Berlin, Nov. 2011.  
<https://dx.doi.org/10.1145/2068816.2068834>



# Ossification (2/2)

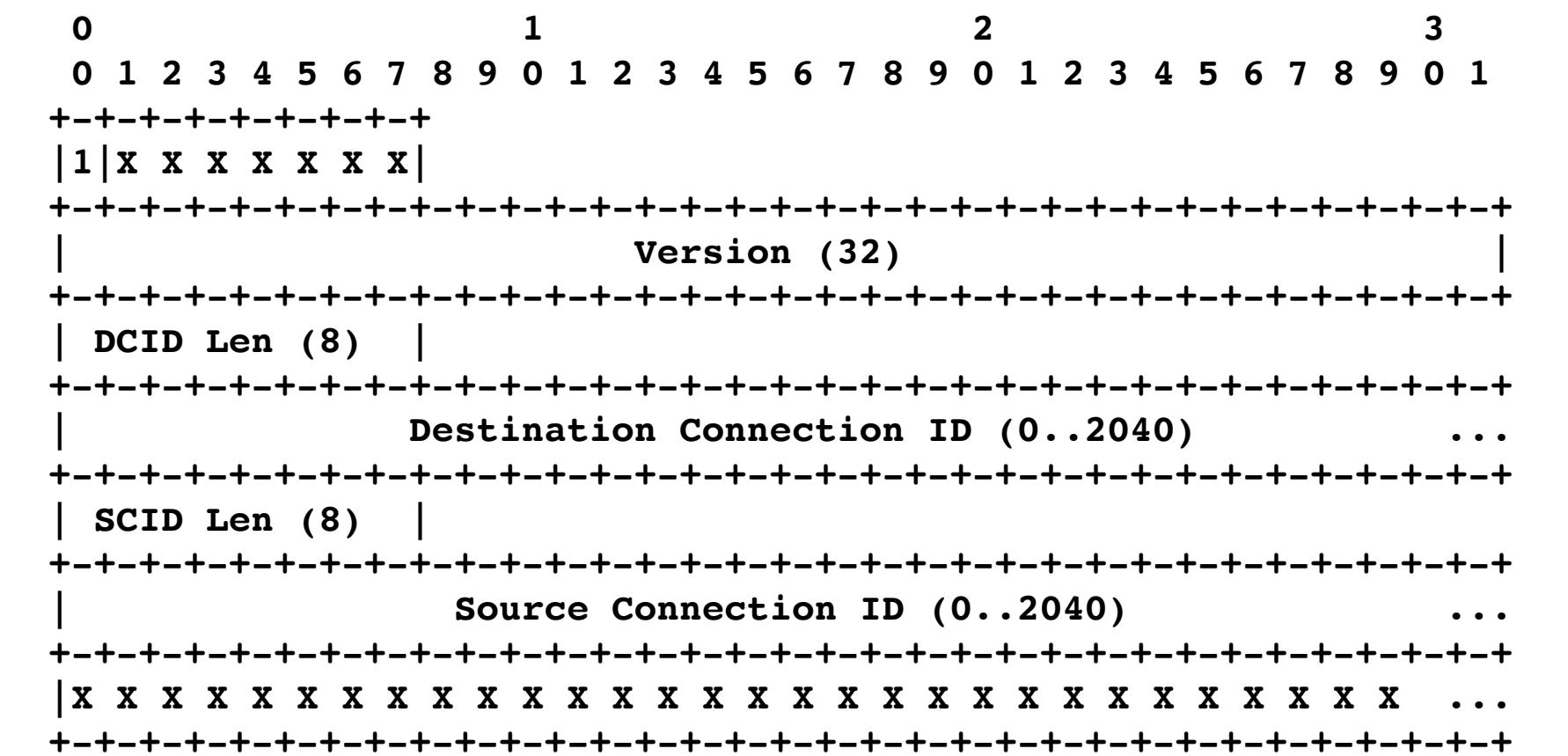
- Protocol ossification affected the design of:
  - TLS 1.3
  - Multipath TCP
  - TCP Fast Open
  - TCP Selective Acknowledgements
  - ...
- Increasingly viewed as a problem in the standards community
  - Difficult to evolve network protocols to address new requirements
  - A system that can no longer evolve and change will die

# Avoiding Ossification in QUIC

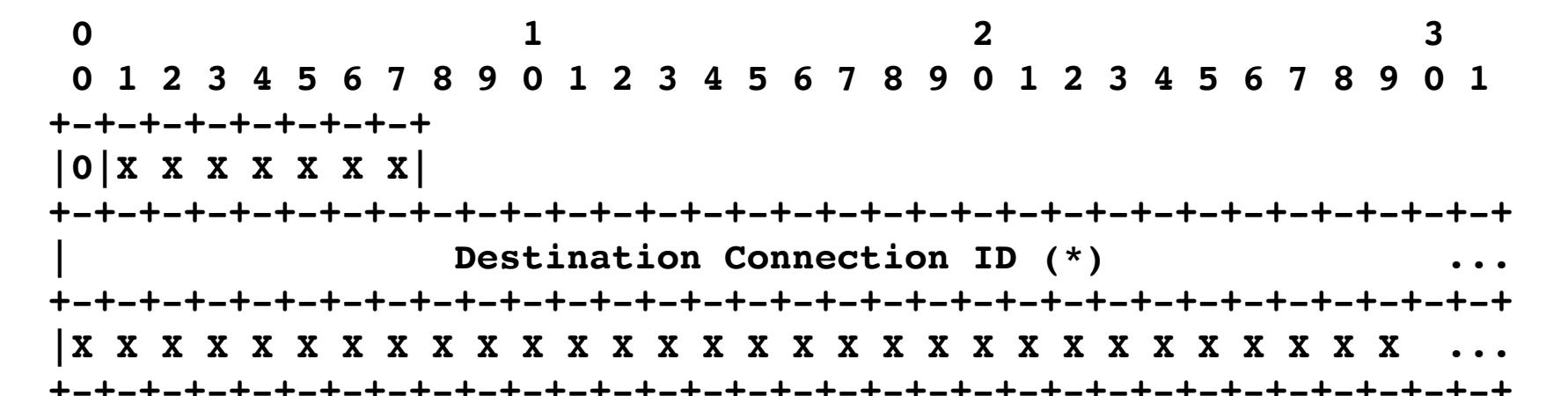
- Three tools to prevent ossification of QUIC:
  - Published protocol invariants
  - Pervasive encryption of transport headers
  - GREASE
- QUIC is a new design – want to **avoid ossification**, starting with initial deployments
- Design the protocol to make it difficult, ideally impossible, for middleboxes to interfere with QUIC connections

# QUIC Invariants

- Properties of QUIC that will remain unchanged as new versions of the protocol developed
  - Explicit guidance to middlebox designers: can assume these properties of QUIC will not change
  - The IETF **will** change other fields or properties between QUIC versions
- What invariants are guaranteed?
  - Packets start with a long header or a short header
    - The first bit of long header packets is always 1; they include version number, destination- and source-connection identifiers
    - The first bit of short header packets is always 0; they include a destination connection identifier
  - Connections can arbitrarily switch between long header and short header packets



QUIC Long Header Invariants



QUIC Short Header Invariants

x = bit may take arbitrary value

<https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/>

# Avoiding Ossification via Pervasive Encryption

- QUIC encrypts as much data as possible
  - Entire packet **except** invariant fields and the last 7-bits of the first byte is encrypted; entire packet is authenticated
  - Encryption keys for initial handshake packets are derived from connection identifiers
  - Contain **ClientHello** and **ServerHello**, which are unencrypted when using TLS over TCP, and **don't need to be encrypted**
  - QUIC provides no more security than TLS over TCP, but makes it **expensive** for middleboxes to read handshake
  - Rest of QUIC connection protected by TLS 1.3 as normal
- QUIC authenticates **all** data
  - If a middlebox changes the headers, the change can be detected

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  
+++-+-+-+--+  
|1|x x x x x x|  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| | Version (32) |  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| DCID Len (8) |  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| | Destination Connection ID (0..2040) | ...  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| SCID Len (8) |  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| | Source Connection ID (0..2040) | ...  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
|x ...  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
  
QUIC Long Header Invariants

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  
+++-+-+-+--+  
|0|x x x x x x|  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
| | Destination Connection ID (\*) | ...  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
|x ...  
+++-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+  
  
QUIC Short Header Invariants

x = bit may take arbitrary value

<https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/>

# Avoiding Ossification via GREASE

- QUIC makes extensive use of GREASE
    - Every field encrypted or has a value that is unpredictable
      - New connections use randomly chosen connection identifiers
      - Clients randomly try to negotiate new version numbers
        - Version numbers matching **0x?a?a?a?a** for testing version negotiation – will be rejected by servers
      - Unused header fields given randomly chosen values
    - Goal is that middleboxes can't make any assumptions about QUIC header values → nothing in the header is predictable
    - Hopefully avoids ossification → nothing to ossify around

QUIC Long Header Invariants

QUIC Short Header Invariants

`x = bit` may take arbitrary value

<https://datatracker.ietf.org/doc/draft-ietf-quic-invariants/>

# QUIC Benefits and Costs

- **Why is QUIC desirable?**
  - Reduces secure connection establishment latency
  - Reduces risk of ossification; easy to deploy
  - Supports multiple streams within a single connection
- **Why is QUIC problematic?**
  - Libraries and support new, poorly documented, and frequently buggy
  - CPU usage is high compared to TLS-over-TCP
    - TCP stack currently much better optimised
    - TCP and TLS often has hardware offload; QUIC doesn't yet
  - These issues will be resolved – but it will take some years before QUIC is as stable and performant as TLS-over-TCP
- **TCP lasted 40 years – QUIC is a similarly long-term project, that's only just reached version 1.0**



Q U I C

# Improving Secure Connection Establishment

- Limitations of TLS 1.3
- QUIC transport protocol
- TCP has dominated for 40 years – is QUIC the future?

# Reliability and Data Transfer

Networked Systems (H)  
Lecture 5



# Packet Loss in the Internet

- Best effort traffic
- The end-to-end argument
- Timeliness vs. reliability trade-off

# Packet Loss in the Internet

- The Internet is a best effort packet delivery network – **it is unreliable**
  - IP packets may be lost, delayed, reordered, or corrupted in transit
  - How often this happens varies significantly
    - Wireless links are less reliable than wired links
    - Countries with well-developed infrastructure tend to have reliable Internet links; countries with less robust or lower capacity infrastructure tend to see more problems
    - Some protocols intentionally try to push links to capacity, causing temporary overload as they try to find the limit
      - TCP and QUIC do this, when certain widely congestion control algorithms are used → lecture 6
    - **Some packet loss is inevitable**
  - The **transport layer** must adapt the quality of service provided by the network to match application needs

# The End-to-End Argument

- Is it better to place functionality within the network or at the end points?
- Only put functionality that is absolutely necessary in the network, leave everything else to end systems
  - **Example:** let the network provide best effort packet delivery, rather than try to detect and retransmit lost packets
  - If the network is not guaranteed to be **100% reliable**, always, end systems must check for lost packets anyway
  - Since 100% reliability can never be guaranteed, no point in complicating the network trying to make it reliable
- One of the defining principles of the Internet

## End-To-End Arguments in System Design

J. H. SALTZER, D. P. REED, and D. D. CLARK  
Massachusetts Institute of Technology Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. Examples discussed in the paper include bit-error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgment. Low-level mechanisms to support these functions are justified only as performance enhancements.

CR Categories and Subject Descriptors: C.0 [General] Computer System Organization—system architectures; C.2.2 [Computer-Communication Networks]: Network Protocols—protocol architecture; C.2.4 [Computer-Communication Networks]: Distributed Systems; D.4.7 [Operating Systems]: Organization and Design—distributed systems

General Terms: Design

Additional Key Words and Phrases: Data communication, protocol design, design principles

### 1. INTRODUCTION

Choosing the proper boundaries between functions is perhaps the primary activity of the computer system designer. Design principles that provide guidance in this choice of function placement are among the most important tools of a system designer. This paper discusses one class of function placement argument that has been used for many years with neither explicit recognition nor much conviction. However, the emergence of the data communication network as a computer system component has sharpened this line of function placement argument by making more apparent the situations in which and the reasons why it applies. This paper articulates the argument explicitly, so as to examine its nature and to see how general it really is. The argument appeals to application requirements and provides a rationale for moving a function upward in a layered system closer to the application that uses the function. We begin by considering the communication network version of the argument.

This is a revised version of a paper adapted from End-to-End Arguments in System Design by J. H. Saltzer, D.P. Reed, and D.D. Clark from the 2nd International Conference on Distributed Systems (Paris, France, April 8–10) 1981, pp. 509–512. © IEEE 1981.  
This research was supported in part by the Advanced Research Projects Agency of the U.S. Department of Defense and monitored by the Office of Naval Research under contract N00014-75-C-0661.

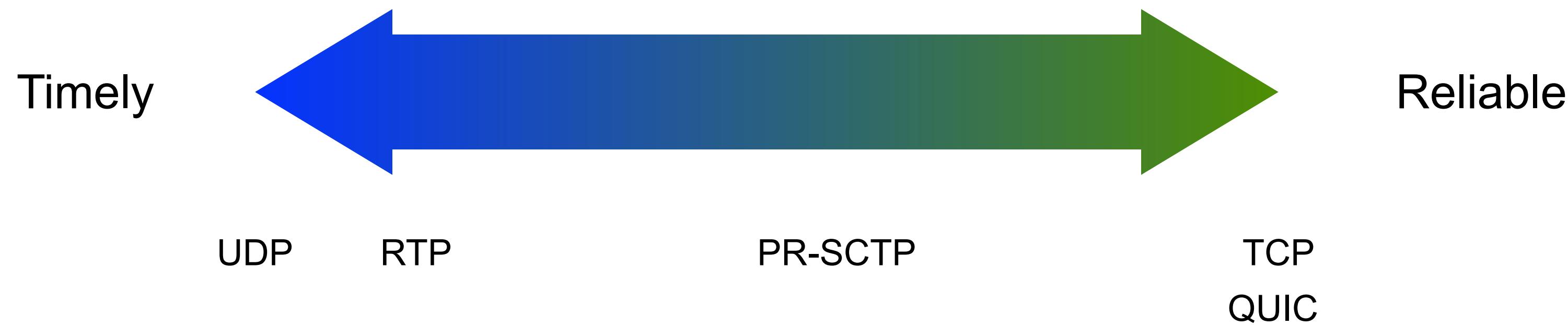
Authors' address: J. H. Saltzer and D. D. Clark, M.I.T. Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. D. P. Reed, Software Arts, Inc., 27 Mica Lane, Wellesley, MA 02181.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.  
© 1984 ACM 0734-2071/84/1100-0277 \$00.75

ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984, Pages 277–288.

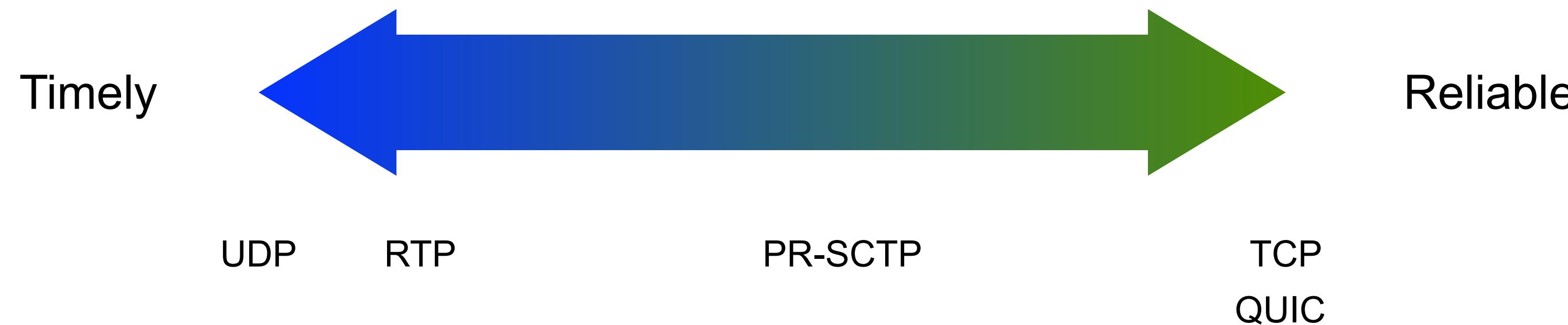
J. Saltzer, D. Reed, and D. Clark, "End-to-end arguments in system design", ACM Transactions on Computer Systems, November 1984. <http://dx.doi.org/10.1145/357401.357402>

# Timeliness vs Reliability Trade-off (1/2)



- Repairing or retransmitting lost packets takes time
- Fundamental trade-off:
  - If a connection is to be reliable, it **cannot** guarantee timeliness
  - If a connection is to be timely, it **cannot** guarantee reliability

# Timeliness vs Reliability Trade-off (2/2)



- Different applications make different timeliness vs. reliability trade-offs:
  - Web, email, etc. → data must be delivered in order sent; no strong timeliness requirement
  - Telephony and video conferencing → tolerates some data loss, but requires timeliness
- Implication for network architecture:
  - Network layer should provide a timely but unreliable service
  - Transport layer protocols can add reliability, if needed

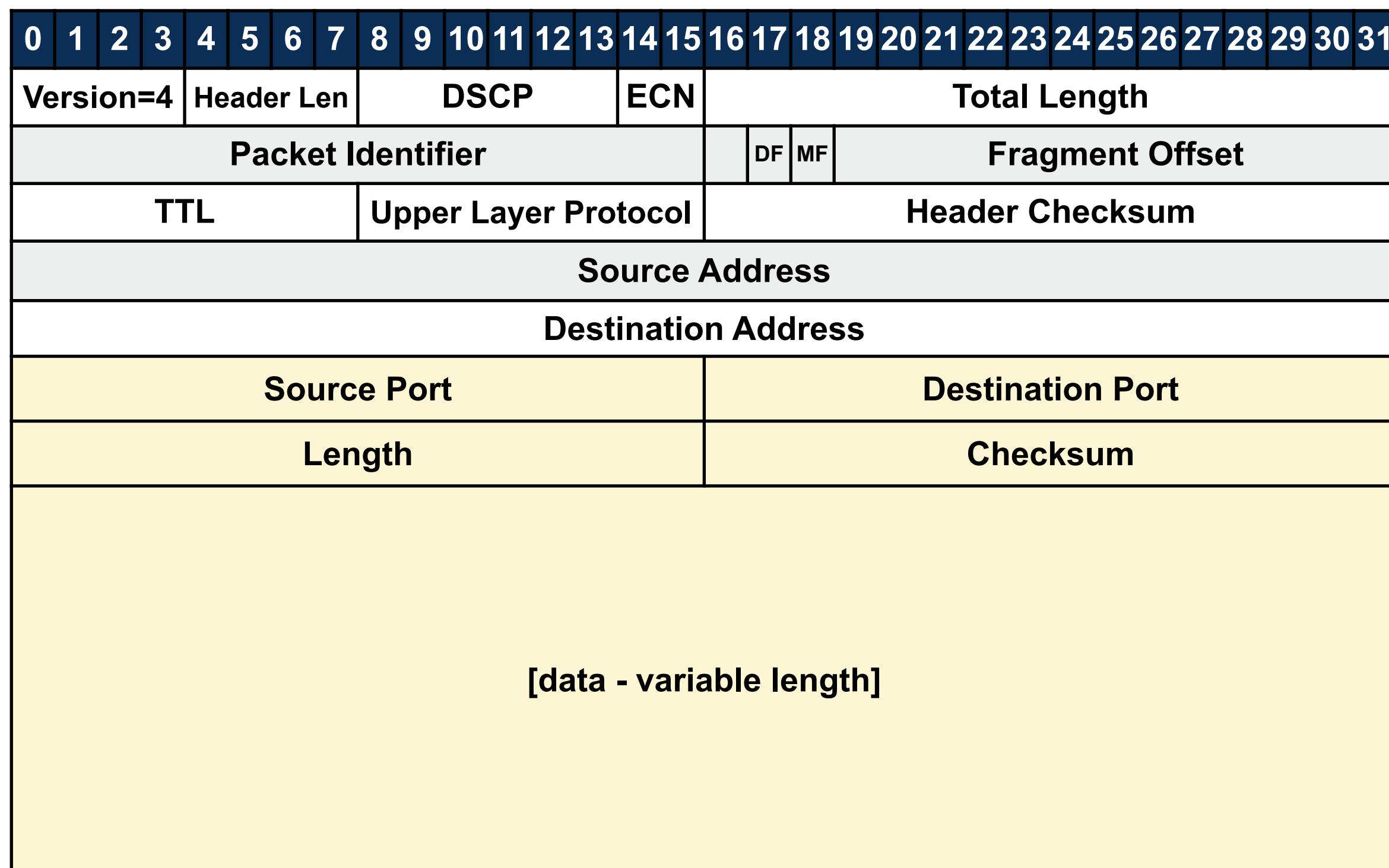
# Packet Loss in the Internet

- Best effort traffic
- The end-to-end argument
- Timeliness vs. reliability trade-off

# Unreliable Data Using UDP

- UDP service model
- Sending and receiving data
- Higher-layer protocols

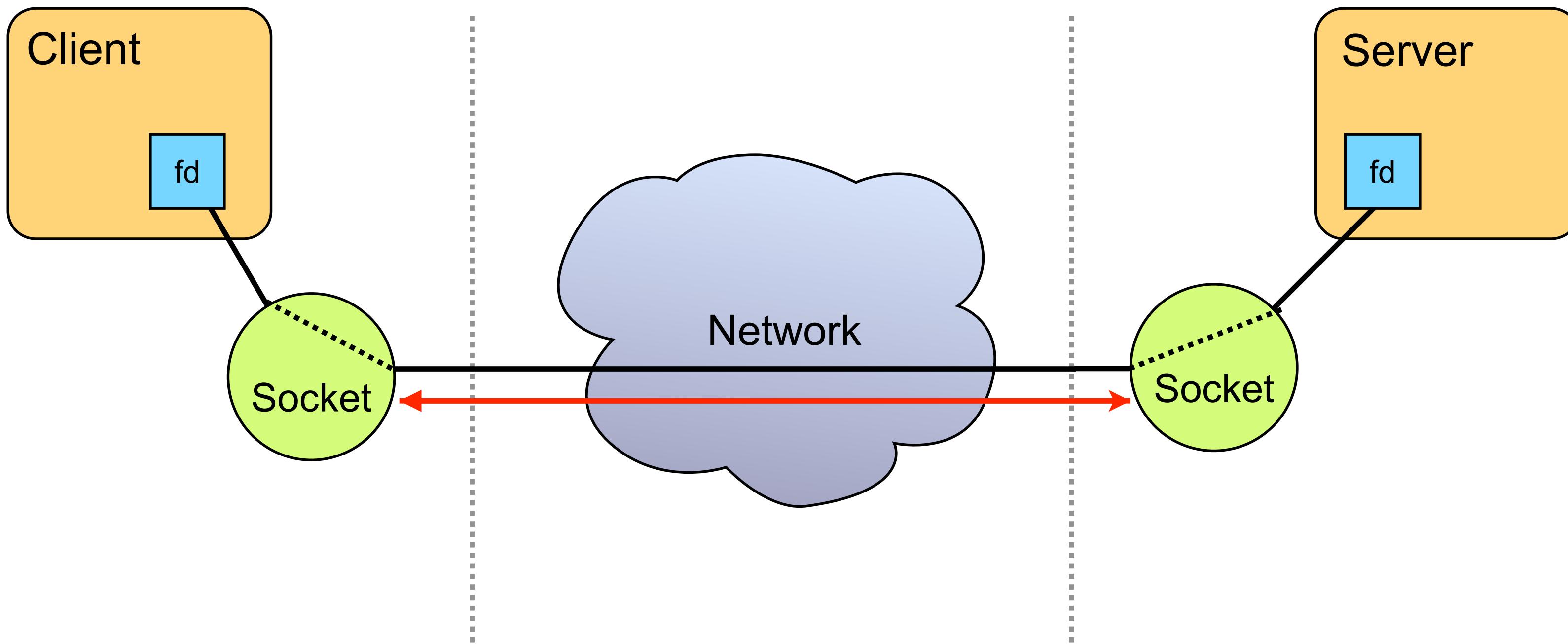
# Packet Loss and UDP



- UDP → **unreliable**, connectionless, datagram service
  - Adds port numbers and a checksum to IP
  - Does not provide reliability or congestion control – best effort packet delivery
- A **substrate for building new transport protocols**
  - QUIC → Lecture 4
  - RTP → Lecture 7
  - DNS → Lecture 8

Well-known UDP port numbers: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

# Using UDP Datagrams



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

bind(fd, ..., ...)

sendto(fd, data, datalen, addr, addrlen) <-----  
recvfrom(fd, buffer, buflen, flags, addr, addrlen) .....
```

close(fd)

# Sending UDP Datagrams

- The **sendto()** call sends a single datagram
- Each call to **sendto()** can send to a different address, even though they use the same socket.

```
int fd;
char buffer[...];
int buflen = sizeof(buffer);
struct sockaddr_in addr;
...
if (sendto(fd, buffer, buflen, (struct sockaddr *) &addr, sizeof(addr)) < 0) {
    // Error...
}
```

- Alternatively, **connect()** to an address, then use **send()** to send the data
  - There is no connection made by the UDP layer, a **connect()** call only sets the destination address for future packets.

# Receiving UDP Datagrams

- The `recv()` call may be used to read a single datagram, but doesn't provide the source address of the datagram.
- Most code uses `recvfrom()` instead – this fills in the source address of the received datagram:

```
int          fd;
char         buffer[...];
int          buflen = sizeof(buffer);
struct sockaddr    addr;
socklen_t     addr_len = sizeof(addr);
int          rlen;
...
rlen = recvfrom(fd, buffer, buflen, 0, &addr, &addrlen);
if (rlen < 0) {
    // Error...
}
```

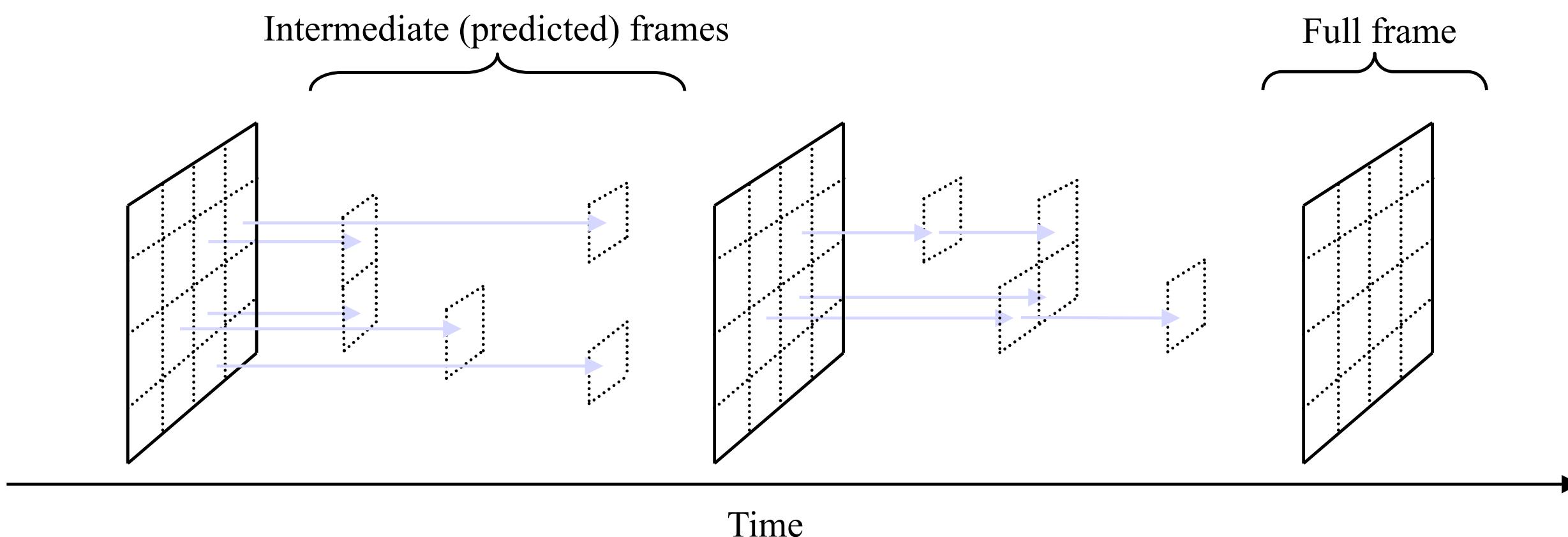
- Packets can be lost, delayed, or reordered in transit – UDP **does not** attempt to recover from this

# UDP Framing and Reliability (1/2)

- Each `sendto()` call generates exactly one UDP datagram
  - Each `recvfrom()` corresponds to a single `sendto()`
- But, UDP is unreliable:
  - Datagrams may be lost, delayed, reordered, or duplicated in transit
  - Data sent with `sendto()` might never arrive
    - Data sent with `sendto()` might arrive more than once
    - Data sent in consecutive calls to `sendto()` might arrive out-of-order
    - UDP **does not** attempt to correct this
- The **protocol built on UDP** is responsible for correcting the order, detecting duplicates, and repairing loss – if necessary
  - The protocol running within UDP generally includes a sequence number to support this
  - e.g., RTP and QUIC both include a packet sequence number inside UDP packets

# UDP Framing and Reliability (2/2)

- UDP provides framing – data is delivered a packet at a time – but is unreliable
- Application must organise the data so it's useful if some datagrams lost
  - e.g. video conferencing with I- and P-frames



See also: D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols", ACM SIGCOMM Conference, Philadelphia, PA, USA, September 1990. <http://dx.doi.org/10.1145/99517.99553>

# UDP Sequencing and Reliability

- UDP does not attempt to provide sequencing, reliability, or timing recovery
  - It provides a best-effort datagram delivery service, and identifies applications according to port numbers – substrate on which application protocols can be implemented
  - The syntax and semantics of the datagrams depends on the application layer protocol running within UDP
    - This might provide sequence numbers and acknowledgements
    - This might provide retransmission and/or forward error correction
    - This might provide timestamps to recover timing
    - This might provide payload identification, or other information

# Guidelines for writing UDP applications

- IETF provides guidelines for writing UDP-based applications
  - RFC 8085 – <https://tools.ietf.org/html/rfc8085>
  - **Read this before trying to write UDP-based applications**

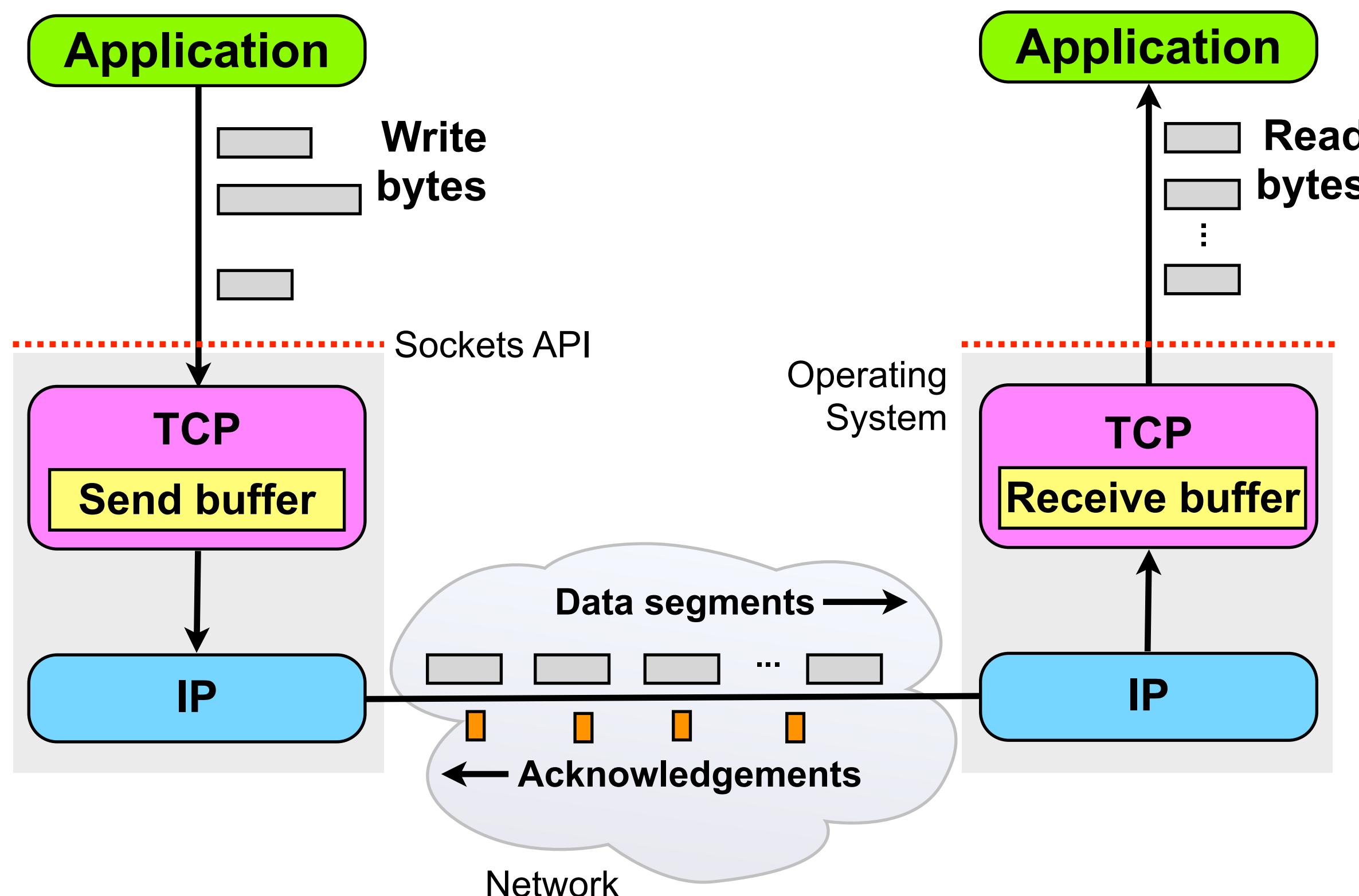
# Unreliable Data Using UDP

- UDP service model
- Sending and receiving data
- Higher-layer protocols

# Reliable Data With TCP

- TCP service model
- Sequence numbers and ACKs
- Packet loss detection and recovery

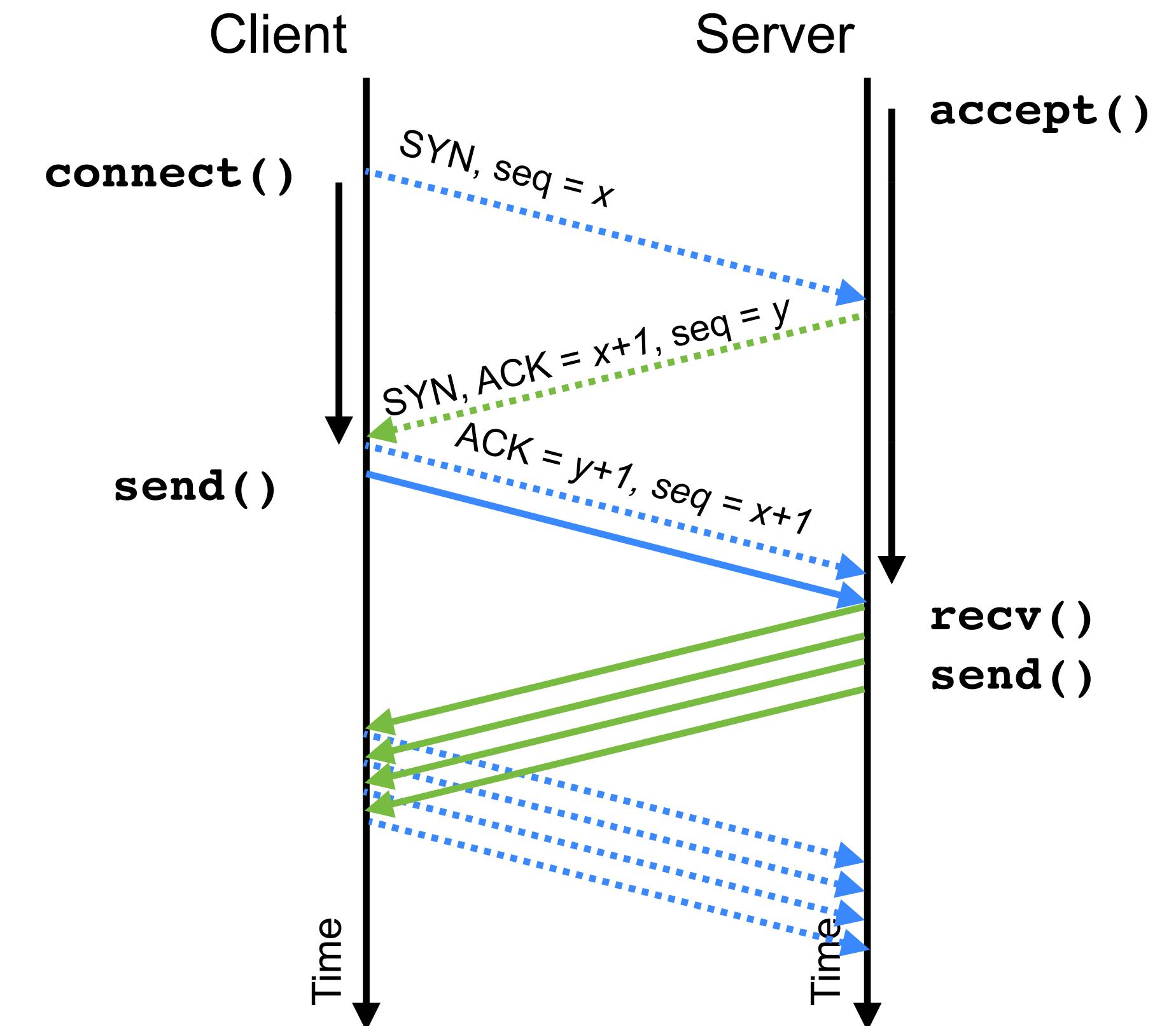
# TCP Service Model



- Reliable, ordered, byte stream delivery service running over IP

# Reliable Data Transfer with TCP

- TCP delivers an ordered, reliable, byte stream
- After connection established, client and server can **send()** or **recv()** data
  - Data can flow in either direction within a TCP connection
  - No requirement that data transfers follow a request-response pattern; client and server can send in any order
- TCP ensures data delivered reliably and in order
  - TCP sends acknowledgements for segments as they are received; retransmits lost data
  - TCP will recover original transmission order if segments are delayed and arrive out of order



# Reading and Writing Data on a TCP Connection (1/2)

```
char data[] = "Hello, world!";
int datalen = strlen(data);
...
int sent = send(fd, data, datalen, 0);
if (sent == -1) {
    // Error has occurred
    ...
} else if (sent < datalen) {
    // Couldn't send it all, retry unsent
    ...
}
```

- The **send()** function transmits data
- Blocks until the data can be written
  - Might not be able to send all the data, if the connection is congested
- Returns actual amount of data sent
  - Returns -1 if error occurs, sets **errno**

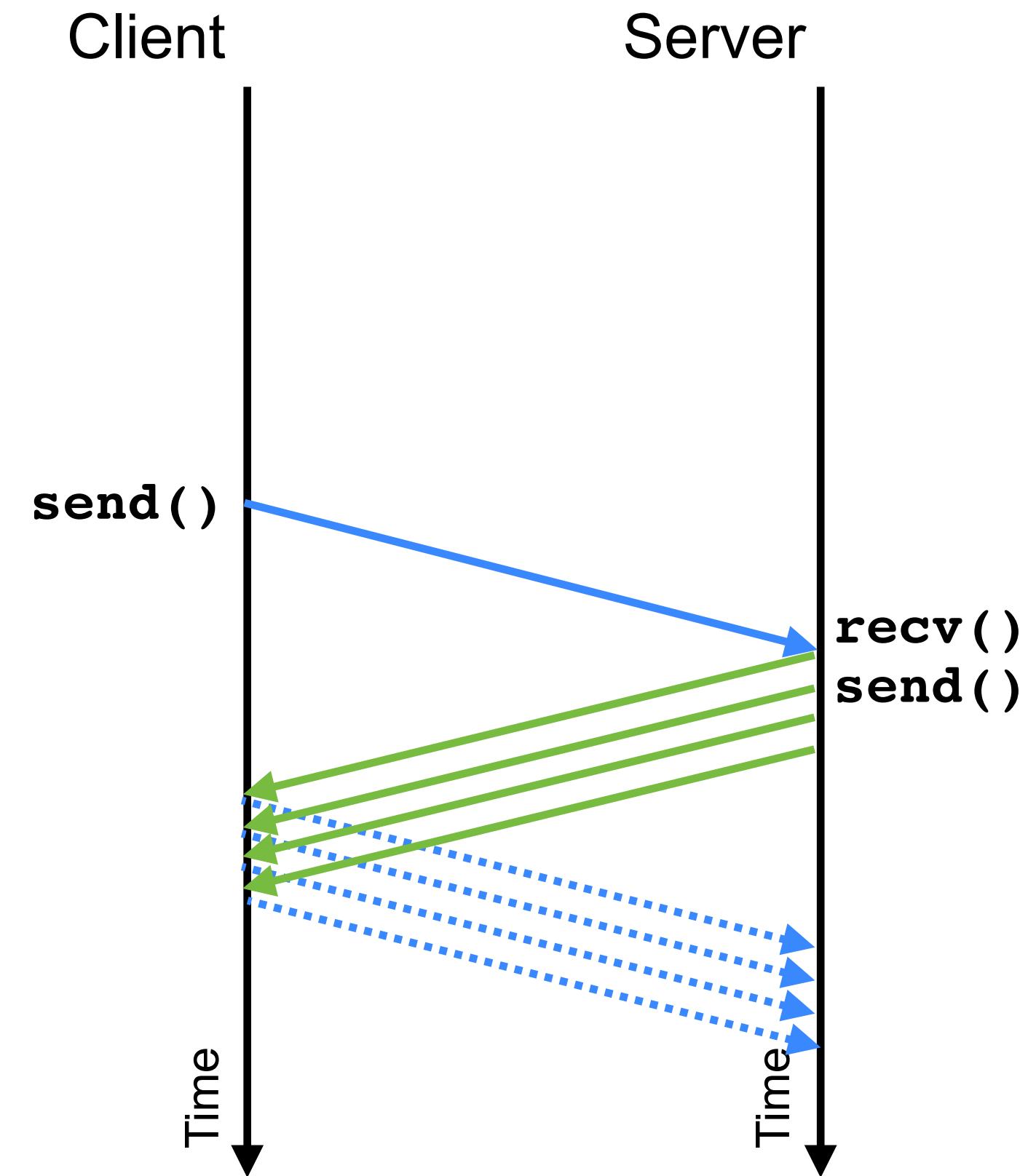
# Reading and Writing Data on a TCP Connection (2/2)

- The `recv()` function receives data
  - Blocks until data available or connection closed; reads up to `BUFSIZE` bytes
  - Returns number of bytes read
    - If connection closed, returns 0
    - If error occurs, returns -1 and sets `errno`
  - Received data is **not** null terminated
  - **This is a significant security risk**

```
#define BUFSIZE 1500
...
ssize_t i;
ssize_t rcount;
char    buf[BUFSIZE];
...
rcount = recv(fd, buf, BUFSIZE, 0);
if (rcount == 0) {
    // Sender closed the connection
}
else if (rcount == -1) {
    // Error has occurred
}
else {
    // Successfully read rcount bytes
    for (i = 0; i < rcount; i++) {
        printf("%c", buf[i]);
    }
    printf("\n");
}
```

# TCP Segments and Sequence Numbers (1/2)

- The `send()` call enqueues data for transmission
- Data is split into **segments**, each placed in a **TCP packet**,
- TCP packets sent in IP packets, when allowed by congestion control algorithm
  - Each segment has a sequence number
  - Sequence numbers start from the value sent in the TCP handshake
    - Initial sequence number is sent in first packet sent from client→server (the packet that has the **SYN** bit set to 1); chosen randomly by the client
    - First data packet has sequence number one higher than the initial packet
    - Subsequent data packets increase sequence number by number of bytes sent
    - Separate sequence number spaces in each direction

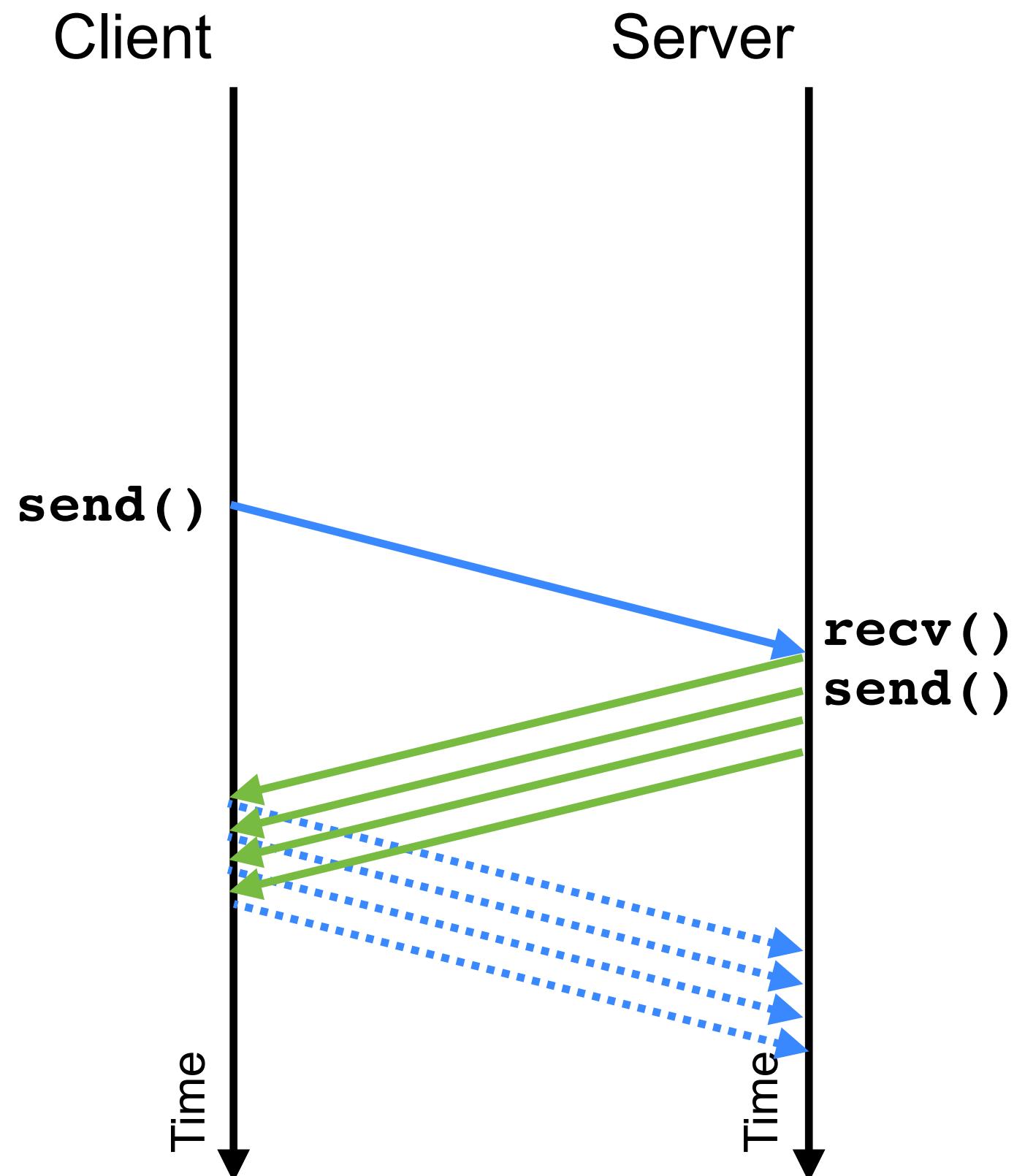


# TCP Segments and Sequence Numbers (2/2)

- Calls to **send()** don't directly map to TCP segments
  - If data given to **send()** is too large to fit in one segment, TCP will split it across several segments
  - If data given to **send()** is small, TCP might wait to send the data, combining it with later data into a single larger segment
    - Known as **Nagle's algorithm** – improves efficiency, but adds some delay
    - Can be disabled with the **TCP\_NODELAY** socket option:

```
int param = 1;

if (setsockopt(fd, SOL_TCP, TCP_NODELAY, &param, sizeof(param)) < 0) {
    // Error occurred
} else {
    // Successfully disabled Nagle's algorithm
}
```



- Implication: data returned by **recv()** does not always correspond to a single **send()** call

# TCP Does Not Preserve Message Boundaries

HTTP Headers

```
HTTP/1.1 200 OK
Date: Mon, 19 Jan 2009 22:25:40 GMT
Server: Apache/2.0.46 (Scientific Linux)
Last-Modified: Mon, 17 Nov 2003 08:06:50 GMT
ETag: "57c0cd-e3e-17901a80"
Accept-Ranges: bytes
Content-Length: 3646
Connection: close
Content-Type: text/html; charset=UTF-8
```

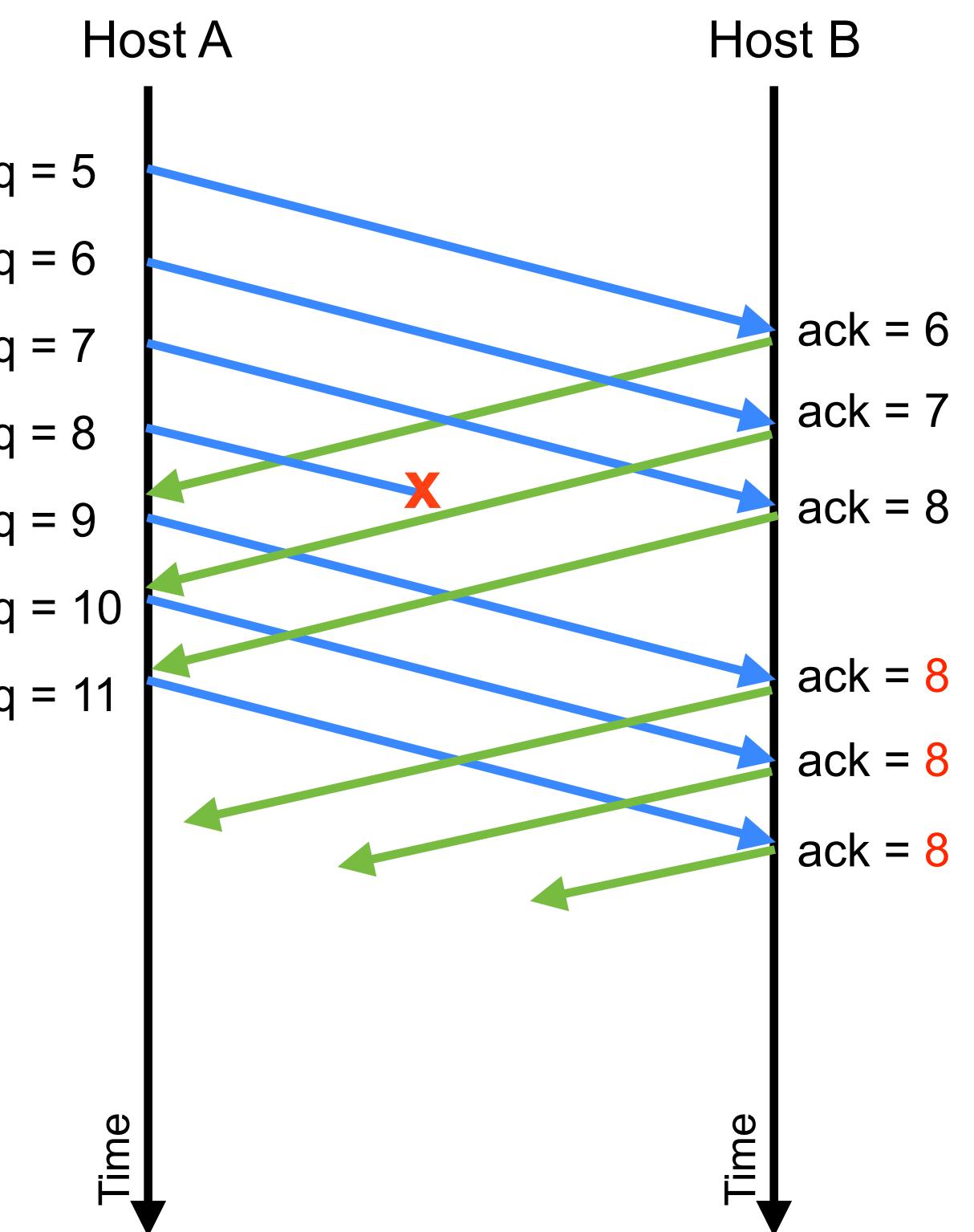
Body

```
<HTML>
<HEAD>
<TITLE>Computing Science, University of Glasgow </TITLE>
...
</BODY>
</HTML>
```

- The `recv()` call returns data reliably, and in the order sent, but doesn't frame data
  - Desirable if one `recv()` call would fetch all the HTTP headers, then another the entire body
  - TCP not does guarantee this – might split the response arbitrarily, e.g., matching the colours
  - Complicates code using TCP
- Implication: you must parse the data to know how much remains to be read

# TCP Acknowledgements

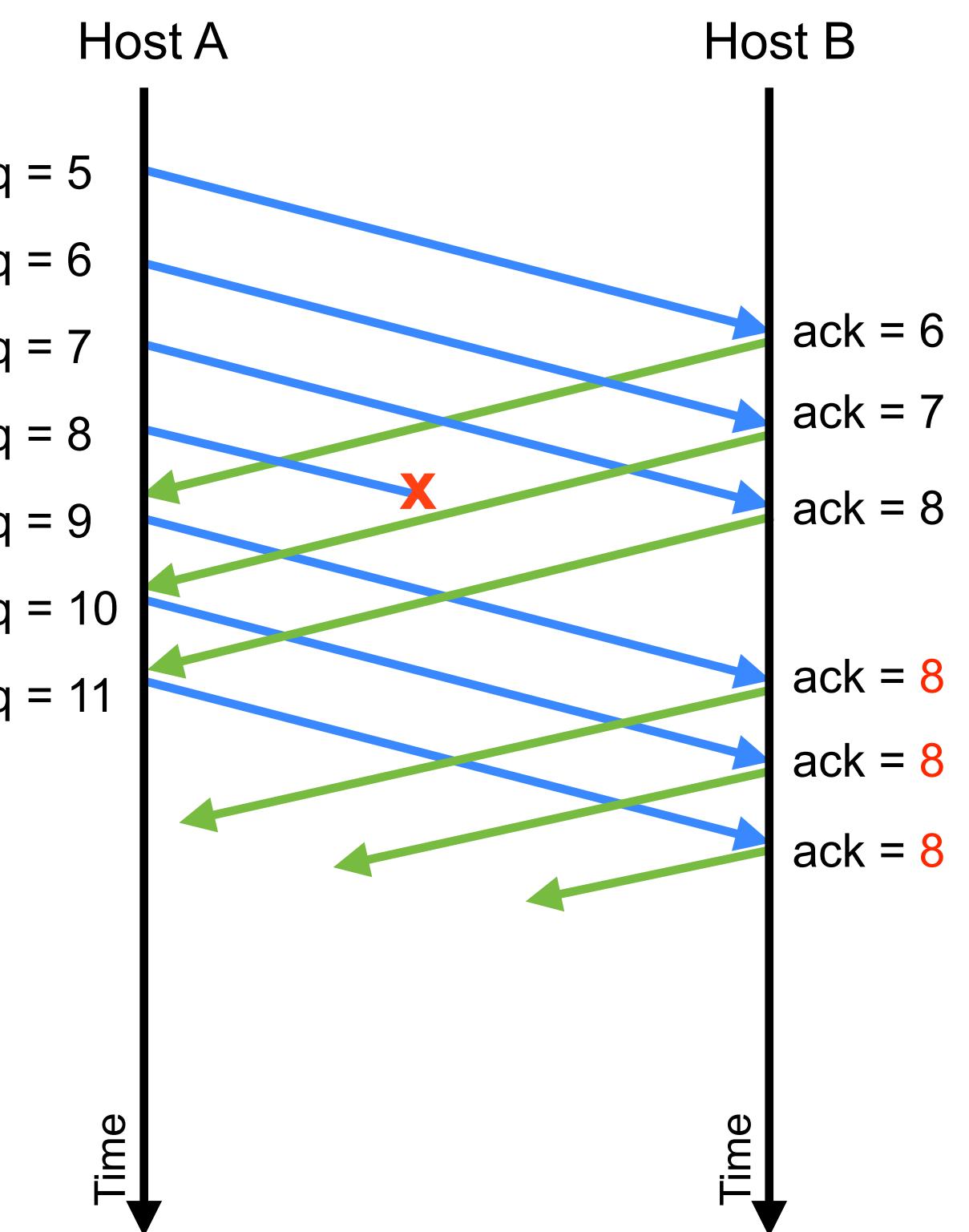
- TCP receiver acknowledges each segment received
  - Each TCP segment has **sequence number** and **acknowledgement number**
  - Segments sent to acknowledge each received segment – contains acknowledgement number indicating sequence number of the **next contiguous byte expected**
    - Includes data if any ready to send, or can be empty apart from acknowledgement
  - If a packet is lost, subsequent packets will generate duplicate acknowledgements
    - In example, segment with sequence number 8 is lost
    - When segment with sequence number 9 arrives, the receiver acknowledges 8 again – since that's still the next contiguous sequence number expected
  - Can send **delayed acknowledgements**
    - e.g., acknowledge every second packet if there is no data to send in reverse direction



TCP connection progress, showing sequence numbers and acknowledgements; in this example, each segment carries one byte of data; the segment with sequence number 8 is lost.

# TCP Acknowledgements: Loss Detection (1/3)

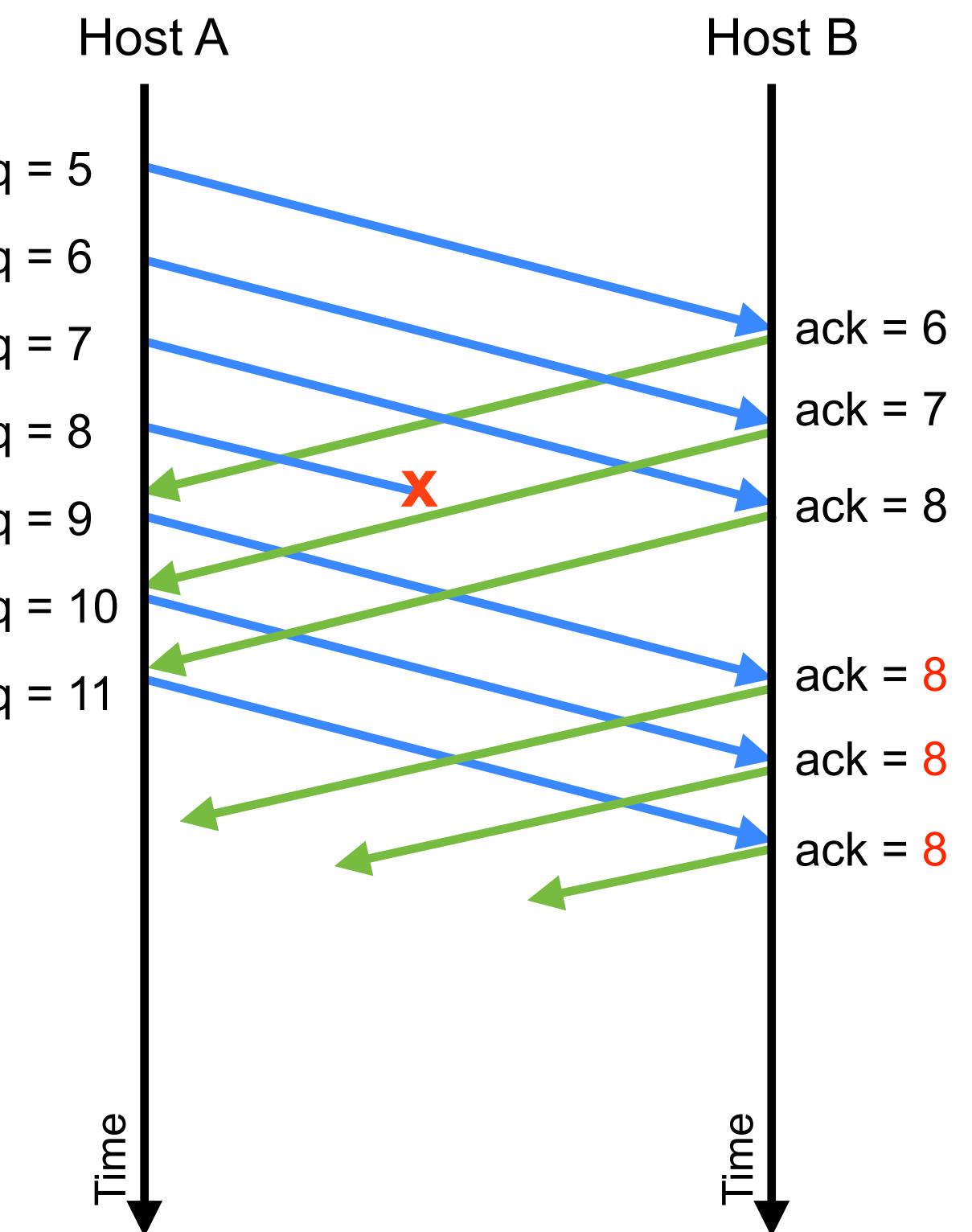
- TCP uses acknowledgements to detect lost segments
  - If data is being sent, but no acknowledgements return, then either receiver or the network failed
  - TCP treats a **timeout** as an indication of packet loss and retransmits unacknowledged segments



TCP connection progress, showing sequence numbers and acknowledgements; in this example, each segment carries one byte of data; the segment with sequence number 8 is lost.

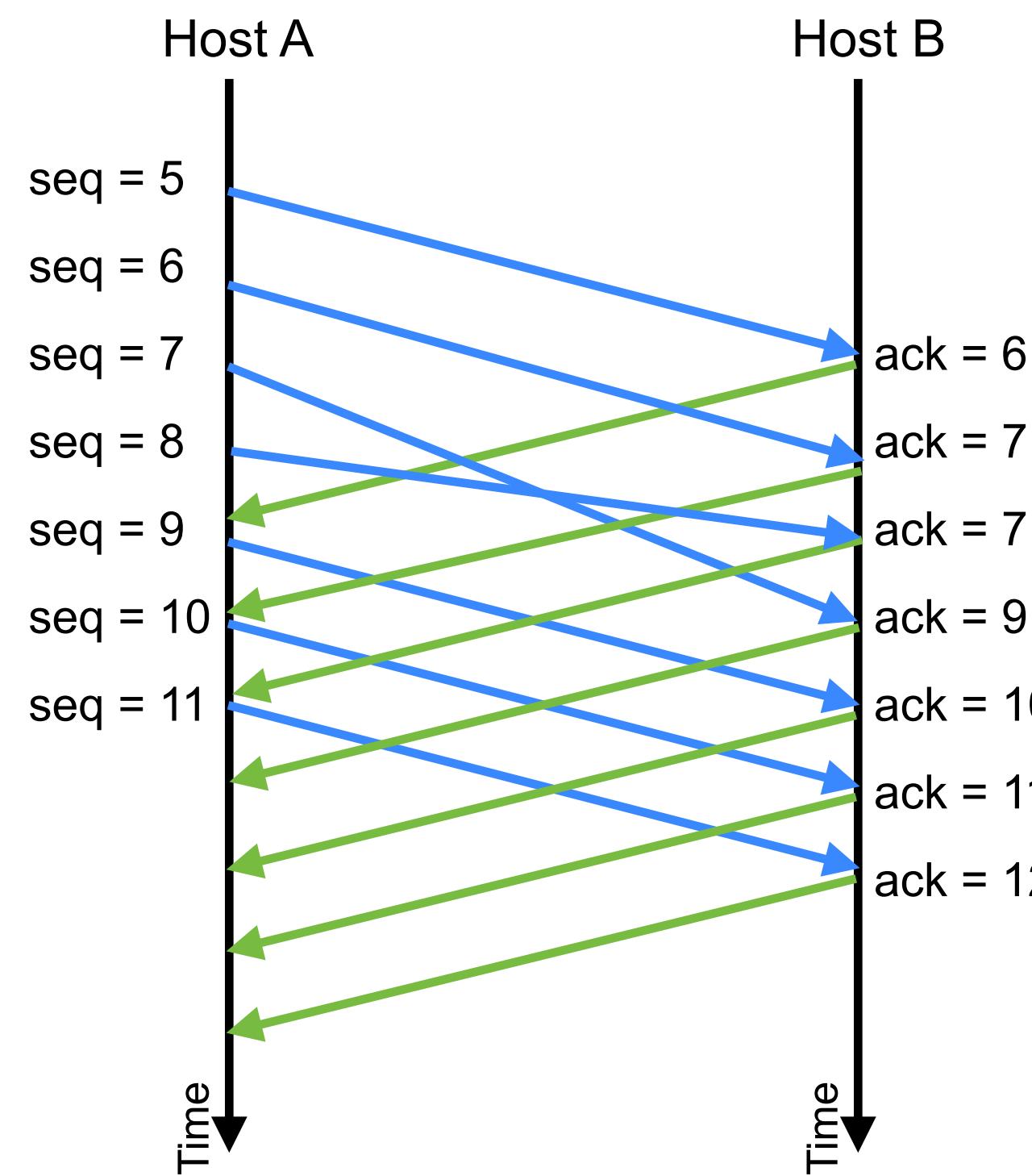
# TCP Acknowledgements: Loss Detection (2/3)

- TCP uses acknowledgements to detect lost segments
  - Duplicate acknowledgements received → some data lost, but later segments arrived
  - TCP treats a **triple duplicate acknowledgement** – four consecutive acknowledgements for the same sequence number – as an indication of packet loss
  - Lost segments are retransmitted



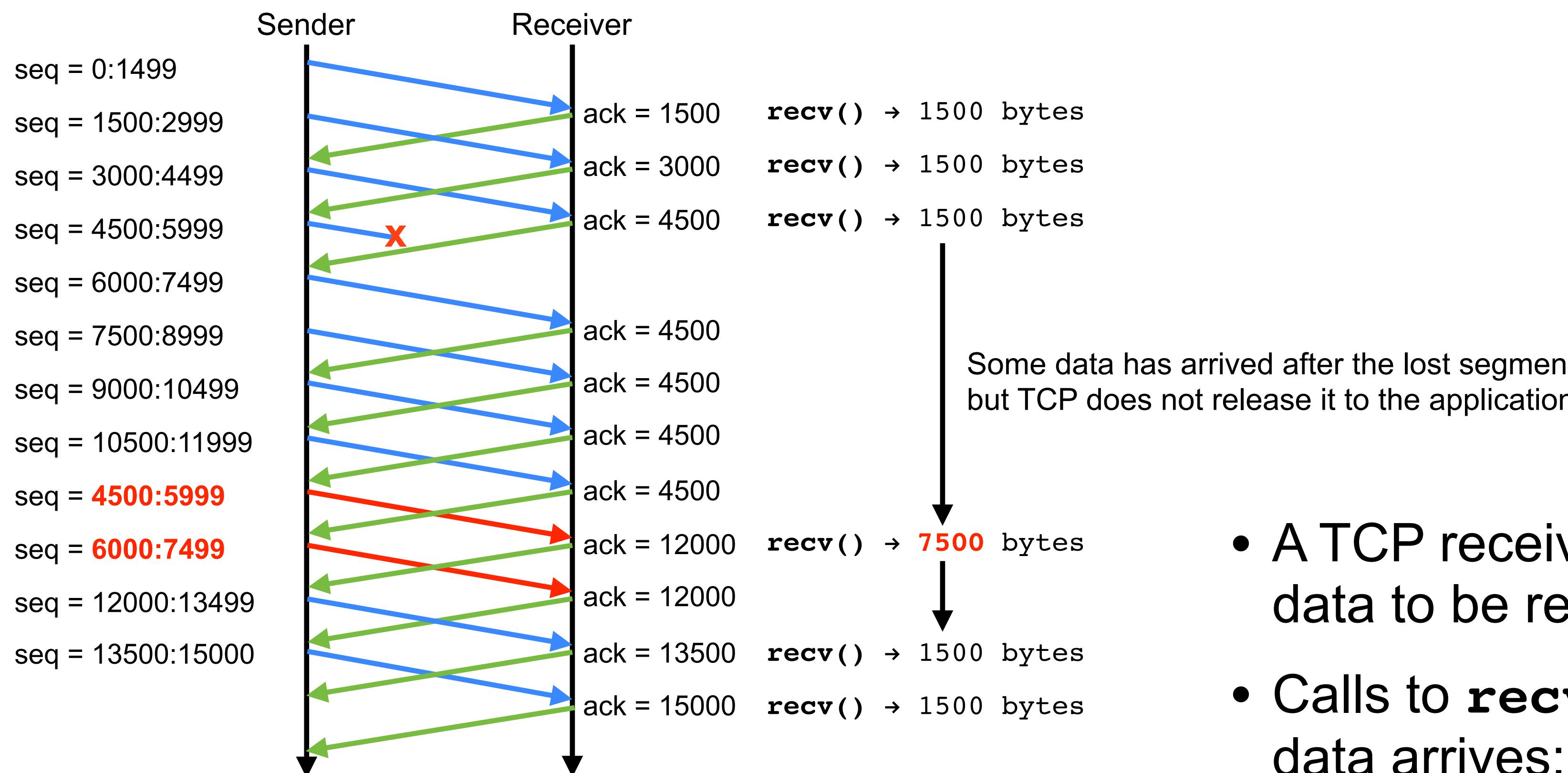
TCP connection progress, showing sequence numbers and acknowledgements; in this example, each segment carries one byte of data; the segment with sequence number 8 is lost.

# TCP Acknowledgements: Loss Detection (3/3)



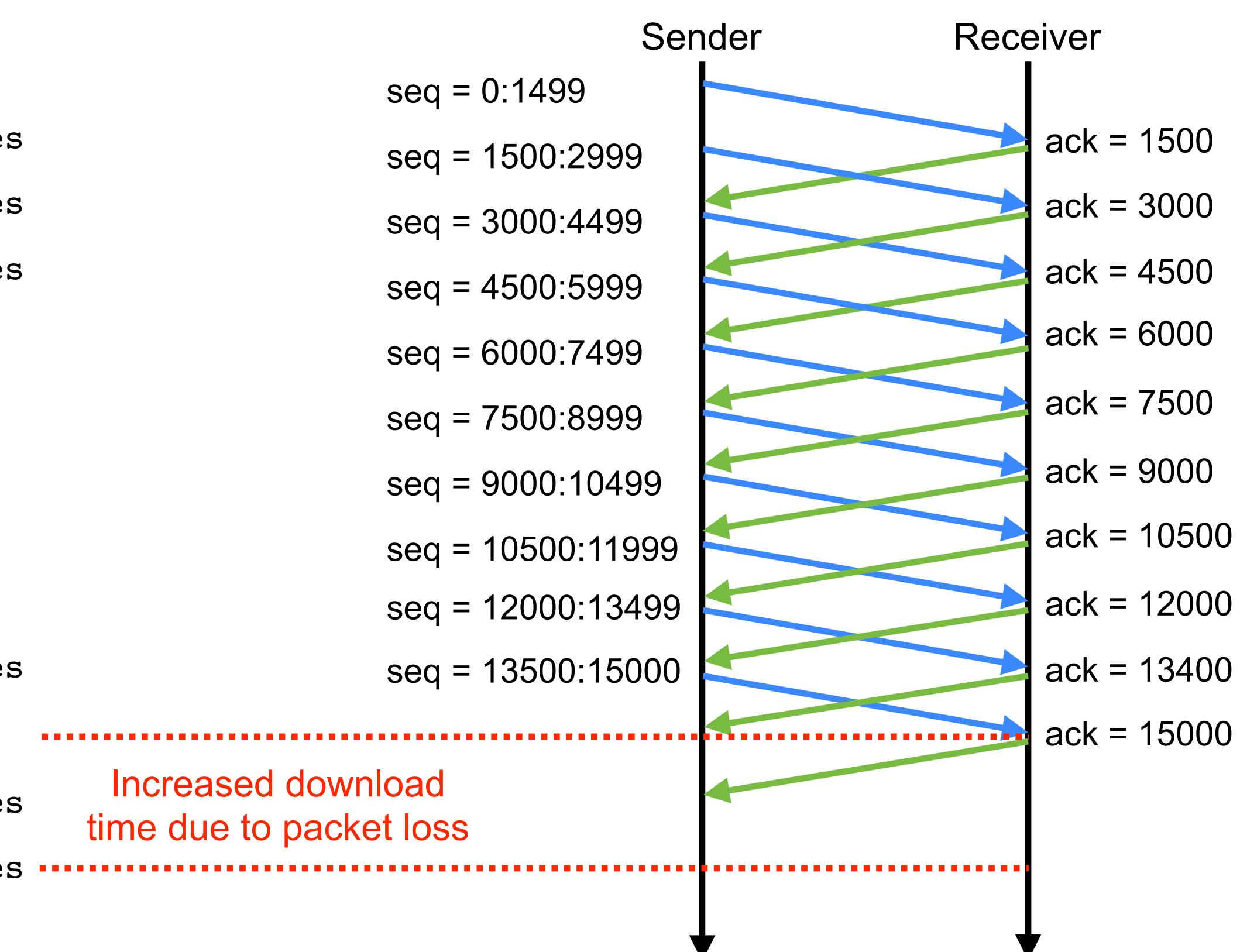
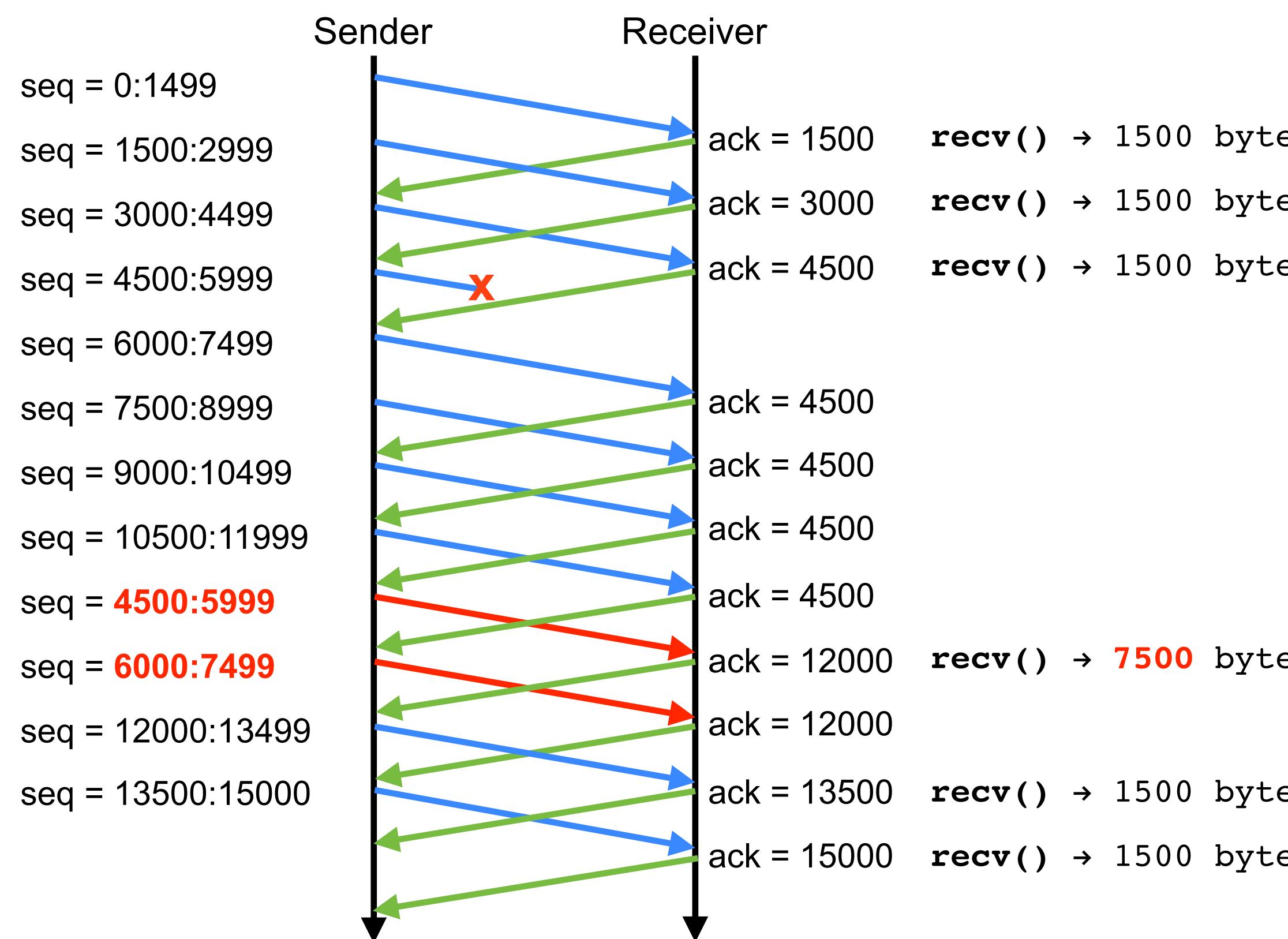
- Why a **triple duplicate acknowledgement**?
  - Packet delay leading to reordering will also cause duplicate acknowledgement to be generated
  - Gives appearance of loss, when the segments are merely delayed
  - Use of triple duplicate ACK to indicate packet loss stops reordered packets from being unnecessarily retransmitted
  - Packets delayed enough to cause one or two duplicate acknowledgements is relatively common
  - Packets being delayed enough to cause three or more duplicates is rare
  - Balance speed of loss detection vs. likelihood of retransmitting a packet that was merely delayed

# Head-of-line Blocking in TCP (1/3)



- A TCP receiver will wait for missing data to be retransmitted
- Calls to `recv()` block until missing data arrives; TCP **always** delivers data to the application in a contiguous, ordered, sequence

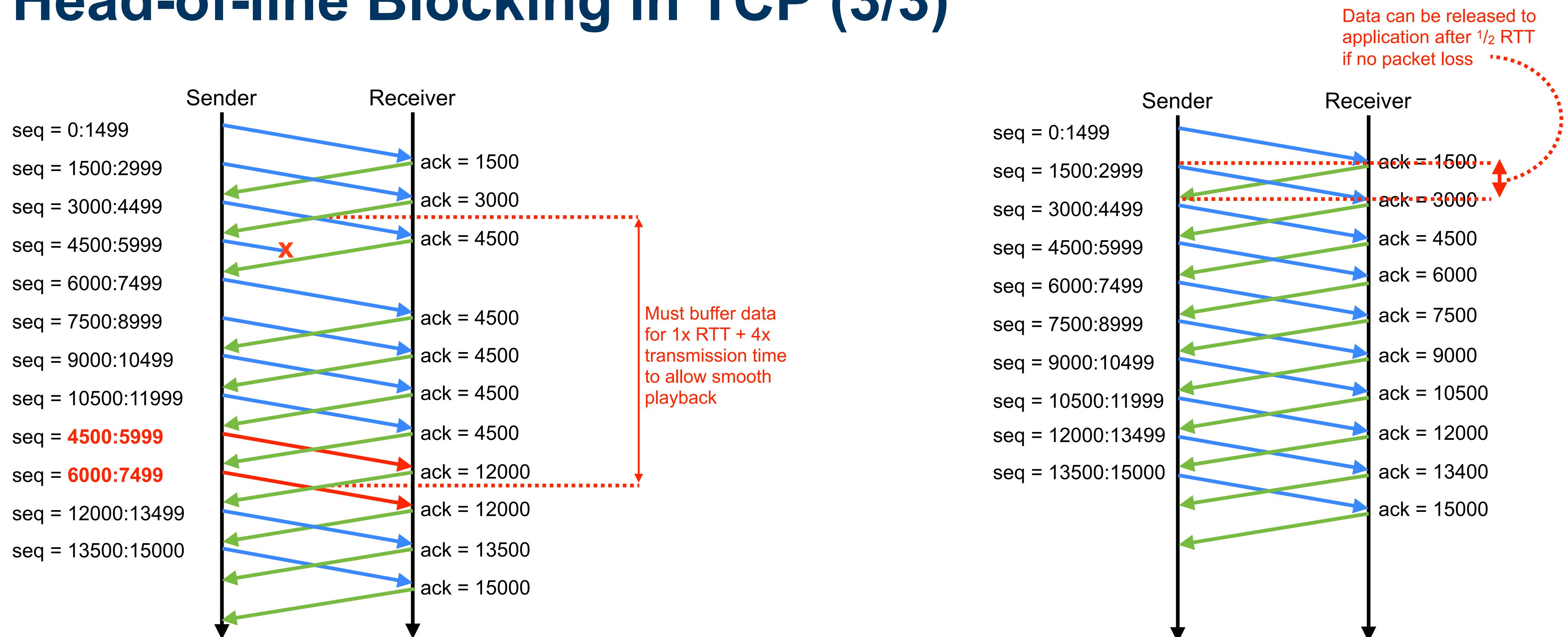
# Head-of-line Blocking in TCP (2/3)



Increased download  
time due to packet loss

- Head-of-line blocking increases total download time
- Delay depends on relative values of RTT and packet serialisation delay

# Head-of-line Blocking in TCP (3/3)



- Head-of-line blocking increases latency for progressive and real-time applications
- **Significant latency increase** if file is being played out as it downloads

# Reliable Data Transfer with TCP

- **TCP provides an ordered, reliable, byte stream service**
  - Service model is simple to understand – it's like reading from a file
  - Head of line blocking can significantly impact progressive and real-time uses
    - MPEG DASH (“Dynamic Adaptive Streaming over HTTP”) → Lecture 7
  - Lack of framing can complicate application design

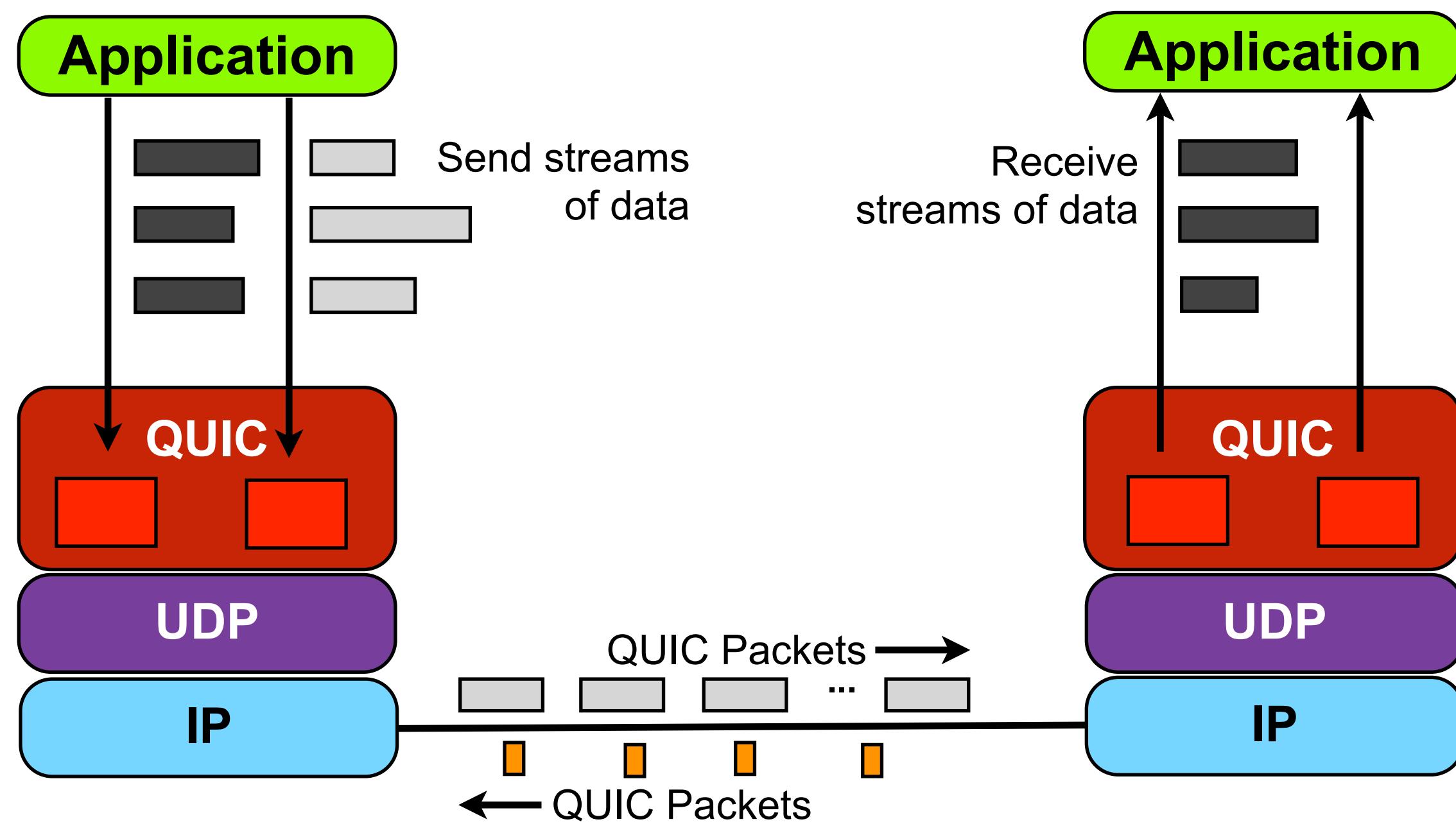
# Reliable Data With TCP

- TCP service model
- Sequence numbers and ACKs
- Packet loss detection and recovery

# Reliable Data Transfer with QUIC

- QUIC service model
- Multi-streaming
- Limited head-of-line blocking

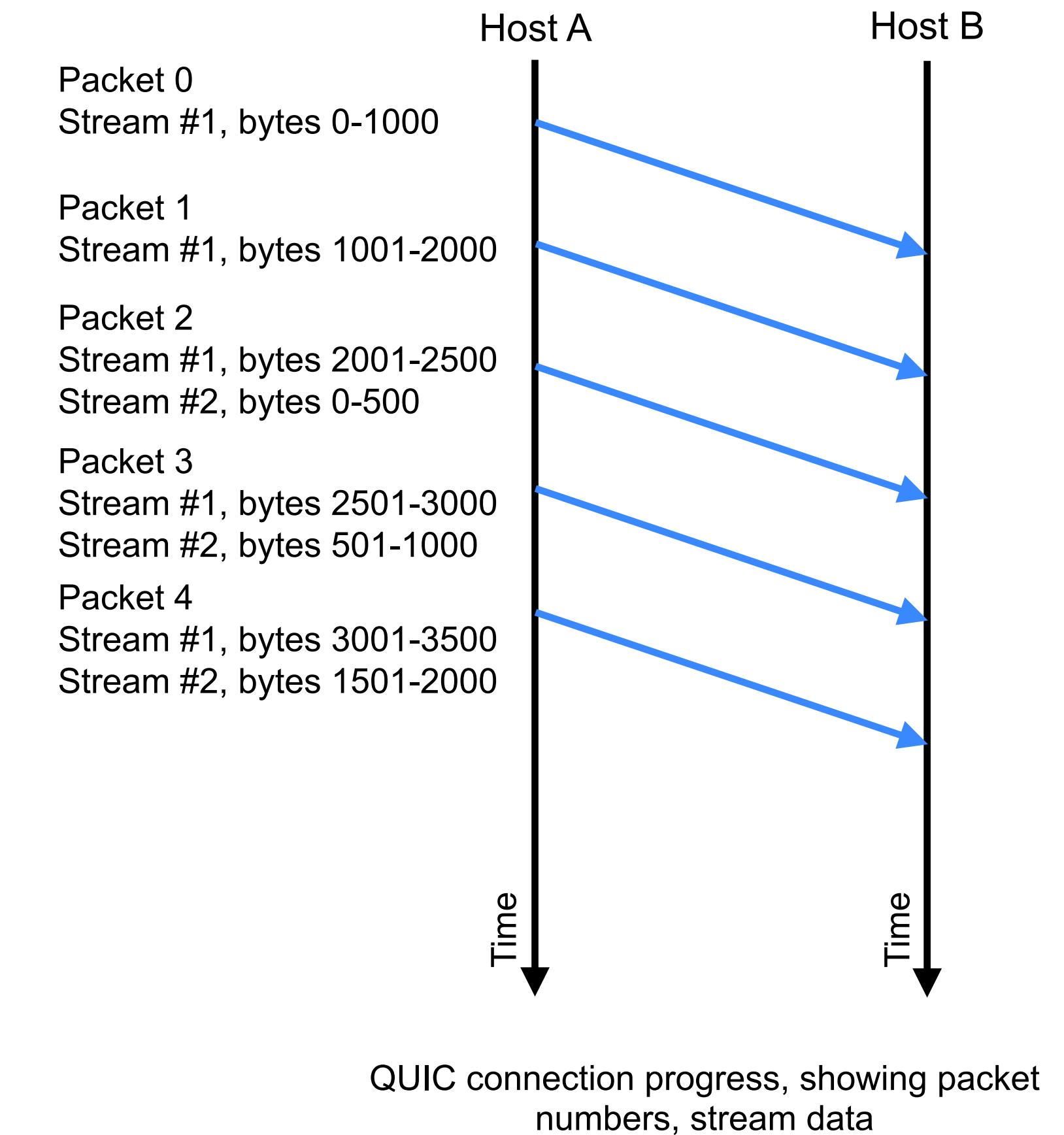
# QUIC Service Model



- TCP delivers reliable, ordered, byte stream
- QUIC delivers **several** ordered reliable byte streams within a single connection
  - Each stream is delivered reliably and in-order
  - Order is not preserved between streams within a QUIC connection

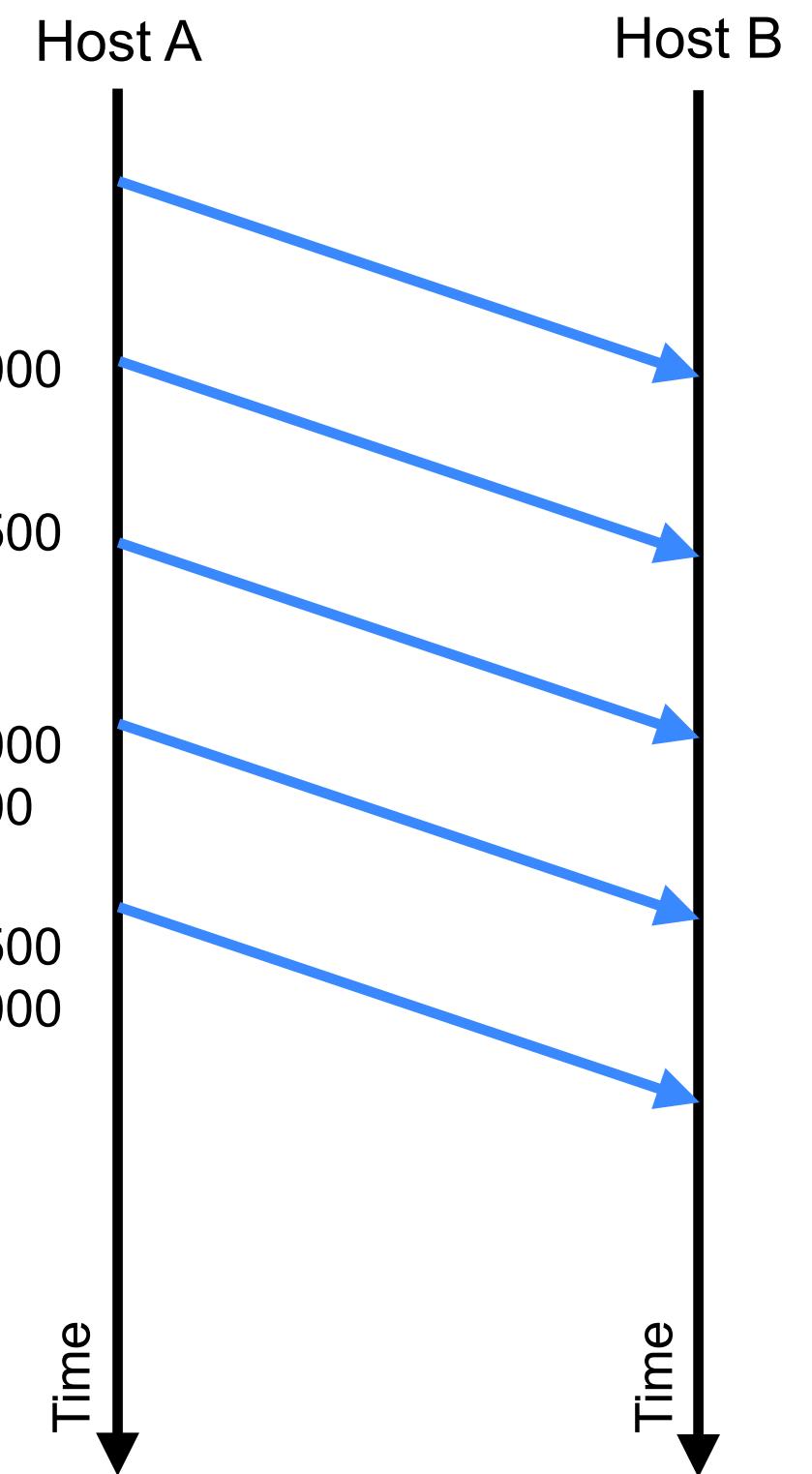
# QUIC Packets and Sequence Numbers (1/2)

- Each QUIC packet has a **Packet Number**
  - Two packet number spaces
    - Packets sent during initial handshake
    - Packets sent to carry data
  - Within each space, packet number starts at zero, increases by one for each packet sent
    - Counts number of packets sent
    - **Different to TCP**, where the sequence number records the offset within the byte stream
- Each QUIC packet contains one or more frames
  - STREAM frames carry data
  - Each has a stream identifier, and carries the length of the data and its offset from start of stream – closer to TCP sequence number



# QUIC Packets and Sequence Numbers (2/2)

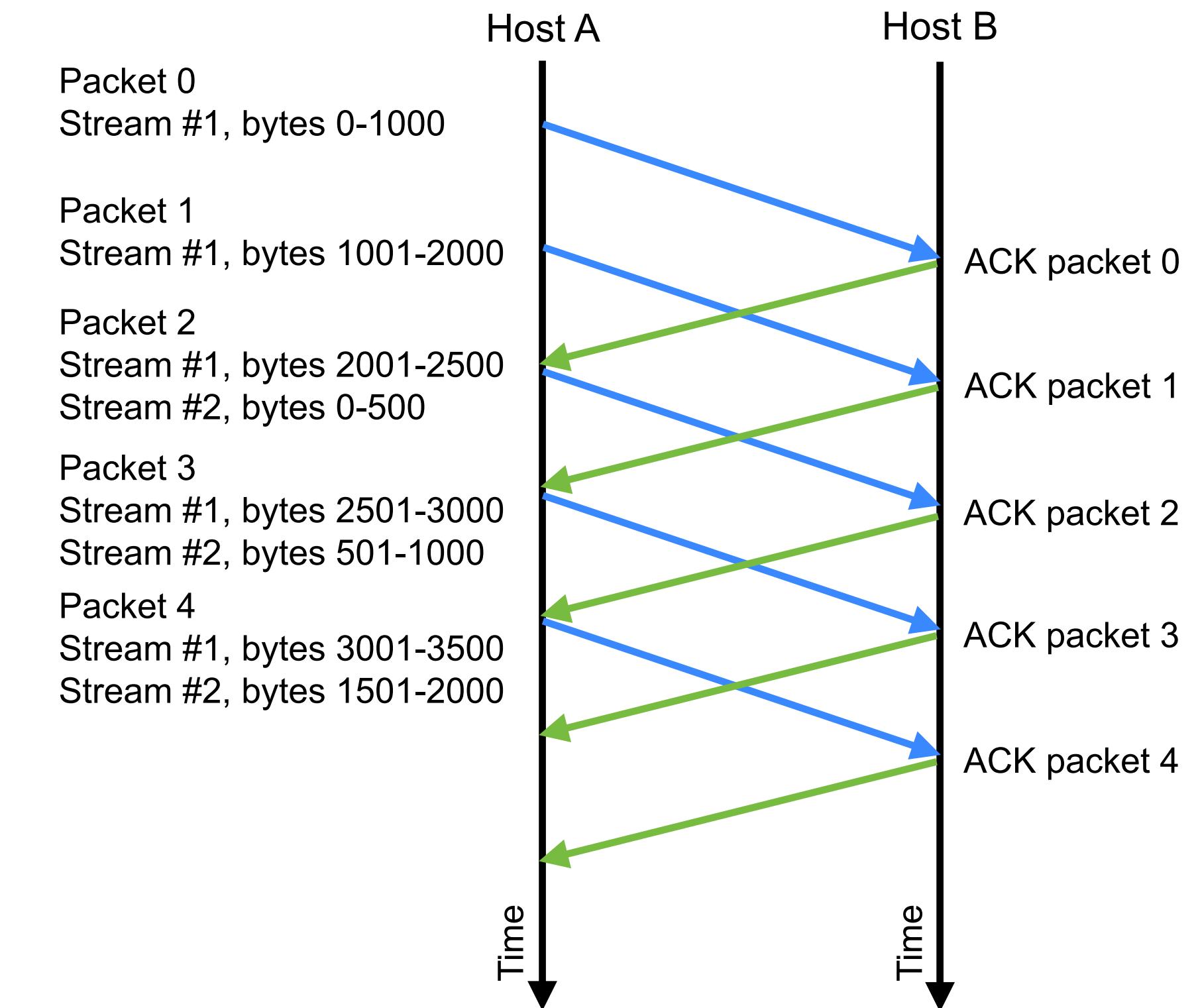
- QUIC doesn't preserve message boundaries in streams
  - If data written to stream is too big for a packet, it will be split across multiple packets
  - If data written to stream is too small for a packet, it may be delayed and sent with other data to fill complete packet
  - Data from more than one stream might be sent in a single packet
  - If >1 stream has data to send, the QUIC sender can choose how to prioritise and deliver frames from each stream



QUIC connection progress, showing packet numbers, stream data

# QUIC Acknowledgements (1/2)

- A QUIC receiver sends acknowledgements for **packets** it receives
  - Unlike TCP, it **does not** acknowledge sequence numbers within each stream
  - The sender needs to remember what data from each stream was in each packet, to know what parts of each stream have been received



QUIC connection progress, showing packet numbers, stream data, and acknowledgements

# QUIC Acknowledgements (2/2)

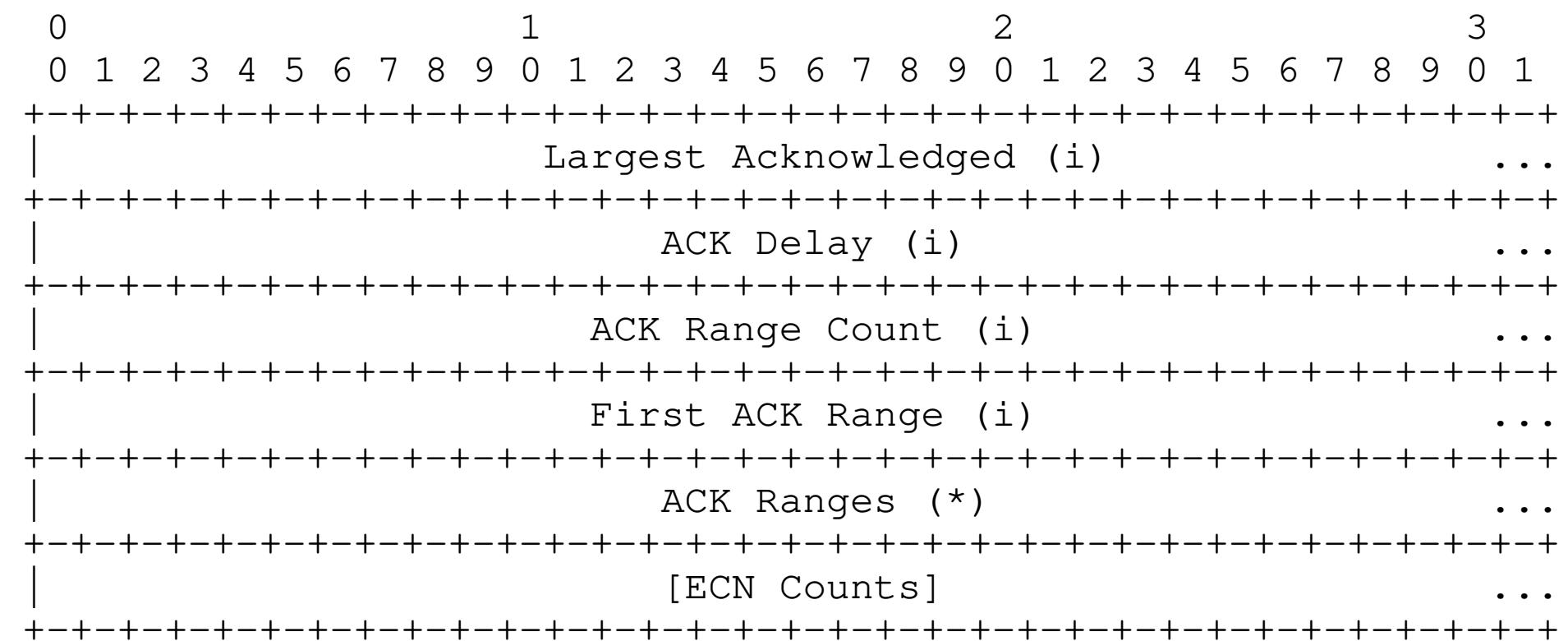
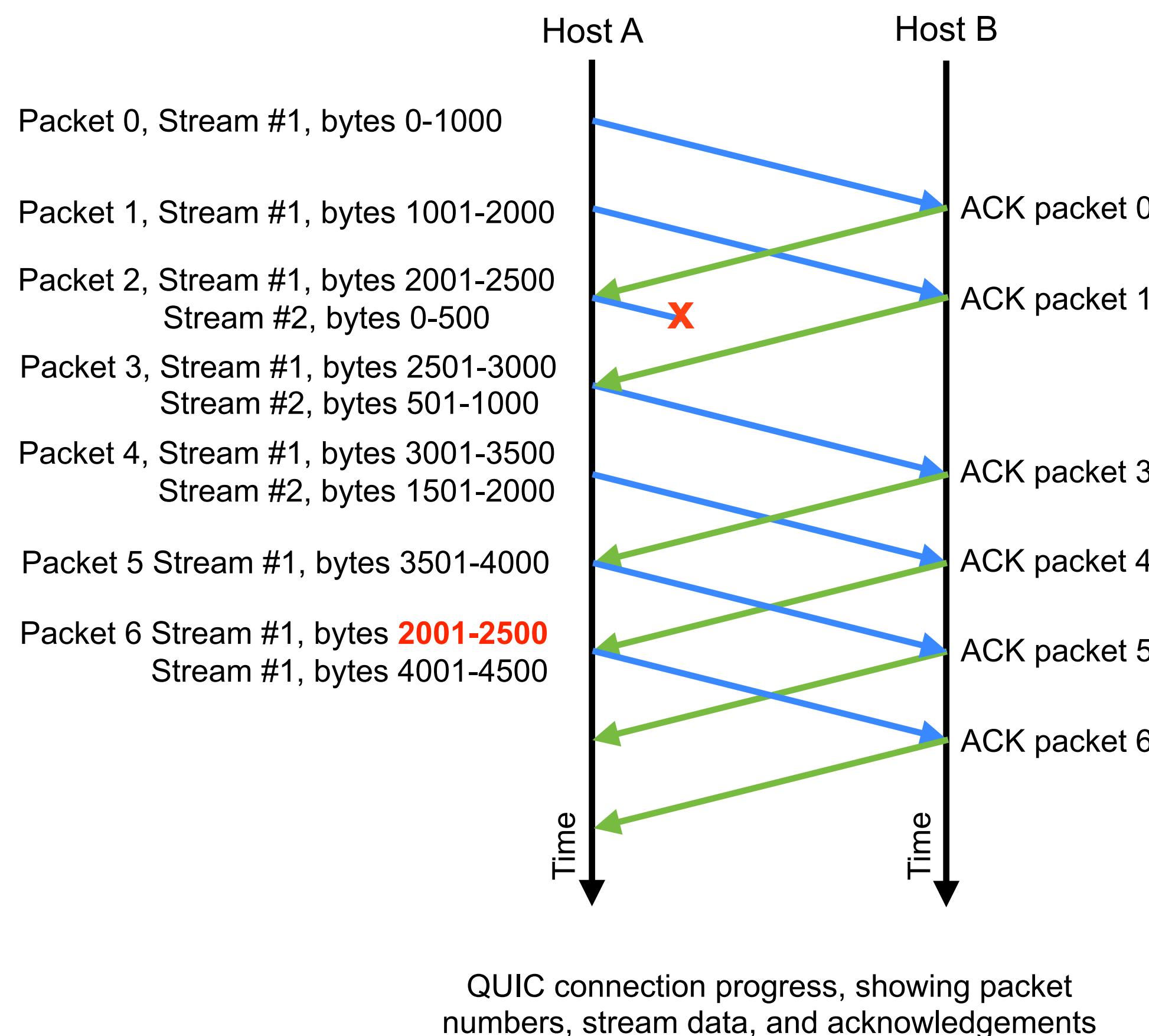


Figure 19: ACK Frame Format

- Acknowledgements sent as reverse path frames
- Acknowledgements can be delayed
  - e.g., send acknowledgement after every second packet is received
- The **ACK Delay** field measures this delay to allow accurate network RTT calculation
- Acknowledgements contain ranges
  - e.g., a single ACK Frame can indicate that packets 1-4 and 6-8 were received
  - Like TCP Selective Acknowledgement extension

# QUIC Retransmissions and Loss Recovery



- QUIC does **not** retransmit lost packets
  - Every packet has a unique packet number and is only transmitted once
  - Packets declared lost when at least three later packets have been acknowledged
    - Equivalent to TCP triple duplicate ACK
  - Packets declared lost after timeout – like TCP
- QUIC retransmits lost data in new packets
  - Packetisation of STREAM frames into packets may differ when data retransmitted
  - Data ordering between streams not preserved

# Head-of-Line Blocking in QUIC

- QUIC can deliver several streams within a single connection
  - Data for each stream is delivered reliably and in-order
    - Whether packet loss affects one or more streams depends on how the sender chooses to distribute stream data between packets
      - Each QUIC packet can contain data from one stream, alternating between streams
      - Each QUIC packet can contain data from each stream
      - The specification places no requirements on how streams are split across packets
    - Depending on loss patterns, each stream can independently suffer head-of-line blocking
      - Affects performance **of that stream** in the same way head-of-line blocking affects TCP
  - Order is not preserved between streams within a QUIC connection
    - One stream might be blocked waiting for a retransmission, while other streams continue to deliver data without loss
    - Data from different streams is sent and received independently → careful use of streams can limit duration of head-of-line blocking

# Reliable Data Transfer with QUIC

- QUIC delivers **several** ordered reliable byte streams within a single connection
  - Can be treated like several parallel TCP connections → instead of opening several TCP connections to a server, open one QUIC connection and send multiple streams within it
  - Can be treated as a framing device → each stream represents a single object, with end-of-stream signalling the object is complete
  - Order is not preserved between streams within a QUIC connection
- Best practices for use of multiple streams are still evolving

# Summary

- Internet → best effort packet delivery
- Data transfer in Internet transports
  - UDP – unreliable but timely; substrate
  - TCP – reliable, ordered, stream
  - QUIC – many reliable, ordered, streams
- Different services for different needs



University  
ofGlasgow

# Lowering Latency

Networked Systems (H)  
Lecture 6



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Lecture Outline

- Key factor limiting performance is **latency**
  - Connection establishment latency → **previous lectures**
    - Reduce number of RTTs to improve connection setup time
    - Reduce number of connections needed
    - TCP + TLS 1.3 → QUIC
      - Overlapping connection establishment and security handshake
      - 0-RTT connection re-establishment with idempotent data
      - Stream multiplexing within a single connection
  - Data transfer latency
    - Impact of TCP congestion control
    - Impact of delay-based congestion control
    - Impact of explicit congestion notification
    - Impact of the speed of light

# TCP Congestion Control

# TCP Congestion Control

- Congestion control principles
- Loss-based congestion control
  - How to effectively make use of network capacity
    - TCP Reno
    - TCP Cubic
- Delay-based congestion control
  - Reducing TCP-induced latency
    - TCP Vegas
    - TCP BBR
- Explicit congestion notification
- TCP is a complex, **highly optimised**, protocol
  - This lecture is a **very** simplified review of complex issues

Internet Engineering Task Force (IETF)  
Request for Comments: 7414  
Obsoletes: 4614  
Category: Informational  
ISSN: 2070-1721

M. Duke F5  
R. Braden ISI  
W. Eddy MTI Systems  
E. Blanton Interrupt Sciences  
A. Zimmermann NetApp, Inc.  
February 2015

A Roadmap for Transmission Control Protocol (TCP) Specification Documents

**Abstract**

This document contains a roadmap to the Request for Comments (RFC) documents relating to the Internet's Transmission Control Protocol (TCP). This roadmap provides a brief summary of the documents defining TCP and various TCP extensions that have accumulated in the RFC series. This serves as a guide and quick reference for both TCP implementers and other parties who desire information contained in the TCP-related RFCs.

This document obsoletes RFC 4614.

**Status of This Memo**

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7414>.

Duke, et al. Informational [Page 1]

M. Duke, R. Braden, W. Eddy, E. Blanton, and A. Zimmermann,  
“A Roadmap for TCP Specification Documents”, IETF, RFC  
7414, February 2015 (<https://tools.ietf.org/html/rfc7414>)

# QUIC Congestion Control

- What about QUIC?
- The QUIC v1 standard adopts congestion control as used in TCP Reno
- QUIC implementations tend to use Cubic or BBR congestion control in practice



# Congestion Control Principles

Congestion Avoidance and Control

Van Jacobson\*

University of California  
Lawrence Berkeley Laboratory  
Berkeley, CA 94720  
van@helios.ee.lbl.gov

In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and three IMP hops) dropped from 32 Kbps to 40 bps. Mike Karels<sup>1</sup> and I were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. We wondered, in particular, if the 4BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

Since that time, we have put seven new algorithms into the 4BSD TCP:

- (i) round-trip-time variance estimation
- (ii) exponential retransmit timer backoff
- (iii) slow-start
- (iv) more aggressive receiver ack policy
- (v) dynamic window sizing on congestion
- (vi) Karn's clamped retransmit backoff
- (vii) fast retransmit

Our measurements and the reports of beta testers suggest that the final product is fairly good at dealing with congested conditions on the Internet.

\* This work was supported in part by the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

<sup>1</sup>The algorithms and ideas described in this paper were developed in collaboration with Mike Karels of the UC Berkeley Computer System Research Group. The reader should assume that anything clever is due to Mike. Opinions and mistakes are the property of the author.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

1988 ACM 0-89791-279-9/88/008/0314

314

- Key congestion control principles:
  - Packet loss as a congestion signal
  - Conservation of packets in flight
  - Additive increase/multiplicative decrease



Van Jacobson

Source: PARC



Sally Floyd

Source: Sally Floyd

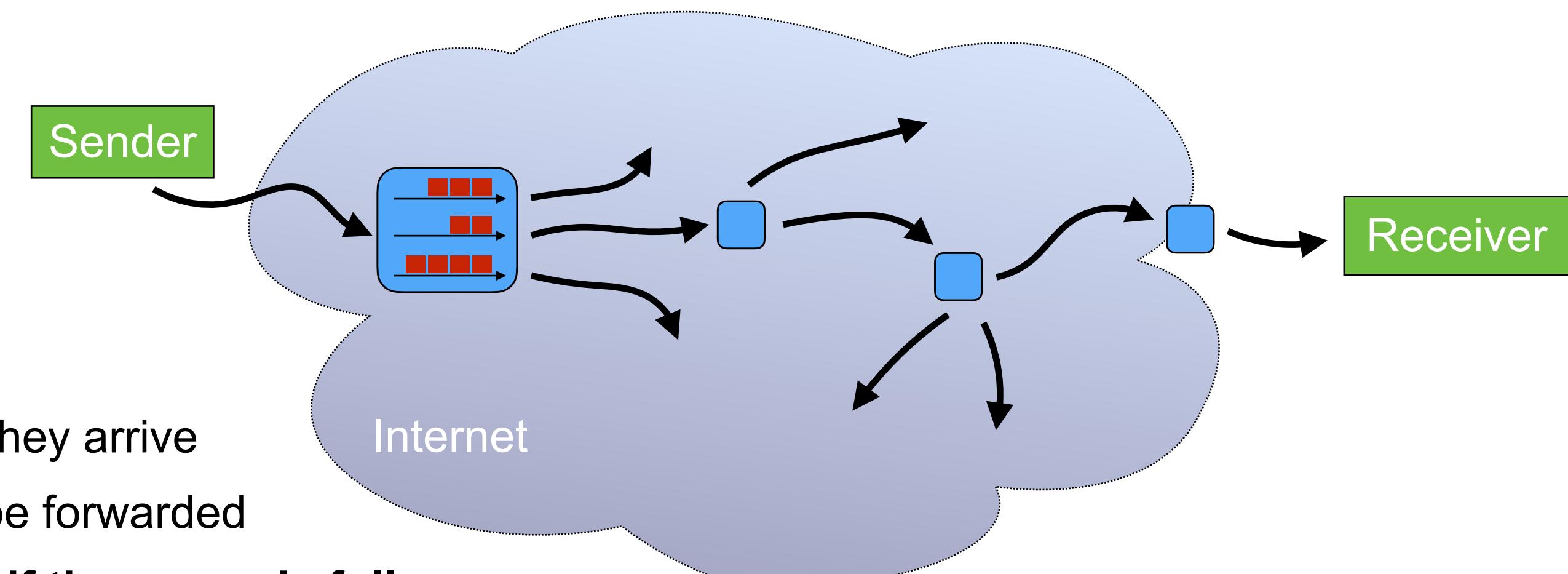
- Practical TCP congestion control:
  - Robust IETF standards for TCP
  - Embodies congestion control principles
  - Extremely high performance

<https://www.nytimes.com/2019/09/04/science/sally-floyd-dead.html>

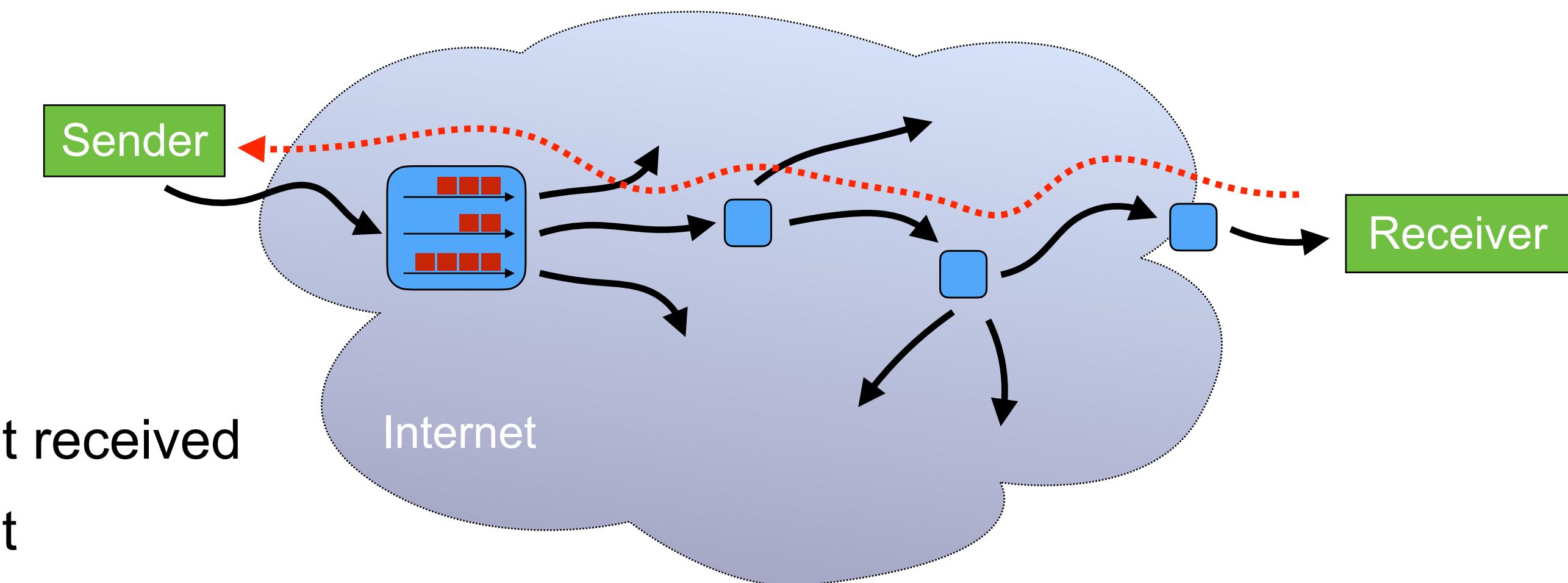
V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM Conference, Stanford, CA, USA, August 1988..  
<http://dx.doi.org/10.1145/52324.52356>

# Key Principles: Packet Loss as a Congestion Signal

- Data flows from sender to receiver through a series of **IP routers**
- Routers perform two functions:
  - **Routing**: receive packets, determine appropriate route to destination
  - **Forwarding**: enqueue packets on outgoing link for delivery
    - Queues shrink if packets are forwarded faster than they arrive
    - Queues grow if packets arrive faster than they can be forwarded
    - Queues have maximum size → **packets discarded if the queue is full**
    - Congestion control can use this packet loss as a congestion signal

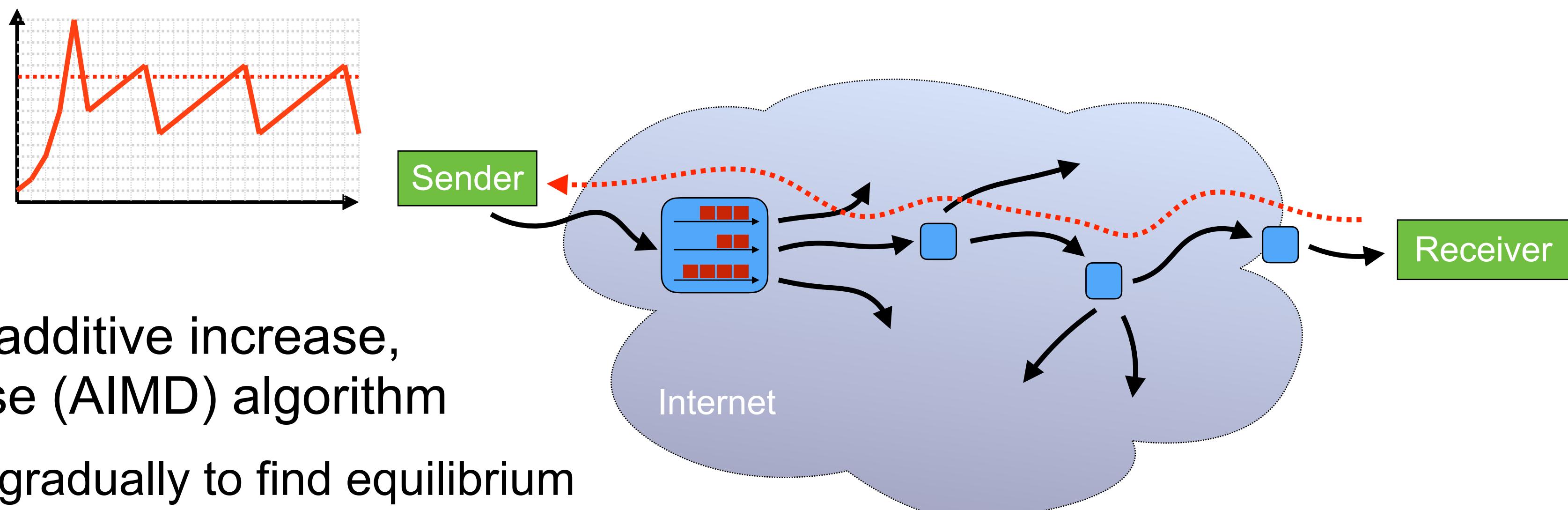


# Key Principles: Conservation of Packets



- Acknowledgements returned by receiver
  - Send one packet for each acknowledgement received
  - Total number of packets in transit is constant
    - System is in equilibrium; queues neither increase nor decrease in size
    - **ACK clocking** – each acknowledgement “clocks out” the next packet
  - Automatically reduces sending rate if network gets congested and delivers packets more slowly

# Key Principles: AIMD Algorithms



- Sending rate follows additive increase, multiplicative decrease (AIMD) algorithm
  - Start slowly, increase gradually to find equilibrium
    - Add a small amount to the sending speed each time interval without loss
    - For a window-based algorithm  $w_i = w_{i-1} + \alpha$  each RTT, where  $\alpha = 1$  typically
  - Respond to congestion rapidly
    - Multiply sending window by some factor  $\beta < 1$  each interval loss seen
    - For a window-based algorithm  $w_i = w_{i-1} \times \beta$  each RTT, where  $\beta = 1/2$  typically
  - Faster reduction than increase → stability; TCP Reno

# TCP Congestion Control

- Packet loss as congestion signal
- Conservation of packets in equilibrium
- AIMD

# TCP Reno

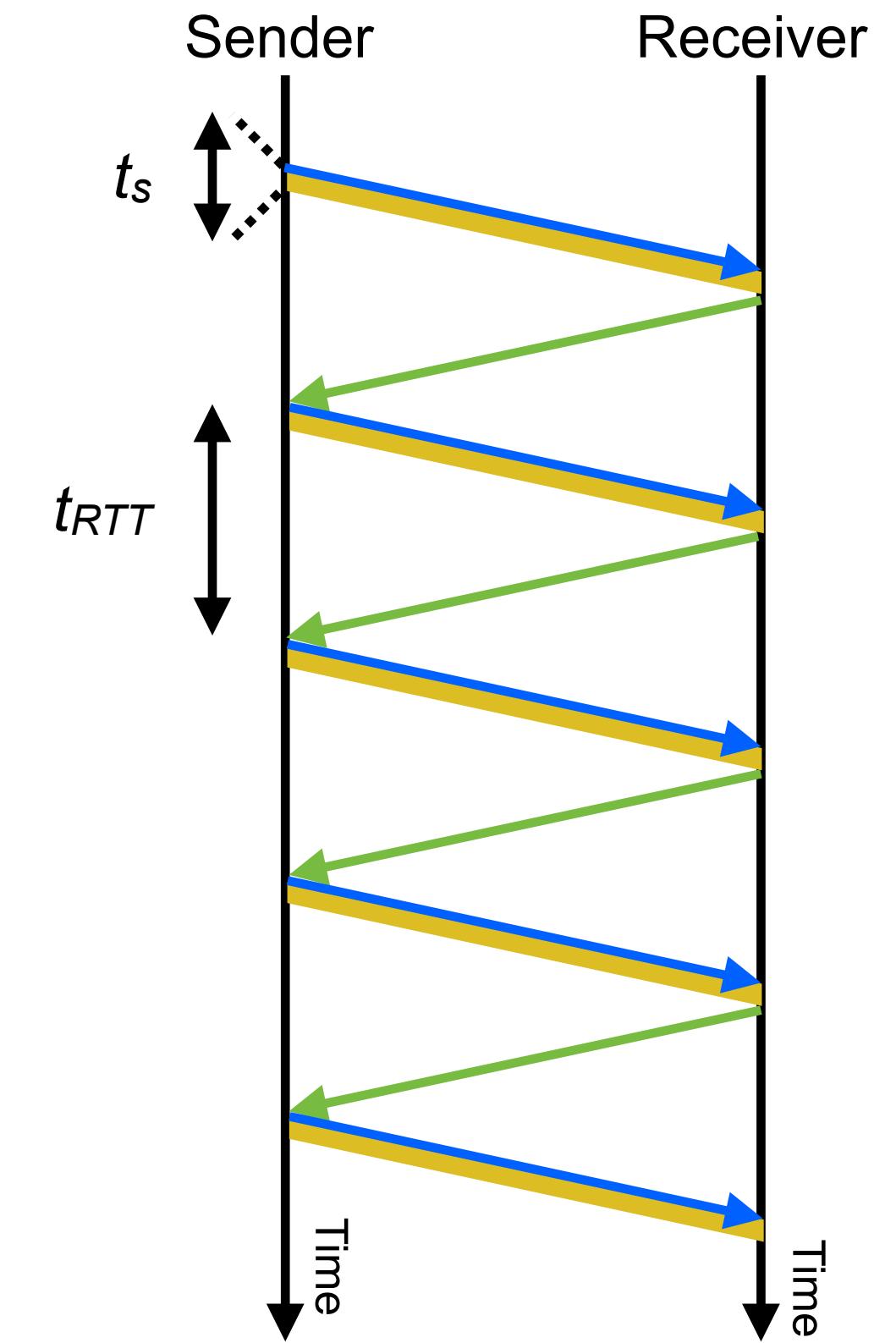
- Basic TCP congestion control
- Sliding window algorithms
- Slow start
- Congestion avoidance

# Congestion Control in TCP Reno

- TCP uses window-based congestion control
  - Maintains a **sliding window** onto the available data that determines how much can be sent according to the AIMD algorithm
  - Plus slow start and congestion avoidance
  - Gives approximately equal share of the bandwidth to each flow sharing a link
- Basic congestion control algorithm known as **TCP Reno**
  - The state of the art in TCP as of ~1990

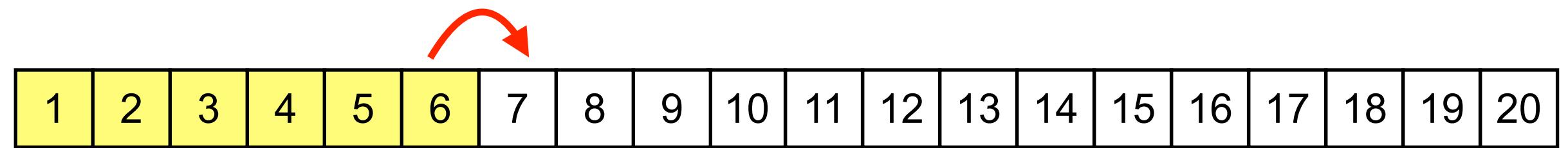
# From Stop-and-Wait to Sliding Window Protocols

- Stop-and wait protocols perform poorly
  - It takes time,  $t_s$ , to send a packet
    - $t_s = (\text{packet size}) / (\text{link bandwidth})$
  - Acknowledgement returns  $t_{RTT}$  seconds later
  - Link utilisation,  $U = t_s / t_{RTT}$ 
    - Fraction of time link is sending packets → want  $U \approx 1.0$
    - Assume a gigabit link sending a 1500 byte packet from Glasgow to London:
      - $t_s = 1500 \times 8 \text{ bits} / 10^9 \text{ bits per second} = 0.000012\text{s}$
      - $t_{RTT} \approx 0.010 \text{ seconds}$
      - $U \approx 0.0012$
      - i.e., the link is in use 0.12% of the time
  - Sliding window protocols improve on stop-and-wait by sending more than one packet before stopping for acknowledgement



# Sliding Window Protocols Improve Link Utilisation

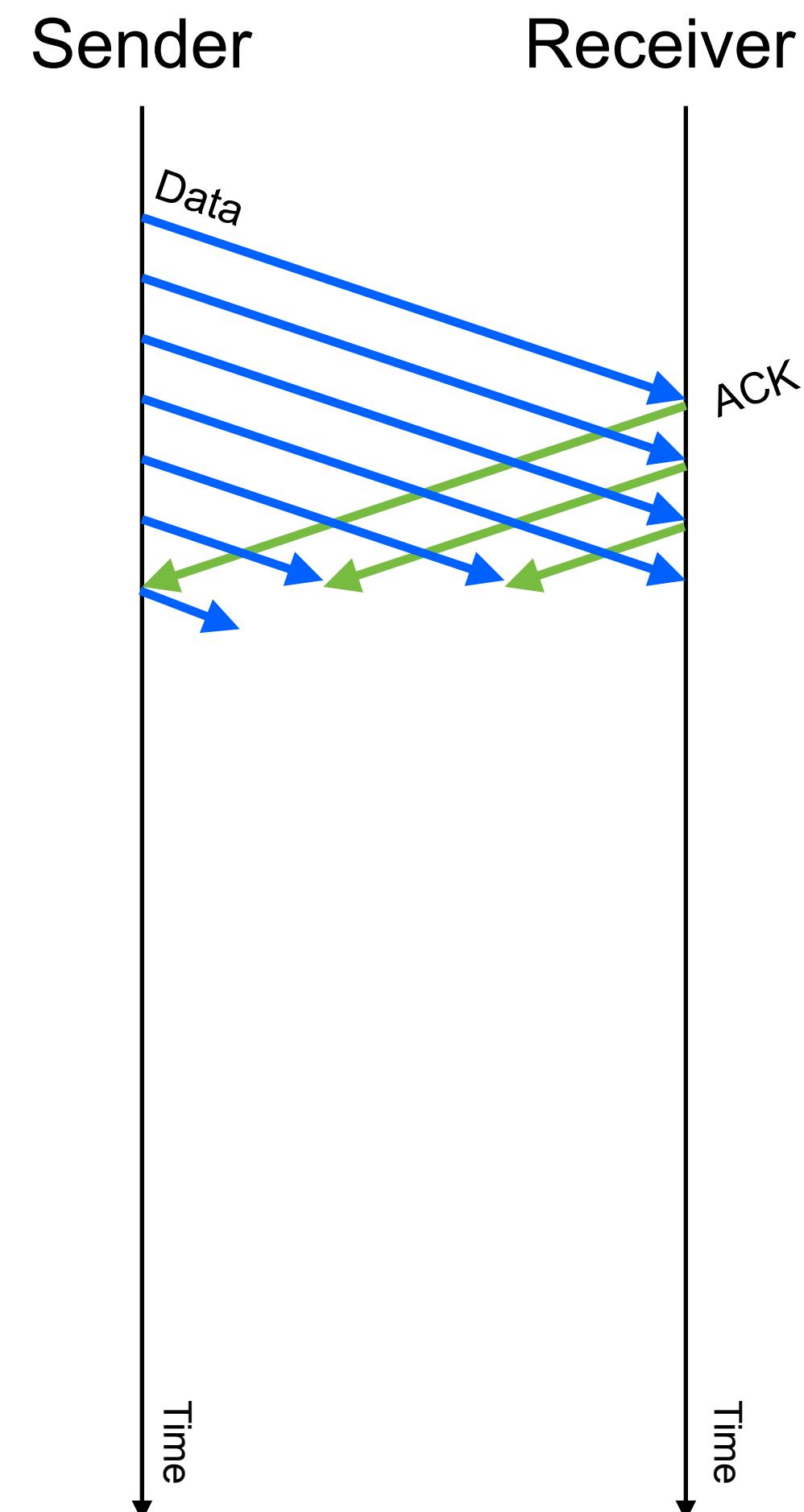
Sliding window protocols improve link utilisation using a **congestion window** – number of packets to be sent before an acknowledgement arrives



In this example, the window size is six packets → acknowledgement for packet 6 arrives just in time to release packet 7 for transmission

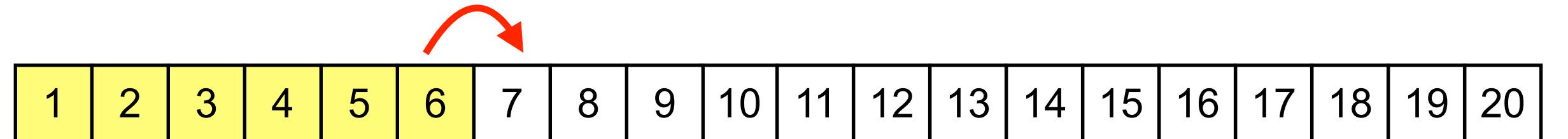
Each returning acknowledgement for new data slides the window along, releasing next packet for transmission → if window sized correctly, each acknowledgement **arrives just in time** to release next packet

What is the optimal size for the window? **bandwidth × delay of path** → **but neither is known to the sender**

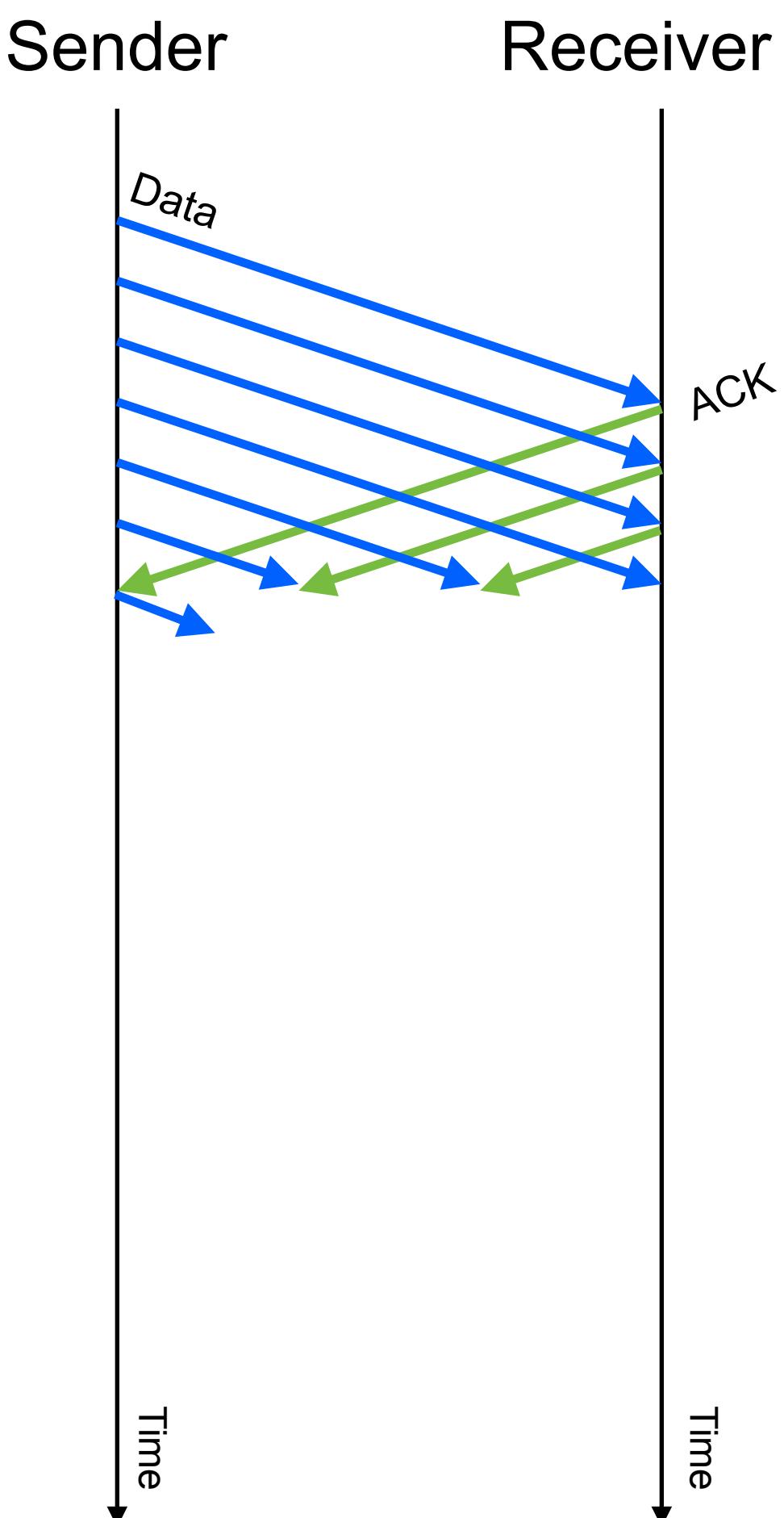


# TCP Reno Congestion Control

- **TCP Reno is a sliding window protocol**
  - Optimised for not having information to know the correct window size



- How to choose initial window?
- How to find the path capacity?
  - Slow start to estimate the bottleneck link capacity
  - Congestion avoidance to probe for changes in capacity



# TCP Reno: Choosing the Initial Window

- How to choose initial window size,  $W_{init}$ ?
  - No information → need to measure path capacity
  - Start with a small window, increase until congestion
  - **$W_{init} = 1$  packet per round-trip time (RTT) is safe**
    - Start at the slowest possible rate, equivalent to stop-and-wait, and increase
  - **$W_{init} = 3$  packets per RTT**
    - Traditional TCP Reno approach
  - **$W_{init} = 10$  packets per RTT**
    - Modern TCP implementations [RFC 6928]
      - Compromise between safety and performance – measurements show this is generally safe at present
      - Expect  $W_{init}$  to gradually be increased as average network performance improves

**An Argument for Increasing TCP's Initial Congestion Window**

Nandita Dukkipati, Tiziana Rerice, Yuchung Cheng, Jerry Chu  
Tom Herbert, Amit Agarwal, Arvind Jain and Natalia Sutin  
Mountain View, CA, USA  
(nanditad, tiziana, ycheng, thcherbert, aagarwal, arvind, nsutin@google.com)

**ABSTRACT**  
TCP flows start with an initial congestion window of at most four segments or approximately 4KB of data. Because most Web pages are much smaller than 4KB, the slow start phase is a critical TCP parameter in determining how quickly flows can finish. While the global network access speeds increased dramatically on average in the past decade, the standard value of TCP's initial congestion window has remained constant. In this paper, we propose to increase TCP's initial congestion window to at least ten segments (about 15KB). Through large-scale Internet experiments, we quantify the latency benefits and costs of using a larger window, as functions of network bandwidth, round-trip time (RTT), bottleneck delay product (BDP), and nature of application. We show that the average latency of HTTP responses improved by approximately 10% with the largest benefits being demonstrated in high RTT and BDP networks. The latency of low bandwidth networks also improved by a significant amount in our experiments. The average retransmission rate increased by a modest 0.5%, with most of the increase coming from applications that effectively circumvent TCP's slow start algorithm. Bottleneck contention was reduced. Based on the results from our experiments, we believe the initial congestion window should be *at least* ten segments and the same was investigated for *at least* one segment. We argue that the initial congestion window should be increased to cover the congestion avoidance phase. The initial congestion window is at most four segments, but more typically is three segments (approximately 4KB) [5] for standard Ethernet MTU and minimum round-trip time (RTT) of Web pages. In fact, and this is before exiting the slow start phase, making TCP's initial congestion window ( $init\_wnd$ ) a crucial parameter in determining flow completion time. Our premise is that the initial congestion window should be increased to speed up short Web transactions while maintaining robustness.

While the global adoption of broadband is growing, TCP's  $init\_wnd$  has remained unchanged since 2002. As per a 2009 study [4], the average connection bandwidth globally is 1.7Mbps. The number of clients have increased with above 2Mbps, while the usage of narrowband (<250Kbps) has shrunk to about 5% of clients. At the same time, applications devised their own mechanisms for faster download of Web pages. Popular Web browsers, including IES [2], Firefox 3 and Opera 10, open up to 10 segments per connection, partly to increase efficiency and avoid head-of-line blocking of independent HTTP requests/responses, but mostly to boost start-up performance when downloading a Web page.

In light of these trends, allowing TCP to start with a higher  $init\_wnd$  offers the following advantages:

(1) *Reduce latency*. Latency of a transfer completing in slow start without losses [8], is

$$[log_2(\frac{S(\gamma - 1)}{init\_wnd} + 1)] * RTT + \frac{S}{C} \quad (1)$$

where  $S$  is transfer size,  $C$  is bottleneck link-rate,  $\gamma = 1.5$  or 2 depending on the acknowledgement policy, not defined in eqn 1 and  $init\_wnd > 1$ . As link speeds scale up, TCP's latency is dominated by the number of round-trip times (RTT) in the slow start phase. Increasing  $init\_wnd$  enables transfers to finish in fewer RTTs.

(2) *Keep latency growing in Web page sizes*. The Internet is growing. Web page size is K [14] including HTTP headers and compressed resources. An average sized page requires multiple RTTs to download when using a single TCP connection with a small  $init\_wnd$ . To improve page

ACM SIGCOMM Computer Communication Review  
Volume 40, Number 3, July 2010

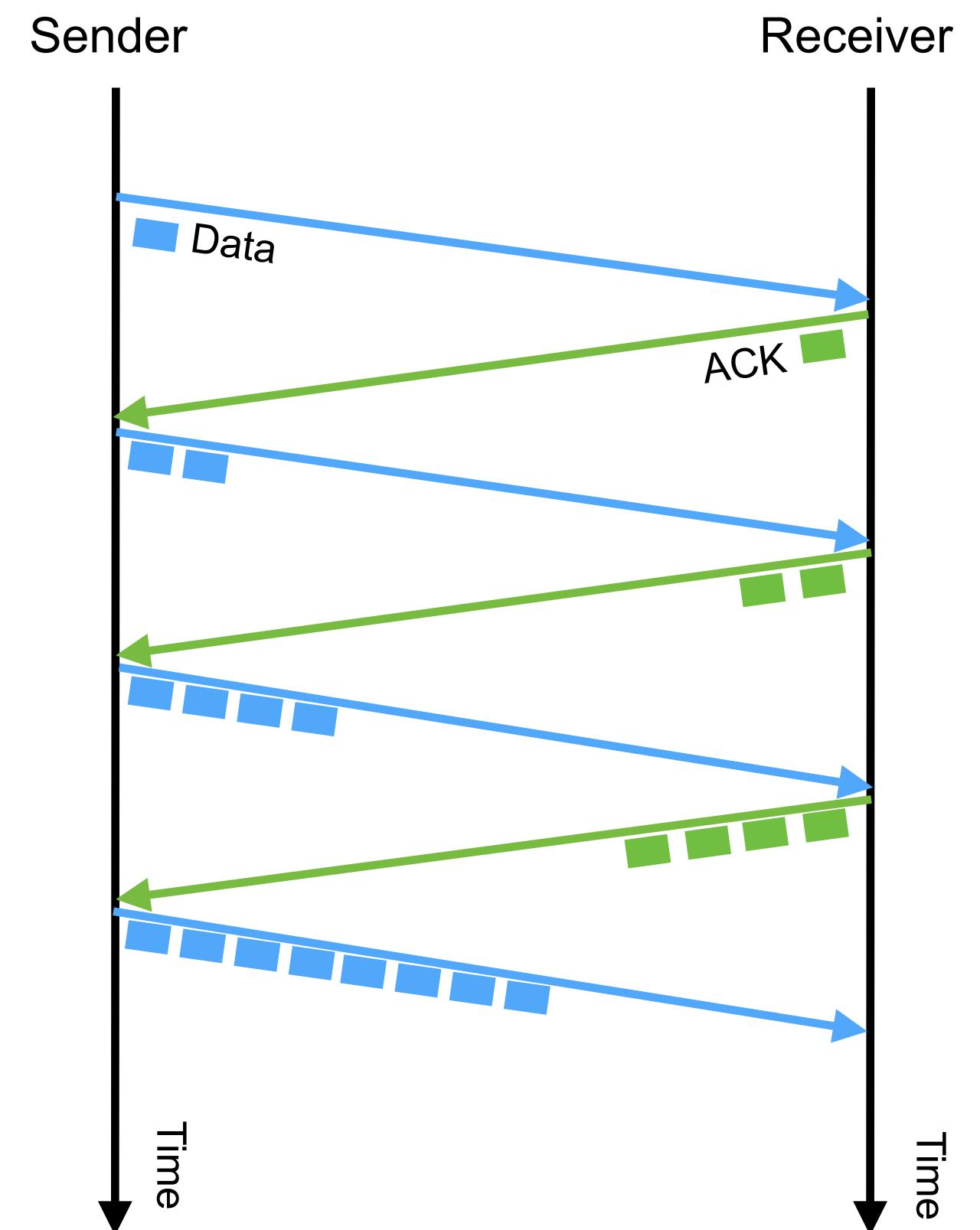
N. Dukkipati et al, “An argument for increasing TCP’s initial congestion window”, ACM Computer Communication Review, 40(3):27–33, July 2010.  
<http://dx.doi.org/10.1145/1823844.1823848>

# TCP Reno: Finding the Path Capacity

- The initial window allows you to send **something**
  - Unlikely to be the optimal window size
  - How to choose the correct window size to match the link capacity?
    - **Slow start** to rapidly find the correct window size for the path
    - **Congestion avoidance** to adapt to changes in path capacity once connection is running

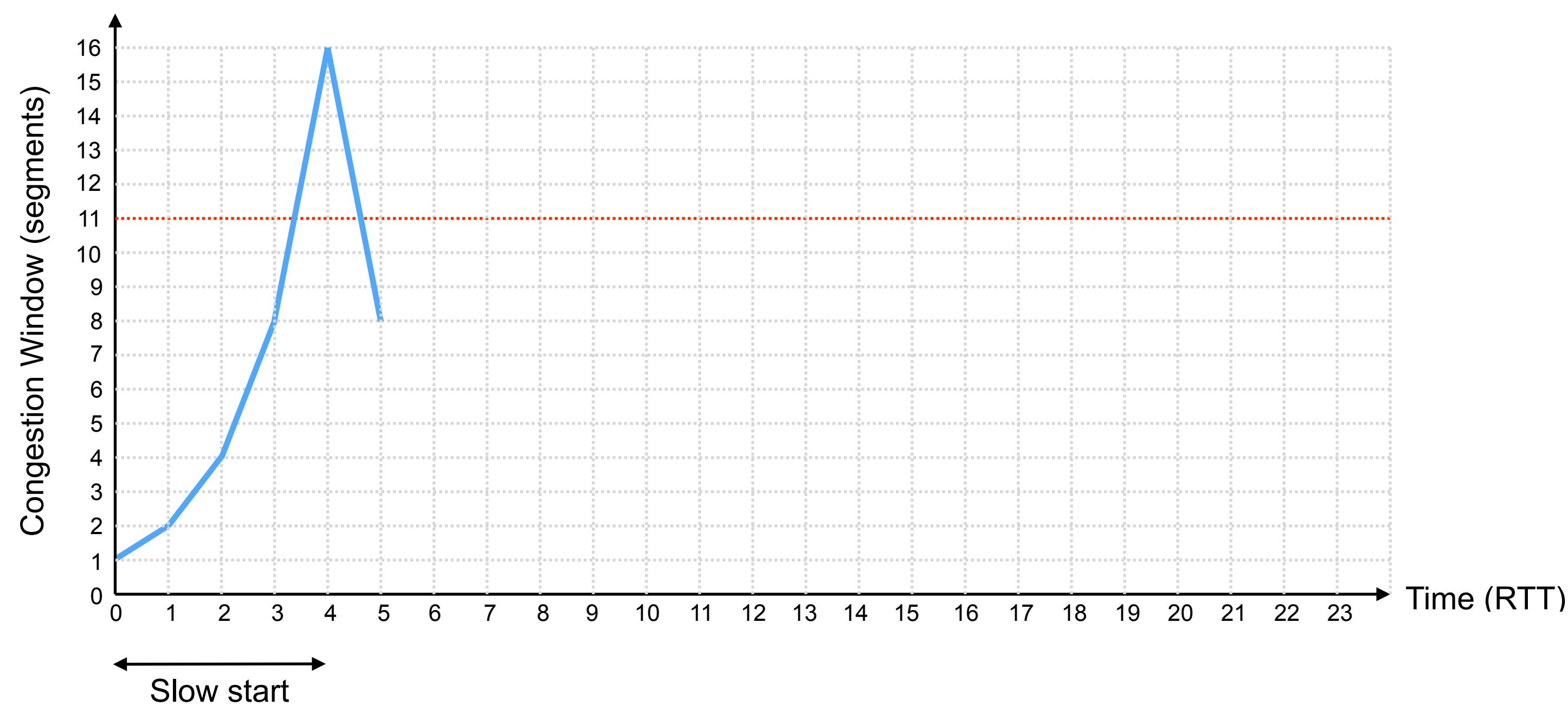
# TCP Reno: Slow Start (1/2)

- Assume  $W_{\text{init}} = 1$  packet per RTT – **slow start to connection**
  - It will be  $W_{\text{init}} = 3$  or  $W_{\text{init}} = 10$  in practice, but using  $W_{\text{init}} = 1$  makes the example simpler...
- Rapidly increase window until network capacity is reached
  - Each acknowledgement for new data increases congestion window,  $W$ , by 1 packet per RTT → congestion window doubles each RTT
  - If a packet is lost, halve congestion window back to previous value and exit slow start



# TCP Reno: Slow Start (2/2)

- TCP Reno slow start phase:
  - Starts sending slowly
  - Rapidly increases sending rate until a packet is lost
  - Resets sending rate to last known good rate when first loss occurs

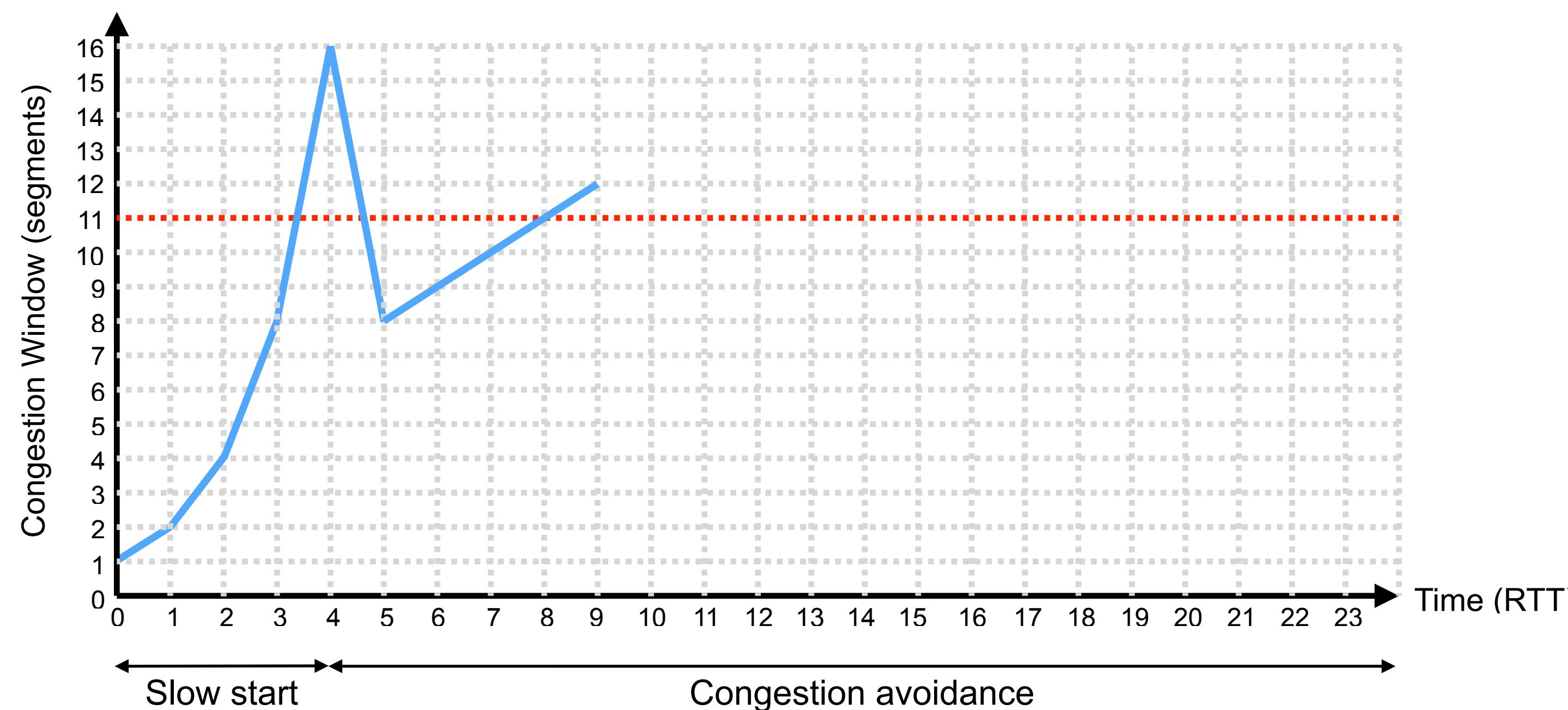


# TCP Reno: Congestion Avoidance (1/4)

- After first packet is lost, TCP switches to **congestion avoidance**
  - The congestion window is now approximately the right size for the path
  - Goal is now to adapt to changes in network capacity
    - Perhaps the path capacity changes – radio signal strength changes for mobile device
    - Perhaps the other traffic changes – competing flows stop; additional cross-traffic starts
  - Additive increase, multiplicative decrease of congestion window

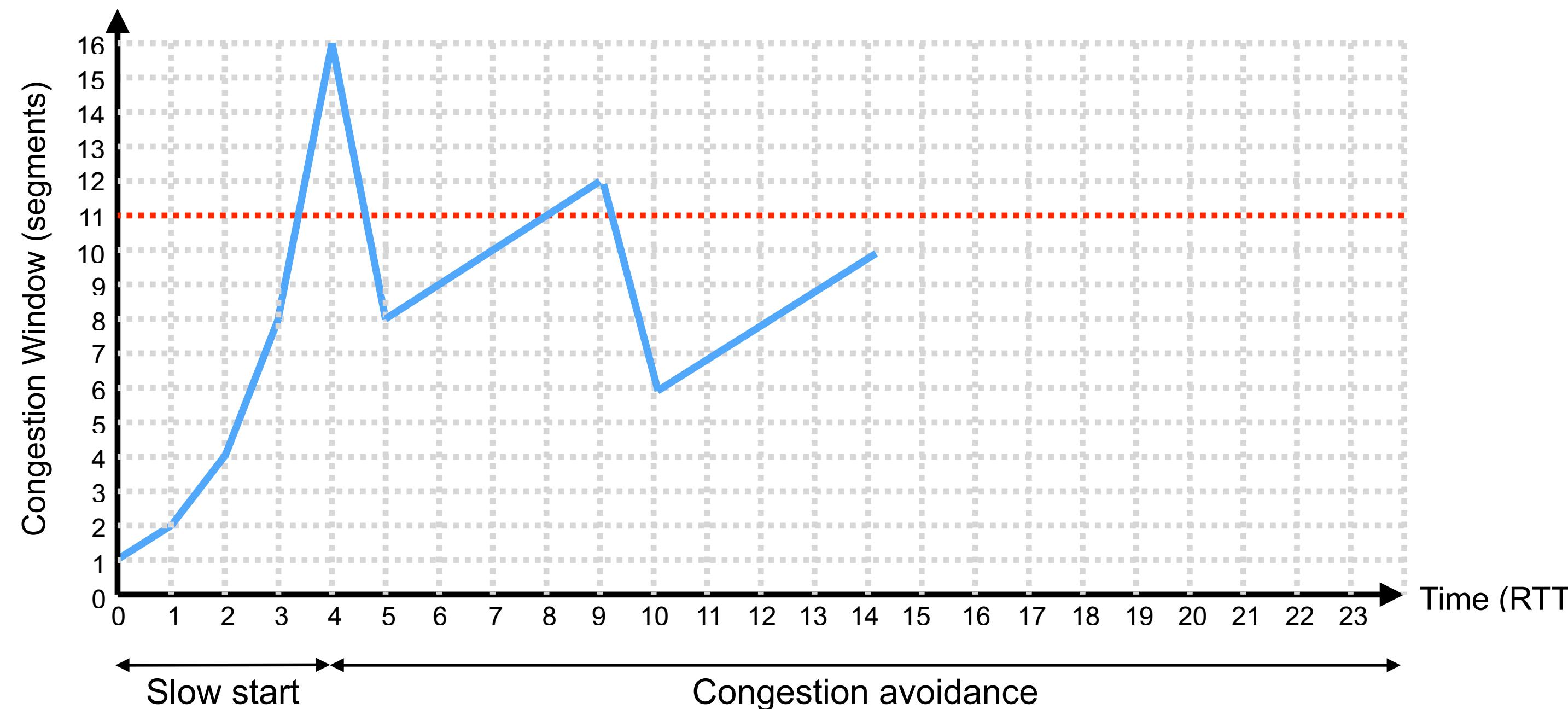
# TCP Reno: Congestion Avoidance (2/4)

- If a complete window of packets is sent without loss:
  - Increase congestion window by 1 packet per RTT, then send next window worth of packets
  - Slow, linear, additive increase in window:  $W_i = W_{i-1} + 1$



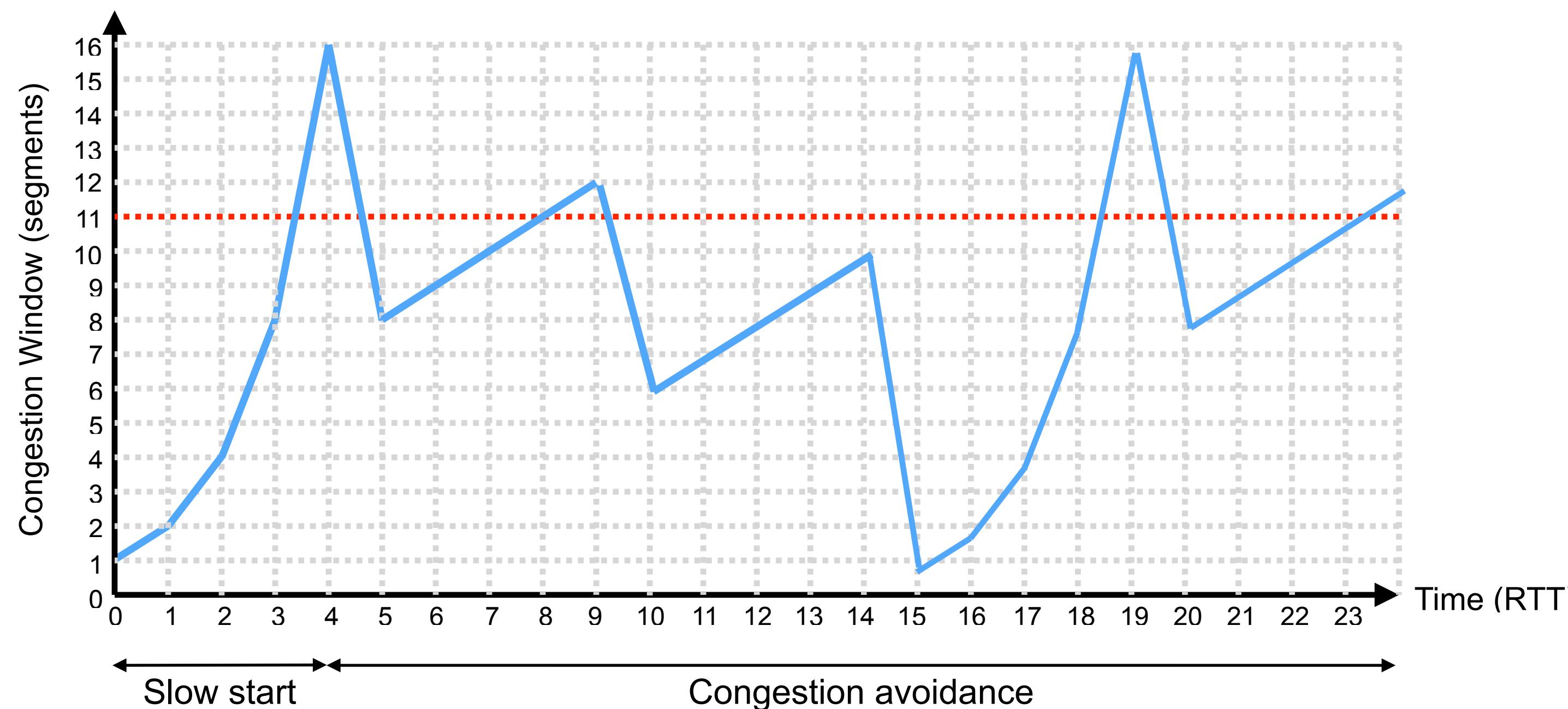
# TCP Reno: Congestion Avoidance (3/4)

- If a packet is lost and detected via triple duplicate acknowledgement:
  - Transient congestion, but data still being received
  - Multiplicative decrease in window:  $W_i = W_{i-1} \times 0.5$
  - Rapid reduction in window allows congestion to clear quickly, avoids congestion collapse
- Then, return to additive increase until next loss

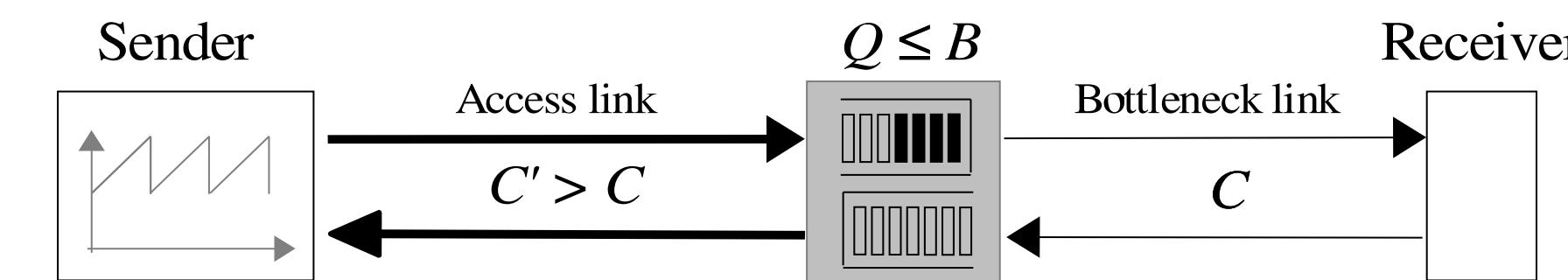
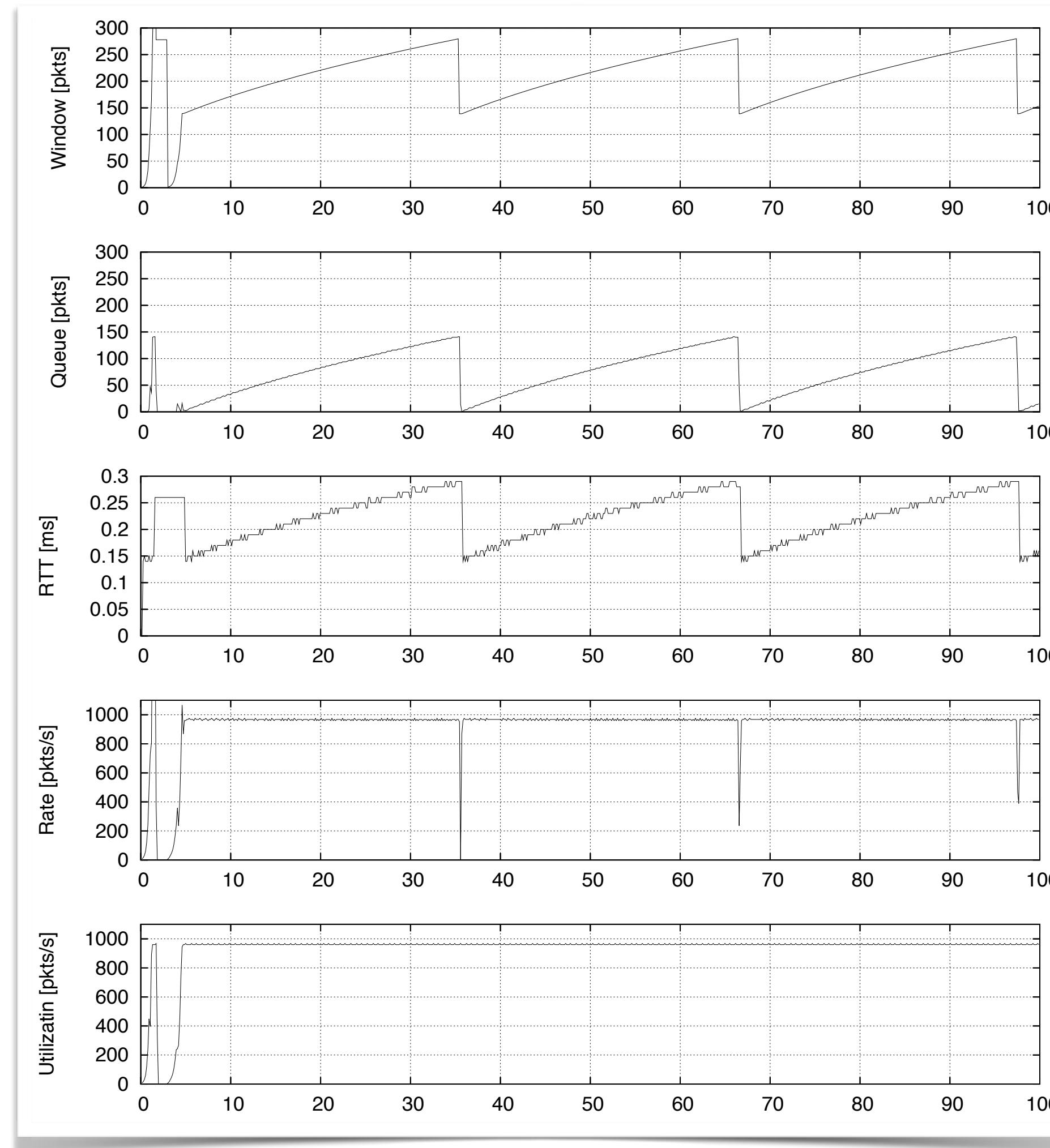


# TCP Reno: Congestion Avoidance (4/4)

- If a packet is lost and detected via timeout:
  - Either receiver or path has failed – reset to  $W_{init}$  and re-enter slow start
  - How long is the timeout?
    - $T_{rto} = \max(1 \text{ second}, \text{average RTT} + (4 \times \text{RTT variance}))$



# Congestion Window Growth, Buffering, Throughput



Bottleneck buffer size = bandwidth  $\times$  delay

- Bottleneck queue never empty
- Bottleneck link never becomes idle  $\rightarrow$  sending rate varies, but receiver sees continuous flow
- Congestion window follows a “sawtooth” pattern, but received rate is constant at approximately bottleneck bandwidth

Source: G. Appenzeller, “Sizing Router Buffers”, PhD thesis, Stanford University, March 2005.  
<http://tiny-tera.stanford.edu/~nickm/papers/guido-thesis.pdf> (Figures 2.1 and 2.2)

# TCP Reno: Discussion

- TCP Reno is effective at keeping bottleneck link fully utilised
  - Trades some extra delay to maintain throughput
  - Provided sufficient buffering in the network: buffer size = bandwidth × delay
  - Packets queued in buffer → delay
- Limitations:
  - Assumes packet loss is due to congestion; non-congestive loss, e.g., due to wireless interference, impacts throughput
  - Congestion avoidance phase takes a long time to use increased capacity

# TCP Reno

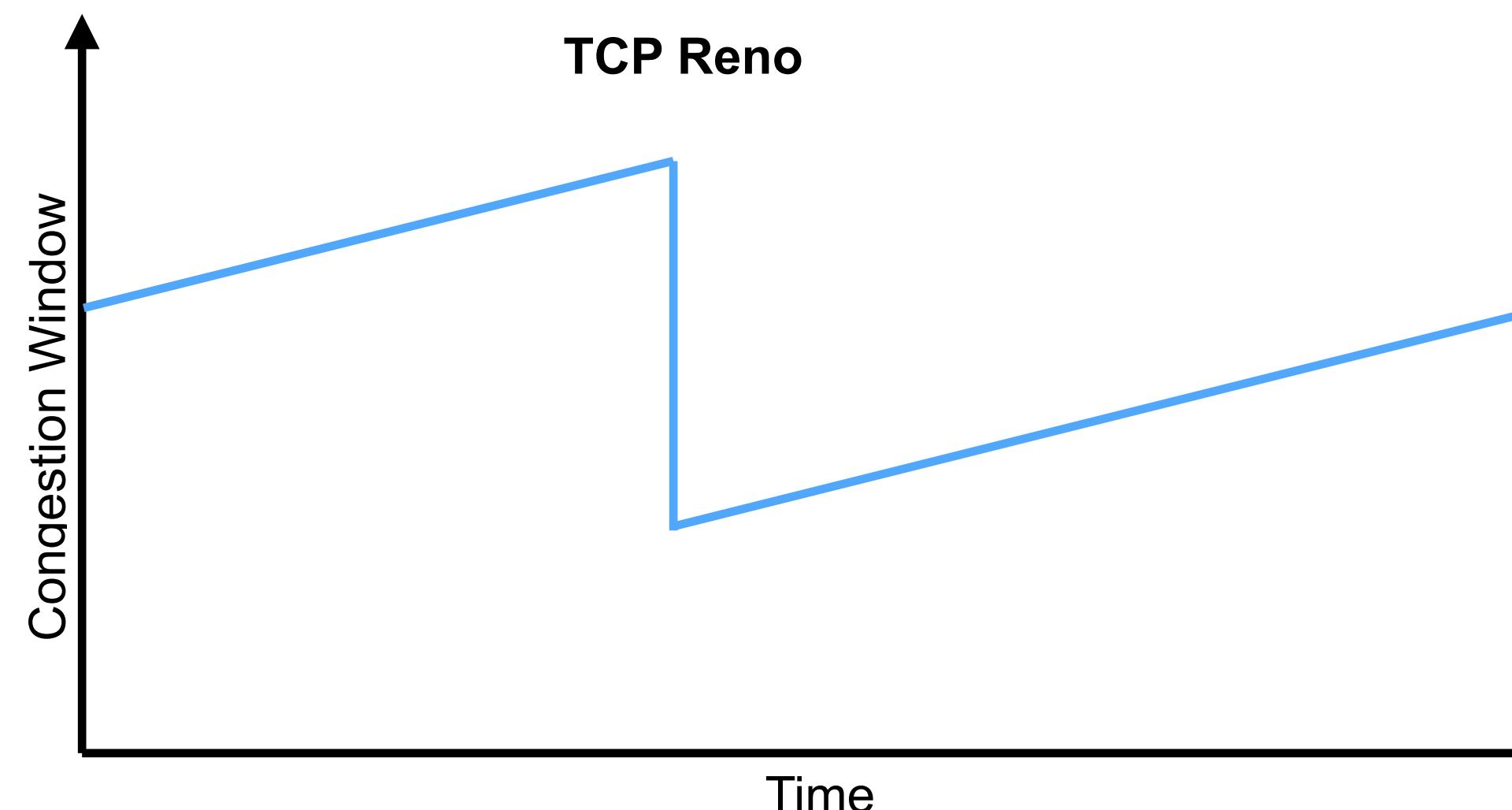
- Basic TCP congestion control
- Sliding window algorithms
- Slow start
- Congestion avoidance

# TCP Cubic

- Improving TCP performance on fast, long-distance, networks

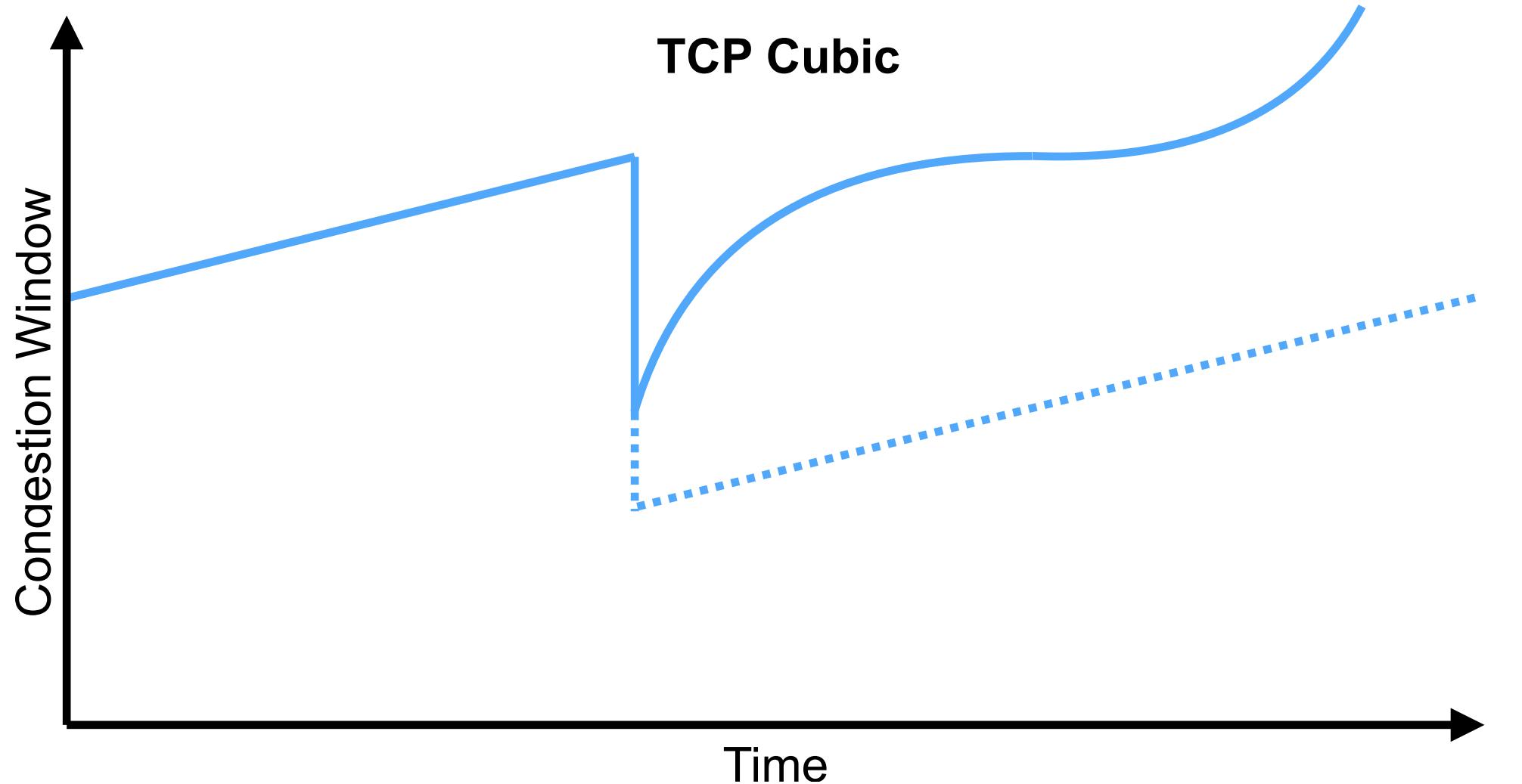
# TCP Performance on Fast Long-distance Networks

- TCP Reno can perform poorly on **fast long-distance networks**
  - e.g., multi-gigabit per second inter-continental links
  - Path with 10Gbps bandwidth and 100ms RTT requires congestion window ~100,000 packets
  - In congestion avoidance, one packet lost and detected via triple duplicate ACK halves the window, then increase by 1 packet per RTT → 50,000 RTTs to recover sending rate
  - Approximately 1.4 hours with 100ms RTT!



# TCP Cubic

- **TCP Cubic** changes the congestion control algorithm
  - During congestion avoidance, increases congestion window faster than TCP Reno on fast long-distance networks
  - Rapidly increases congestion window after packet loss
  - Slows rate of increase as window approaches the largest successfully achieved window



S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating System Review, Vol. 54, Num. 5, pages 64-74, July 2008  
<https://dx.doi.org/10.1145/1400097.1400105>

I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", IETF, February 2018, RFC 8312.  
<https://datatracker.ietf.org/doc/rfc8312/>

**CUBIC: A New TCP-Friendly High-Speed TCP Variant**

Sangtae Ha, Injong Rhee  
Dept of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
(sha2,rhee)@ncsu.edu

Lisong Xu  
Dept of Comp. Sci. and Eng.  
University of Nebraska  
Lincoln, Nebraska 68588  
xu@cse.unl.edu

**ABSTRACT**  
"high-speed" TCP variants are proposed (e.g., FAST [24], HSTCP [15], SCTP [22], HTTP [28], SQRT [19], Westwood [14], and BIC-TCP [20]). In response to them, with TCP, the Linux community responded quickly to implement a majority of these protocols in Linux and ship them as part of its operating system. After a series of third-party testing and performance validation [11, 21], in 2004, from version 2.6.8, it selected BIC-TCP as the default TCP algorithm and the other TCP variants as options.

What makes BIC-TCP stand out from other TCP algorithms is its stability. It uses a binary search algorithm where the window grows to the mid-point between the last window size (i.e., max) where TCP has a packet loss and the last window size (i.e., min) it does not have a loss for one RTT period. This is called "mid-point increment". It is far from the saturation point and the slowly when it is close to the saturation point. This feature allows CUBIC to be very scalable when the bandwidth and delay product of the network is large and, in other words, the size of the congestion window. In standard TCP, TCP-NEWRENO and TCP-SACK, TCP grows its window one per round trip time (RTT). This makes the data transport speed of TCP<sup>\*</sup> used in all major operating systems including Windows and Linux, to be very slow, especially underutilizing the networks especially if the length of flows is much shorter than the time. TCP grows its windows to the full size of the BDP of a path. For instance, if the bandwidth of a network path is 10 Gbps and the RTT is 100 ms, with a packet size of 120 bytes, BDP of the path is 100,000 packets. TCP<sup>\*</sup> grows its window from the mid-point of the BDP, say 50,000, it takes about 50,000 RTTs which amounts to 5000 seconds (1.4 hours). If a flow finishes before that time, it severely under-utilizes the path.

To shorten this under-utilization problem of TCP, many

<sup>\*</sup>A short version [27] of this paper was presented at the International Workshop on Protocols for Fast and Long Distance Networks in 2005.

<sup>†</sup>Note that an convex function (a convex function) grows very slowly at the beginning (slower than a linear function).

<sup>‡</sup>This feature adds to the stability of the protocol because

64

# TCP Cubic Congestion Control

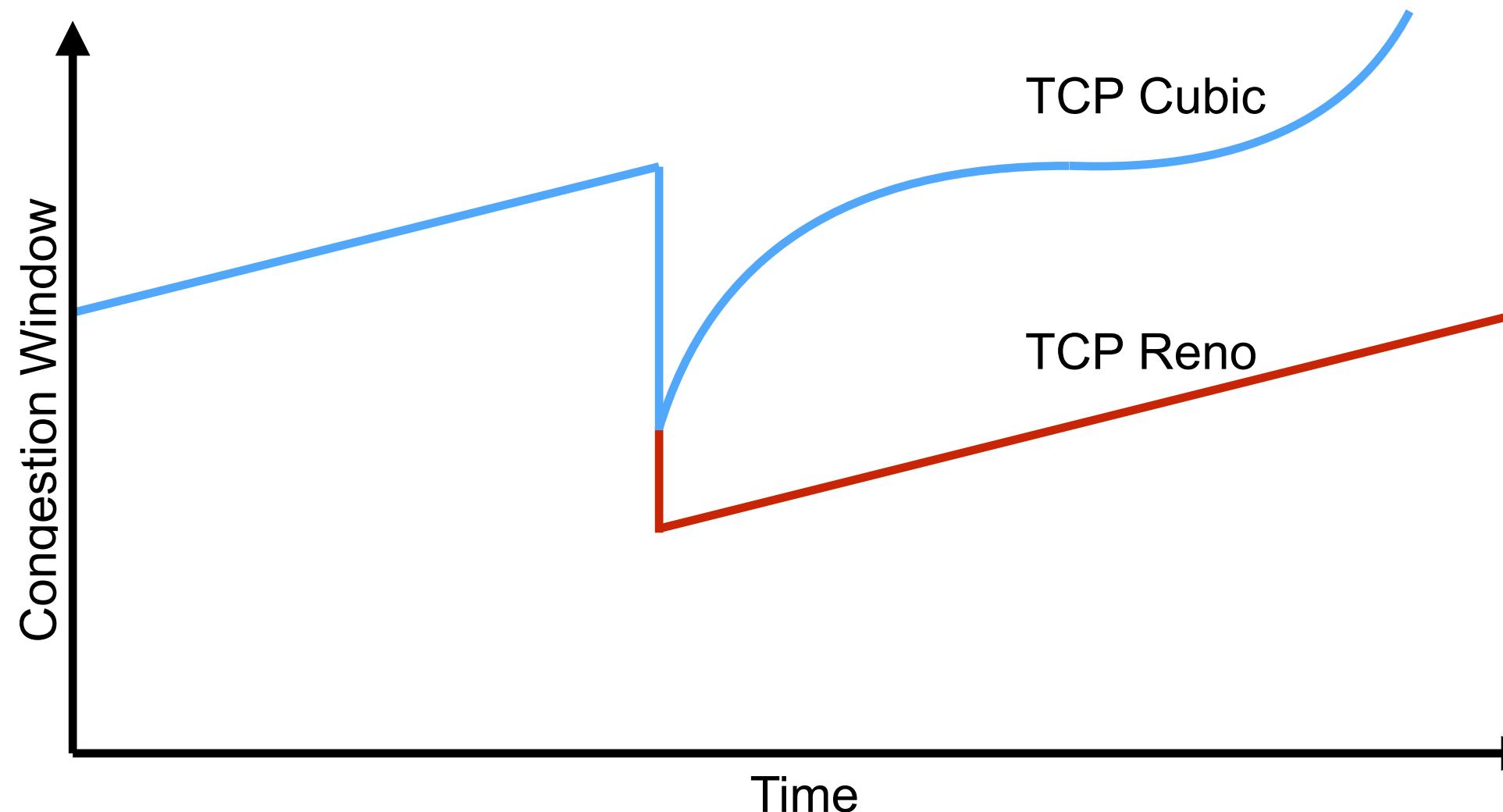
- **On packet loss** during congestion avoidance, TCP Cubic reduces congestion window:  $W_i = W_{i-1} \times 0.7$ 
  - TCP Reno uses  $W_i = W_{i-1} \times 0.5$
  - TCP Cubic is more aggressive
- **After packet loss** during congestion avoidance, TCP Cubic increases the congestion window as:  
$$W_{\text{cubic}} = C(t - K)^3 + W_{\text{max}}$$
  - $W_{\text{max}}$  is the maximum window size reached before the loss
  - $t$  is the time since the packet loss
  - $K$  is the time it will take to increase the window back to  $W_{\text{max}}$ , assuming no further packet losses
  - $C = 0.4$  is a constant that controls fairness to TCP Reno
- **Many additional details included** to ensure fairness with TCP Reno on slower, shorter-RTT, networks

S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating System Review, Vol. 54, Num. 5, pages 64-74, July 2008 <https://dx.doi.org/10.1145/1400097.1400105>

```
Algorithm 1: Linux CUBIC algorithm (v2.2)
Initialization:
  tcp_friendliness ← 1, β ← 0.2
  fast_convergence ← 1, C ← 0.4
  cubic_reset()
On each ACK:
begin
  if dMin then dMin ← min(dMin, RTT)
  else dMin ← RTT
  if cwnd ≤ ssthresh then cwnd ← cwnd + 1
  else
    cnt ← cubic_update()
    if cwnd > cnt then
      cwnd ← cwnd + 1, cwnd_cnt ← 0
    else cwnd_cnt ← cwnd_cnt + 1
  end
Packet loss:
begin
  epoch_start ← 0
  if cwnd < Wlast_max and fast_convergence then
    Wlast_max ← cwnd *  $\frac{(2-\beta)}{2}$  ..... (3.7)
  else Wlast_max ← cwnd
  ssthresh ← cwnd ← cwnd * (1 - β) ..... (3.6)
end
Timeout:
begin
  | cubic_reset()
end
cubic_update(): ..... (3.2)
begin
  ack_cnt ← ack_cnt + 1
  if epoch_start ≤ 0 then
    epoch_start ← tcp_time_stamp
    if cwnd < Wlast_max then
      K ←  $\sqrt[3]{\frac{W_{last\_max}-cwnd}{C}}$ 
      origin_point ← Wlast_max
    else
      K ← 0
      origin_point ← cwnd
    ack_cnt ← 1
    Wtcp ← cwnd
    t ← tcp_time_stamp + dMin - epoch_start
    target ← origin_point + C(t - K)3
    if target > cwnd then cnt ←  $\frac{cwnd}{target-cwnd}$  .. (3.4,3.5)
    else cnt ← 100 * cwnd
    if tcp_friendliness then cubic_tcp_friendliness()
  end
  cubic_tcp_friendliness(): ..... (3.3)
begin
  Wtcp ← Wtcp +  $\frac{3\beta}{2-\beta} * \frac{ack\_cnt}{cwnd}$ 
  ack_cnt ← 0
  if Wtcp > cwnd then
    max_cnt ←  $\frac{cwnd}{W_{tcp}-cwnd}$ 
    if cnt > max_cnt then cnt ← max_cnt
  end
  cubic_reset():
begin
  Wlast_max ← 0, epoch_start ← 0, origin_point ← 0
  dMin ← 0, Wtcp ← 0, K ← 0, ack_cnt ← 0
end
```

# TCP Cubic vs Reno

- TCP Cubic is default in most modern operating systems
  - Much more complex than TCP Reno – core response is relatively straight-forward
  - Much complexity ensures fairness with TCP Reno in its typical operating regime; improves performance for networks with longer RTT and higher bandwidth
- Both algorithms use packet loss as congestion signal and eventually fill router buffers
  - Trade latency for throughput



# TCP Cubic

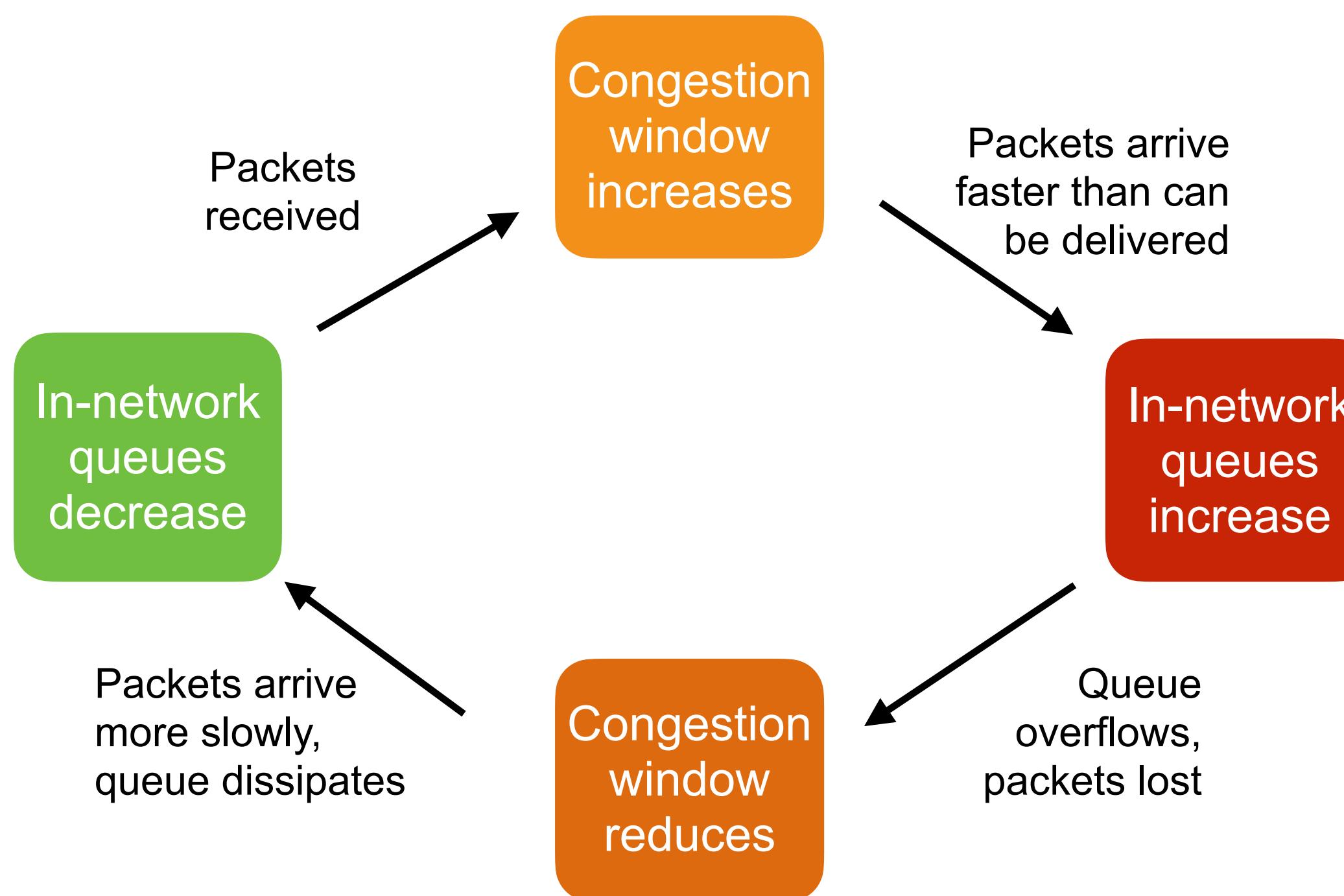
- Improving TCP performance on fast, long-distance, networks

# Delay-based Congestion Control

- Traditional TCP causes latency
- TCP Vegas
- TCP BBR

# Impact of TCP on Latency

- TCP Reno and TCP Cubic **both** aim to fill the network



- TCP Reno and Cubic use packet loss as congestion signal → increase congestion window until queue overflows and packet is lost
- No matter how much big the queue, TCP Reno or Cubic **will** cause it to overflow, if given enough data to send
- Packets waiting in queues within network adds latency → increase RTT

# TCP Vegas: Reducing Latency

- Key insight: if sending faster than the network can deliver, packets will be queued
  - TCP Reno and Cubic wait until the queue overflows and packets are lost before slowing down
  - **TCP Vegas** watches for the increase in delay as the queue starts to fill up → slows down before the queue overflows
    - Queues are smaller → lower latency
    - Packets are not lost, so don't need retransmission
  - Only affects congestion avoidance; slow start unchanged

TCP Vegas: New Techniques for Congestion Detection and Avoidance

Lawrence S. Brakmo      Sean W. O'Malley      Larry L. Peterson\*

Department of Computer Science  
University of Arizona  
Tucson, AZ 85721

**Abstract**

Vegas is a new implementation of TCP that achieves between 40 and 70% better throughput, with one-fifth to one-half the losses, as compared to the implementation of TCP in the Reno distribution of BSD Unix. This paper motivates and describes the three key techniques employed by Vegas, and presents the results of a comprehensive experimental performance study—using both simulations and measurements on the Internet—of the Vegas and Reno implementations of TCP.

**1 Introduction**

Few would argue that one of TCP's strengths lies in its adaptive retransmission and congestion control mechanism, with Jacobson's paper [4] providing the cornerstone of that mechanism. This paper attempts to go beyond this earlier work; to provide some new insights into congestion control, and to propose modifications to the implementation of TCP that exploit these insights.

The tangible result of this effort is a new implementation of TCP that we refer to as TCP Vegas. This name is a take-off of earlier implementations of TCP that were distributed in releases of 4.3 BSD Unix known as Tahoe and Reno; we use Tahoe and Reno to refer to the TCP implementation instead of the Unix release. Note that Vegas does not involve any changes to the TCP specification; it is merely an alternative implementation that interoperates with any other valid implementation of TCP. In fact, all the changes are confined to the sending side.

\*This work supported in part by National Science Foundation Grant IRI-9015407 and ARPA Contract DABT63-91-C-0030.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication, and its date appear, and notice is given that copying is by permission of the Association of Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.  
SIGCOMM 94 -8/94 London England UK  
© 1994 ACM 0-89791-682-4/94/0008-\$3.50

2 Tools

This section briefly describes the tools used to implement and analyze the different versions of TCP. All of the protocols were developed and tested under the University of Arizona's *x-kernel* framework [3]. Our implementation of Reno was derived by retrofitting the BSD implementation into the *x-kernel*. Our implementation of Vegas was derived by modifying Reno.

**2.1 Simulator**

Many of the results reported in this paper were obtained from a network simulator. Even though several good simulators are available—e.g., REAL [9] and Netsim [2]—we decided to build our own simulator based on the *x-kernel*.

<sup>1</sup>We limit our discussion to Reno, which is both newer and better performing than Tahoe. Also note that in terms of the congestion-related algorithms, Reno is roughly equivalent to the Berkeley Network Release 2 (BNR2) implementation of TCP.

24

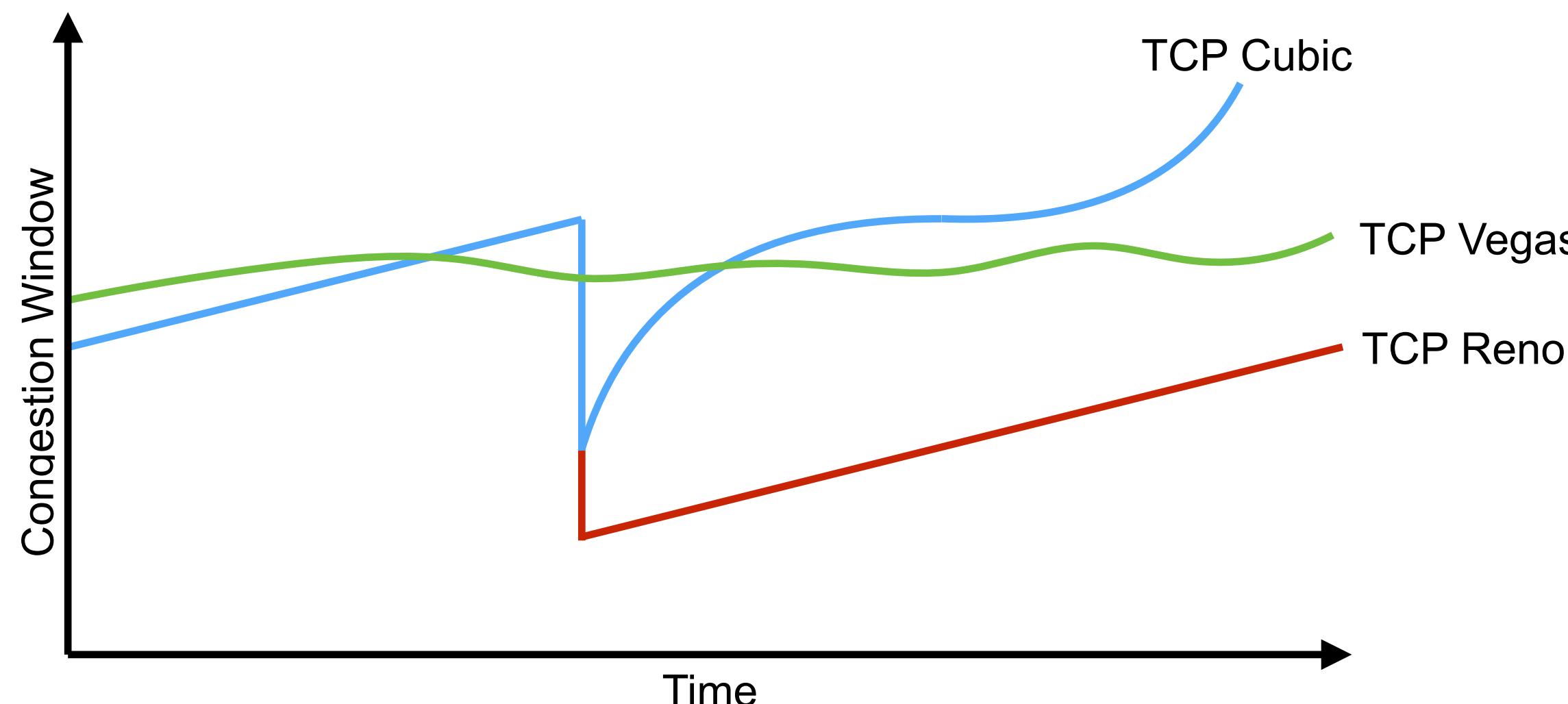
L. Brakmo, S. O'Malley, and L. Peterson, “TCP Vegas: New techniques for congestion detection and avoidance”, ACM SIGCOMM Conference, London, August 1994.  
<https://dx.doi.org/10.1145/190314.190317>

# TCP Vegas Congestion Control (1/2)

- Measure **BaseRTT**
  - The smallest time between sending a packet and getting its acknowledgement
  - BaseRTT will be from a packet delivered without much queueing
- Calculate **ExpectedRate** =  $W/\text{BaseRTT}$ 
  - The congestion window,  $W$ , determines how many packets are sent each RTT
  - If the network can support this sending rate, the complete window should be delivered within the RTT
- Measure **ActualRate**
  - Bytes sent divided by actual RTT, for each packet
  - Measure how fast packets are actually received, based on acknowledgements returned
  - ActualRate  $\leq$  ExpectedRate since packets can't be delivered faster than they're sent

# TCP Vegas Congestion Control (2/2)

- Compare ExpectedRate with ActualRate and adjust window:
  - If  $\text{ExpectedRate} - \text{ActualRate} < R_1$  then additive increase to window
    - Data arriving at close to expected rate, can likely send faster
  - If  $\text{ExpectedRate} - \text{ActualRate} > R_2$  then additive decrease to window
    - Data arriving slower than it's being sent, slow down
  - Parameters  $R_1$  and  $R_2$ , where  $R_1 < R_2$ , critical to performance



Additive increase and decrease in size window → smoother changes in sending rate than Reno or Cubic

# TCP Vegas: Limitations

- Delay-based congestion control is a good idea in principle
  - Reduces latency
  - Reduces packet loss
- But loss- and delay-based congestion control don't cooperate
  - TCP Reno and TCP Cubic aggressively increase queue sizes
  - Increases RTT and reduces ActualRate as measured by TCP Vegas
  - Forces TCP Vegas to slow down; cycle repeats → TCP Vegas sending rate drops to zero over time
- TCP Vegas is not used, because it can't be deployed alongside TCP Reno or TCP Cubic

# TCP BBR

- Is it possible to develop a delay-based congestion control algorithm for TCP that can be deployed on a network alongside TCP Reno and TCP Cubic?
- **Maybe** – TCP BBR (“Bottleneck Bandwidth & RTT”) is one attempt
  - Proposal from Google – measures RTT and bandwidth of the bottleneck link, directly sets congestion window
  - Used by YouTube in some cases, but **highly experimental**
    - Recent work by Ranysha Ware and Justine Sherry shows serious fairness problems with TCP BBR v1
      - <https://dl.acm.org/doi/10.1145/3355369.3355604>
      - <https://vimeo.com/showcase/6531379/video/369121357#t=990s>
    - TCP BBR v2 in development, might solve these problems – this is an active research area



The image shows a page from ACM Queue magazine. At the top right is a large green 'BBR' logo with the text 'Congestion-Based Congestion Control'. Below the logo is a section titled 'MEASURING BOTTLENECK BANDWIDTH AND ROUND-TRIP PROPAGATION TIME'. The main text begins with a large bold 'B'. The authors listed are NEAL CARDWELL, YUCHUNG CHENG, C. STEPHEN GUNN, SOHEIL HASSAS YEGANEH, and VAN JACOBSON. The text discusses the limitations of current TCP congestion control and the need for more accurate measurements. It highlights the work on BBR and its potential to improve network performance.

Y all accounts, today's Internet is not moving data as well as it should. Most of the world's cellular users experience delays of seconds to minutes; public Wi-Fi in airports and conference venues is often worse. Physics and climate researchers need to exchange petabytes of data with global collaborators but find their carefully engineered multi-Gbps infrastructure often delivers at only a few Mbps over intercontinental distances.<sup>6</sup> These problems result from a design choice made when TCP congestion control was created in the 1980s—interpreting packet loss as “congestion.”<sup>13</sup> This equivalence was true at the time but was because of technology limitations, not first principles. As NICs (network interface controllers) evolved from Mbps to Gbps and memory chips from KB to GB, the relationship between packet loss and congestion became more tenuous. Today TCP’s loss-based congestion control—even with the current best of breed, CUBIC!<sup>14</sup>—is the primary cause of these problems. When bottleneck buffers are large,

acmqueue | september-october 2016 20

N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. BBR: Congestion-based congestion control. ACM Queue, 14(5), September 2016. <https://dl.acm.org/doi/10.1145/3012426.3022184>

# Delay-based Congestion Control

- Traditional TCP causes latency
- TCP Vegas
- TCP BBR

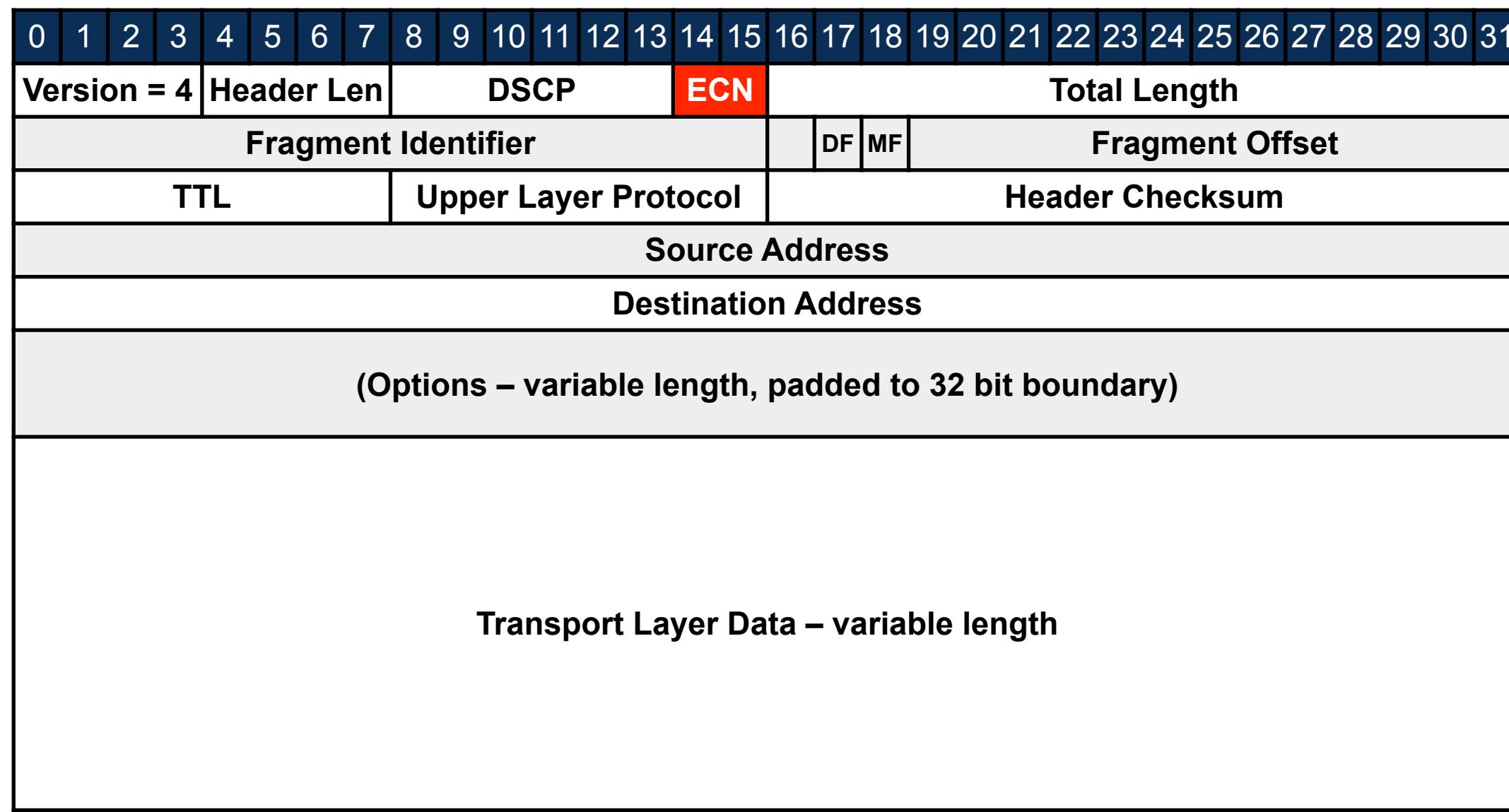
# Explicit Congestion Notification (ECN)

- ECN for TCP and IP

# Explicit Congestion Notification

- TCP has inferred congestion through measurement
  - TCP Reno and Cubic → packets lost when queues overflow
    - Problematic because it increases delay
    - Problematic because of non-congestive loss on wireless links
  - TCP Vegas → increase in delay as queues start to fill
    - Conceptually a good idea, but difficult to deploy when competing with TCP Reno and Cubic
- **Why not have the network tell TCP congestion is occurring?**
  - Explicit Congestion Notification (ECN) field in IP header
  - Tell TCP to slow down if it's overloading the network

# ECN and IP



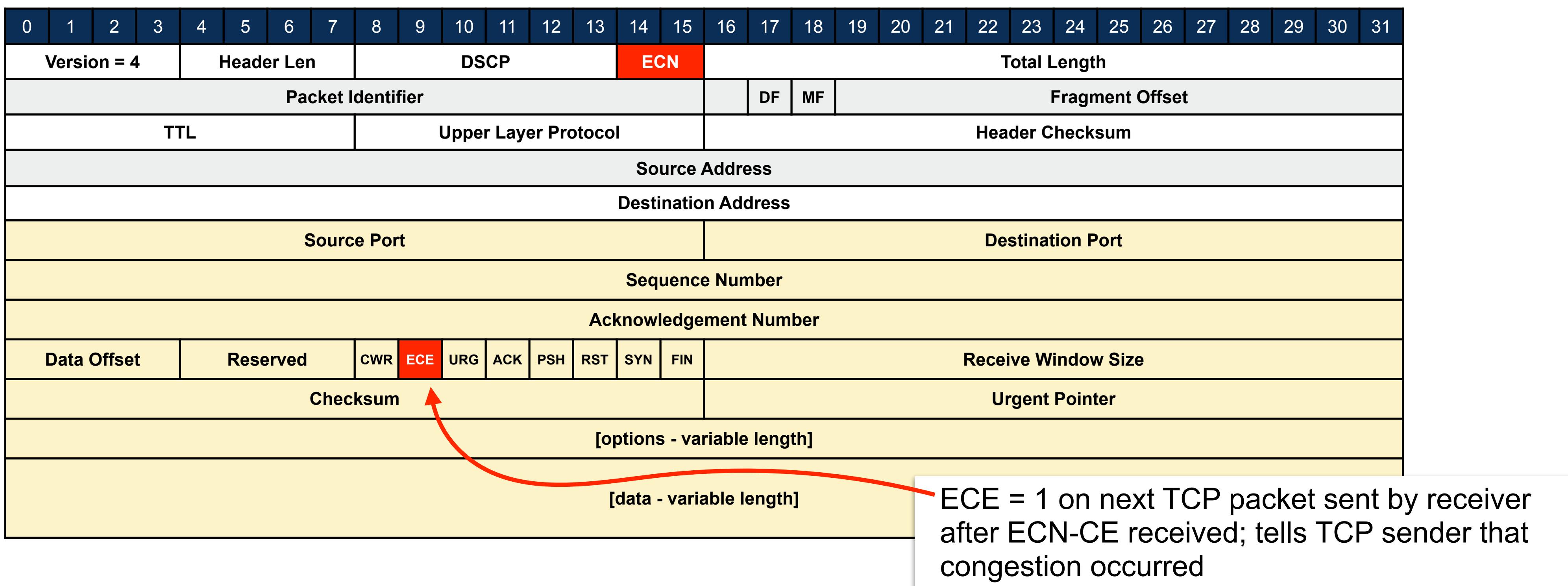
- Sender sets ECN bits on transmission:
  - 00 = Doesn't support ECN, 01 = ECN capable
- If congestion occurs, **congested router** sets ECN bits to 11 ("congestion experienced")
  - Indicates that a queue of packets for some link is getting full, but has not yet overflowed – **signal of congestion set by the network**
  - If the queue overflows, packets are dropped
- Cooperation between network and endpoints

K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", IETF RFC 3168, September 2001. <https://datatracker.ietf.org/doc/rfc3168/>

Diagram shows IPv4, but ECN is also present in IPv6 and works the same

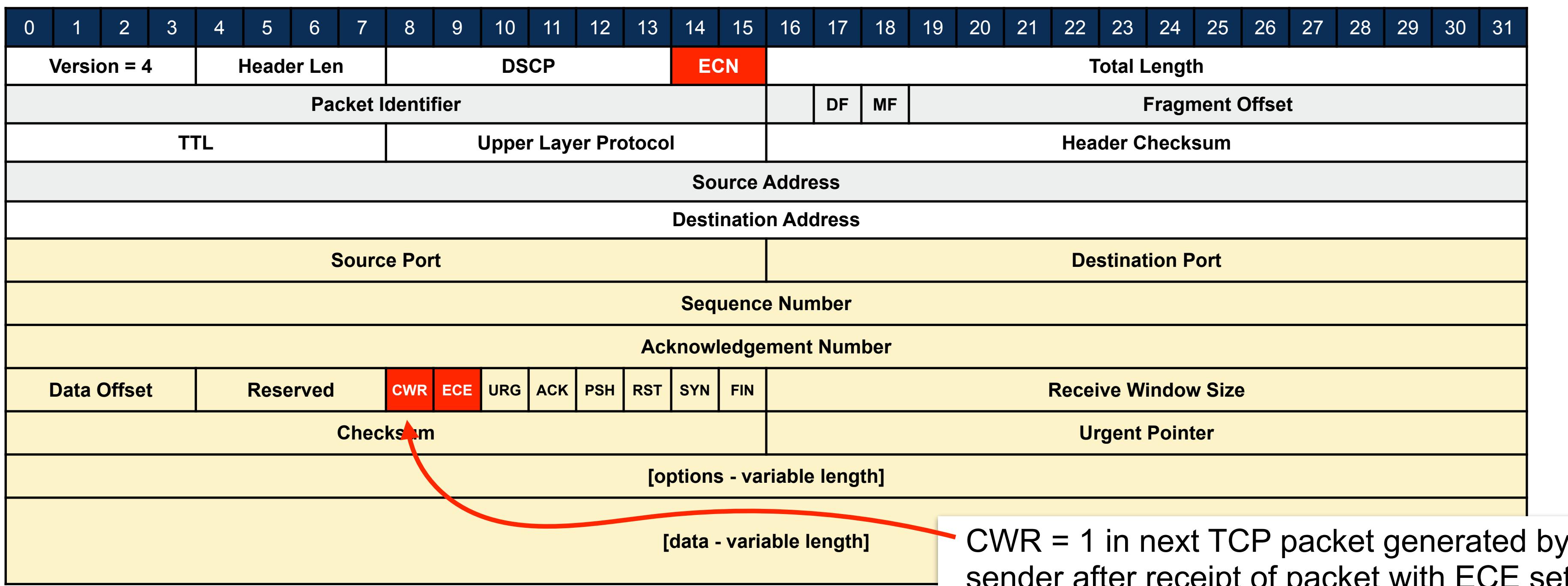
# ECN and TCP (1/3)

- Receiver may get a TCP segment within an IP packet marked ECN Congestion Experienced (ECN-CE)
- ECN Echo (ECE) field in TCP header allow it to signal this back to the sender



# ECN and TCP (2/3)

- Sender reduces congestion window on receiving acknowledgement with ECE = 1 as if the packet had been lost
- Sets CWR = 1 in next packet to inform network and receiver it has done so



# ECN and TCP (3/3)

- ECN lets TCP react to congestion before packet loss occurs
  - Routers signal congestion **before** queues overflow
  - Independent of choice of TCP congestion control algorithm
  - Smaller queues → reduces latency, beneficial for video conferencing and gaming
- ECN extensions also exist in QUIC and RTP

# ECN and TCP: Deployment

- TCP endpoints needed to be updated to support ECN → done
  - Endpoints now support ECN and **many** now enable ECN by default
    - ECN was widely implemented but disabled-by-default for many years, due to concerns that it wouldn't pass through firewalls
    - Apple forced firewalls to be fixed: iOS 9 enabled ECN for 5% of connection; iOS 10 for 50%; iOS 11 and later for all connections
  - Most routers support ECN, relatively few enable it by default
    - Endpoints can react to ECN signals, but network is currently unlikely to provide them
    - Starting to change – e.g., recent DOCSIS cable modems enable ECN
    - Will take time, but latency reduced as ECN is enabled throughout the network

# Explicit Congestion Notification (ECN)

- ECN for TCP and IP

# Light Speed?

- Impact of propagation delay on latency

# Light Speed?

- Two factors impact latency for data transfer
  - The time packets spend in queues within the network
    - Impact of TCP congestion control algorithm
    - ECN to signal congestion before queues overflow
  - **The time packets spend propagating down the links between routers**
    - Speed at which light propagates through an optical fibre
    - Speed at which electrical signals propagate down a wire
    - Speed at which radio signals propagate through the air

# Reducing Propagation Delay (1/3)

- Physically shorter links have lower propagation delay
  - Significant reduction in latency by laying new cables that follow great circle routes between popular destinations
  - e.g., ocean floor cable from Japan to Europe via the north pole, rather than Pacific Ocean – US – Atlantic Ocean



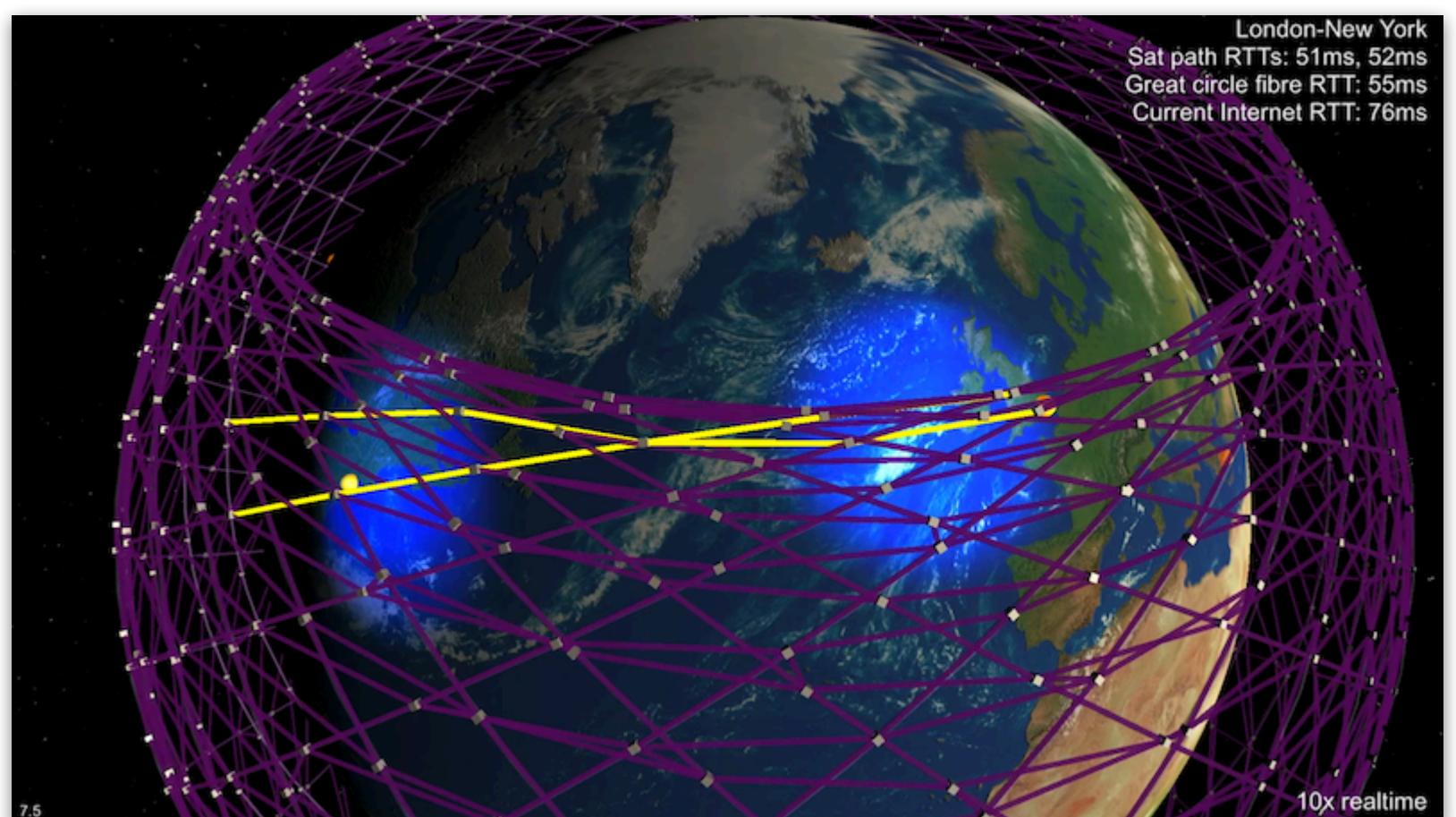
<https://www.scientificamerican.com/article/an-internet-cable-will-soon-cross-the-arctic-circle/>

# Reducing Propagation Delay (2/3)

- Signals propagate at the speed of light **in the transmission medium**
  - Speed of light in optical fibre ~200,000 km/s
  - Speed of light in vacuum 299,792 km/s
- SpaceX Starlink deploying ~4,000 low earth orbit communications satellites
  - ~47% reduction in latency since signals sent through vacuum rather than optical fibre
  - With careful routing, satellite path can follow close to the most direct great circle route



Source: Wikipedia



Source: Mark Handley

M. Handley, "Delay is Not an Option: Low Latency Routing in Space",  
Proc. ACM HotNets '18. <https://doi.org/10.1145/3286062.3286075>

# Reducing Propagation Delay (3/3)

- What application cares so much about reducing latency to spend \$billions launching >4,000 satellites? Or new undersea cables? High frequency share trading

# Lowering Latency

- Latency due to queuing and propagation delays
- Queuing influenced by congestion control
- Propagation delays due to speed of light

# Real-time and Interactive Applications

Networked Systems (H)  
Lecture 7



# Lecture Outline

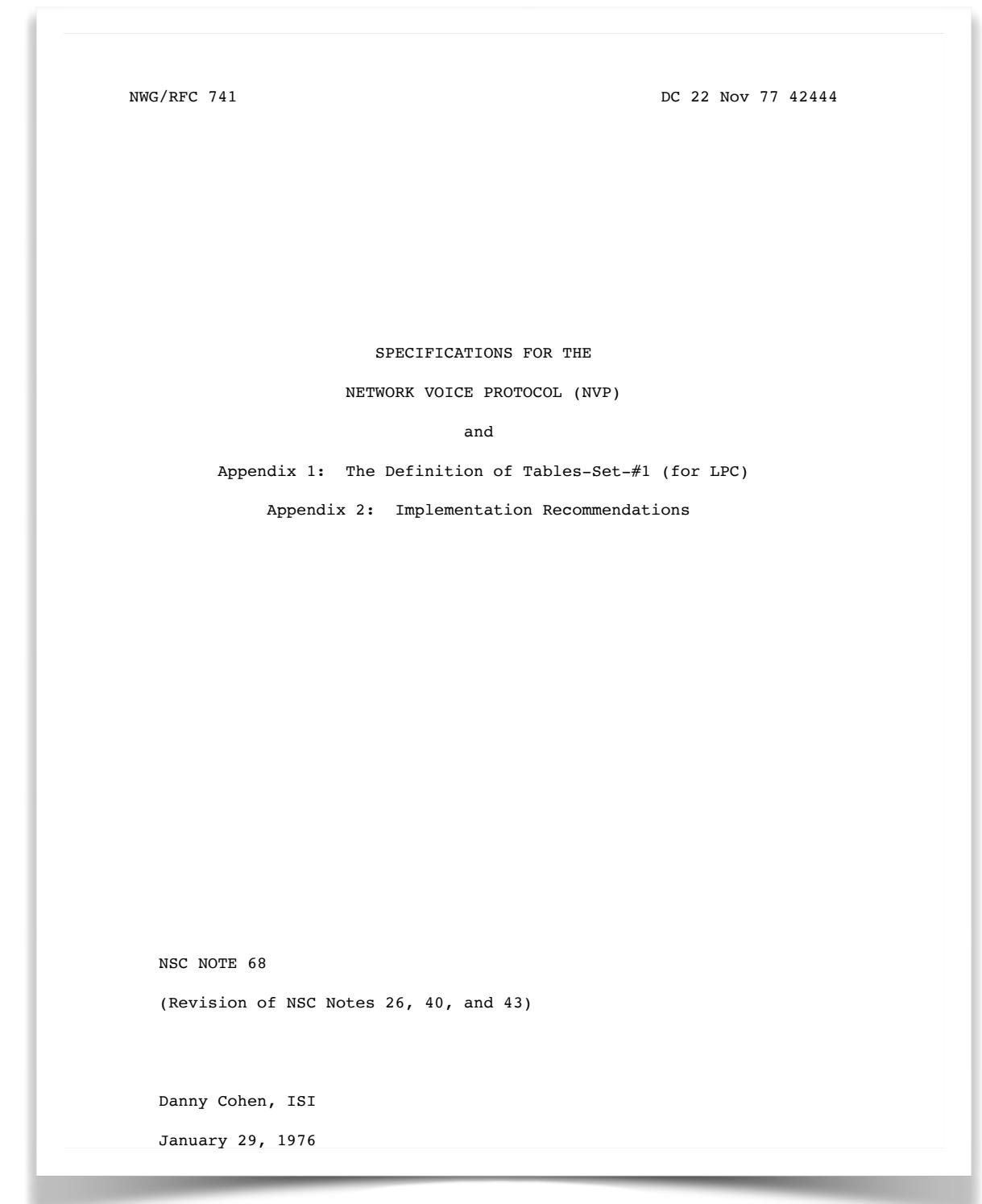
- Real-time applications and the Internet
  - What is real-time traffic?
  - Requirements and constraints
  - Quality-of-service and quality-of-experience
- Interactive applications
  - Conferencing application architecture
  - Signalling, session description, and real-time media traffic
- Streaming applications
  - HTTP adaptive streaming for video on demand

# Real-time Media Over The Internet

- What is real-time traffic?
- Requirements and constraints
- Quality of service vs. user experience

# Real-time Traffic on the Internet

- Long history of real-time traffic over the Internet:
  - Telephony and voice-over-IP (VoIP), Internet radio, video conferencing, streaming video and TV, gaming, sensors, industrial control
  - Initial packet voice experiments → 1970s
    - Network Voice Protocol
  - Core standards defined → mid-1990s
    - SIP, SDP, RTP, RTSP, H.323
  - HTTP adaptive streaming → mid-2010s
    - MPEG DASH
- The design of the network and its transport protocols has evolved to support real-time media



<https://datatracker.ietf.org/doc/rfc741/>

# What is Real-time Traffic?

- Defining characteristic: **deadlines**
  - The system fails if data is not delivered by a certain time
    - Railway signalling must deliver data to change a signal before train arrives – **hard real-time**
    - Streaming video should deliver a new frame every 60<sup>th</sup> of a second, else the movie playback stutters – **soft real-time**
  - What is the probability a deadline is missed? What are the consequences? Engineer accordingly – **no system is 100% reliable**
  - Deadlines can be absolute or relative
    - Data must be delivered before a certain time – **absolute deadline**
    - Data must be delivered periodically, within some time after the previous – **relative deadline**
- Real-time is not necessarily high performance
  - It requires **predictable timing**, not necessarily high bandwidth or low latency

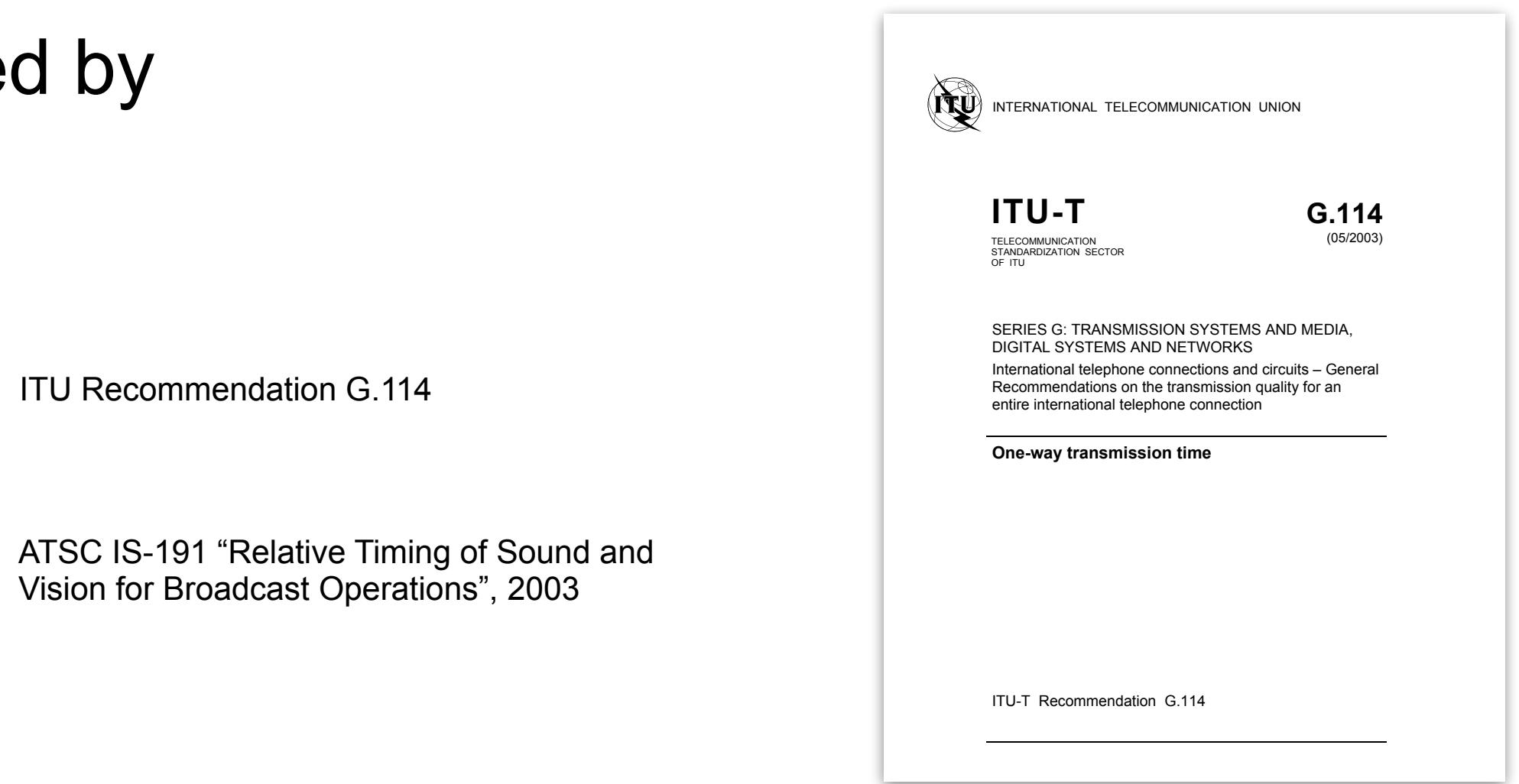
# Requirements for Streaming Applications

- Video on demand has no absolute deadline, but needs regular data to prevent stalls once started
  - Acceptable if it takes a few seconds to start the video
  - Playback should be smooth once it starts
- Live video may have absolute deadlines
  - e.g., live sports playout
- Bit rate depends on desired quality
  - Higher quality is better, up to limits of display
  - Predictable, but lower, quality often preferred to more variable, but on average higher, quality
- For a given bit rate, trade-off between frame rate and frame quality
  - Smoother motion or higher resolution?

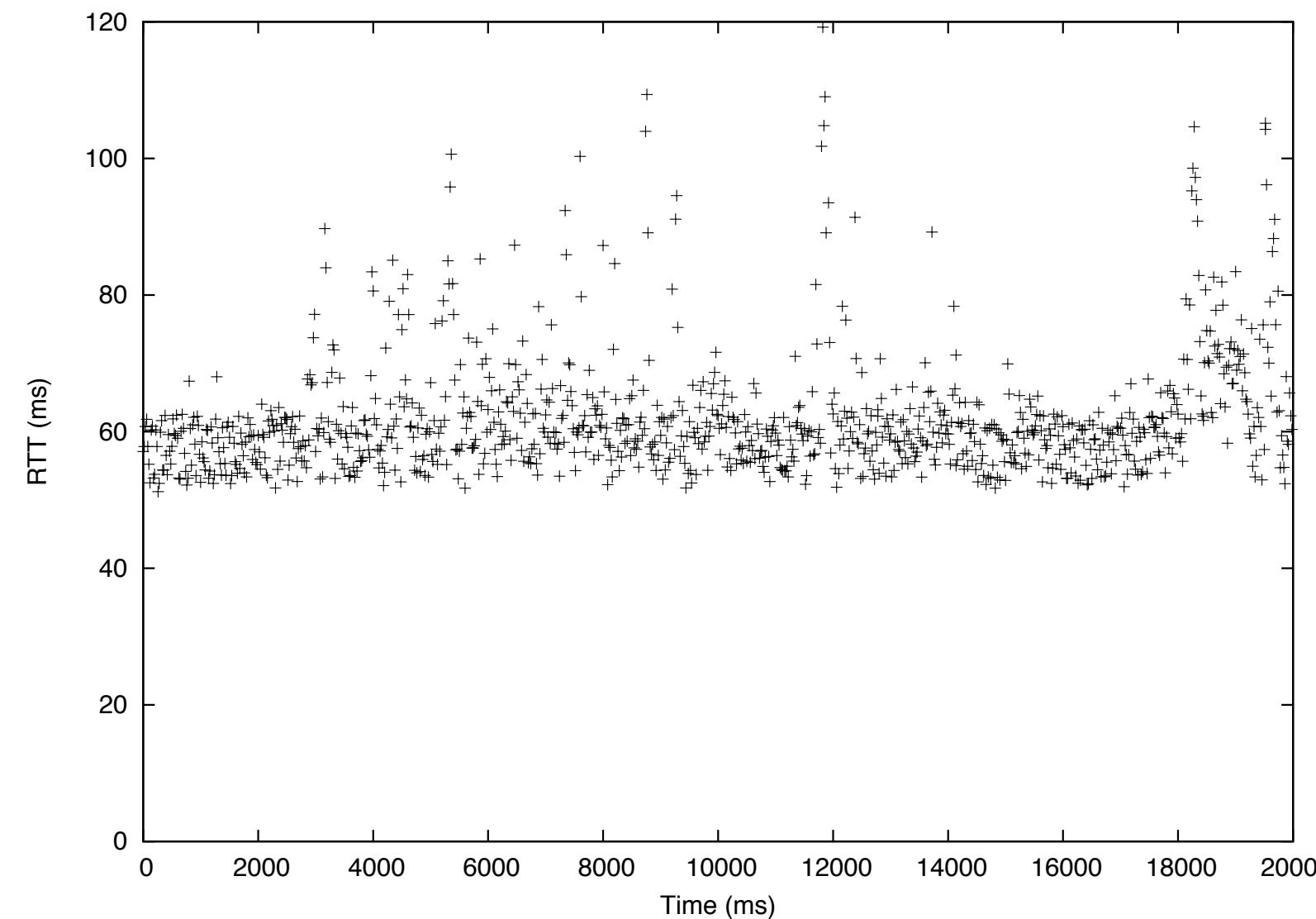


# Requirements for Interactivity

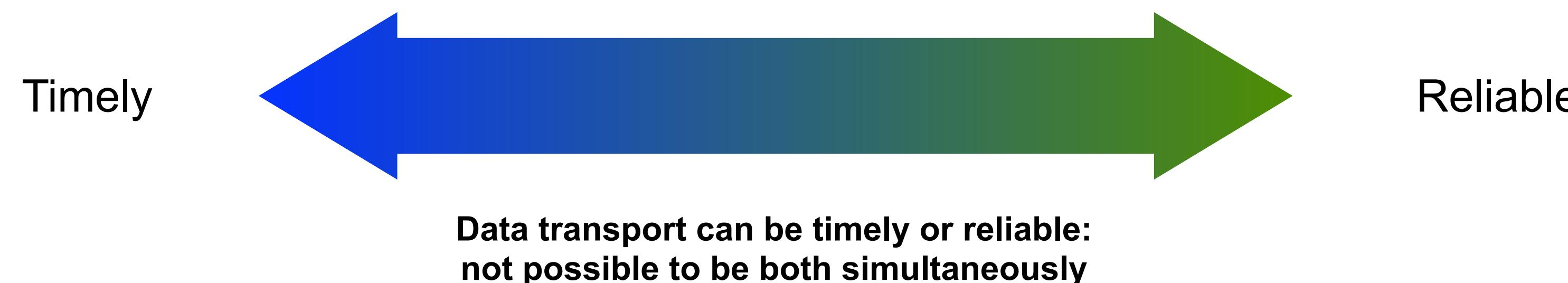
- Requirements for interactive applications determined by task and human perception
  - Phone call or video conference
    - One-way mouth-to-ear delay ~150ms maximum for telephony
    - Video conferences want to lip-sync audio and video
      - Audio should be no more than 15ms ahead, or 45ms behind, video
  - Lecture style
    - Mostly unidirectional with occasional questions → can tolerate much higher latency
  - Distributed music performance
    - One-way latency <50ms desirable
    - Speed of sound: ~15ms to go from one side of a large orchestra to the other



# Real-time Traffic Must Be Loss Tolerant

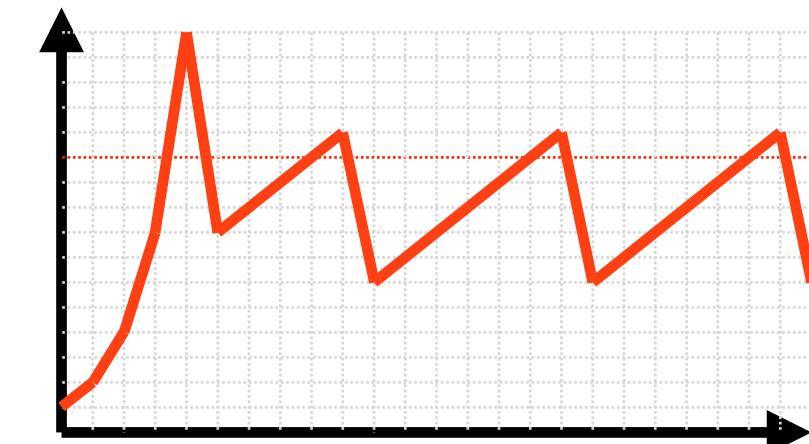


- The Internet is a best effort packet network
  - Timing is not guaranteed – depends on the amount of queueing on the path and on the path taken
  - Packets can be lost – retransmissions take time, and may not arrive before the deadline
  - In some networks these effects can be significant – with careful engineering they can be insignificant
- Real-time applications need to be loss tolerant



# Real-time Traffic Has Limited Elasticity

- TCP congestion control adapts transmission speed to match available capacity
- Many transfers are **elastic** – faster is better, but it doesn't matter what rate the congestion control selects
  - The application can adapt to any chosen rate
- Real-time traffic is **inelastic**
  - Media has a minimum rate, below which it cannot be used
    - e.g., a minimum quality level below which the speech is unintelligible, that needs a certain bit-rate to transmit
    - If the congestion control algorithm cannot sustain this rate – if the network has insufficient capacity – real-time traffic cannot be used
  - Media has a maximum rate → cannot consume more
    - Video limited by capture frame rate and resolution
    - Audio limited by capture sampling rate



# Quality of Service (QoS) Guarantees

- It may be possible to reserve network capacity for real-time traffic
  - Using RSVP, MPLS, 5G network slicing, ...
  - Requires flow setup signalling, authorisation, and accounting
    - Need to tell the network what resources the real-time traffic requires
    - Need to demonstrate that sender is allowed to reserve these resources and can pay for the reserved capacity – reservations prevent other users from accessing the network, so have some cost
  - If the network has capacity for the traffic, reservations don't help
  - If the network does not have capacity, they allow the operator to discriminate in favour of customers who are willing to pay
    - Some customers care enough to pay for reserved capacity
    - Inter-domain reservations and accounting can be difficult to agree and implement
    - Many operators find it cheaper and easier to just buy more capacity, so no need to reserve

# Quality of Experience (QoE)

- What ultimately matters is subjective quality of experience
  - Does the application meet user needs?
  - Does the application allow users to communicate effectively?
  - Does the application provide compelling entertainment?
- QoE is not a one-dimensional metric
  - What aspect of user experience are you evaluating?
    - e.g., “Does it sound good” is not the same as “Can you understand it?”
  - How does your metric relate to the task being performed?
  - Some aspects of user experience can be estimated from technical metrics
    - e.g., the ITU-T E-model for speech quality evaluation based on latency and packet loss
  - Some aspects are subjective and task dependent – need user trials

# Real-time Media Over The Internet

- What is real-time traffic?
- Requirements and constraints
- Quality of service vs. user experience

# Interactive Applications

- Structure of video conferencing systems
- Protocols for video conferencing
- Media transport

# Interactive Conferencing Applications

- What are interactive applications?
  - Telephony
  - Voice-over-IP (VoIP)
  - Video conferencing
- Long history of research and standardisation:
  - Network Voice Protocol (NVP)
    - Packet voice experiments over the ARPAnet
    - RFC 741 published in 1976
  - Modern standards development from mid-1990s
    - Mbone conferencing tools
    - SIP, SDP, and RTP protocols
    - Adopted by 3GPP as basis of mobile telephone standards
  - Browser-based conferencing from mid-2010s
    - WebRTC



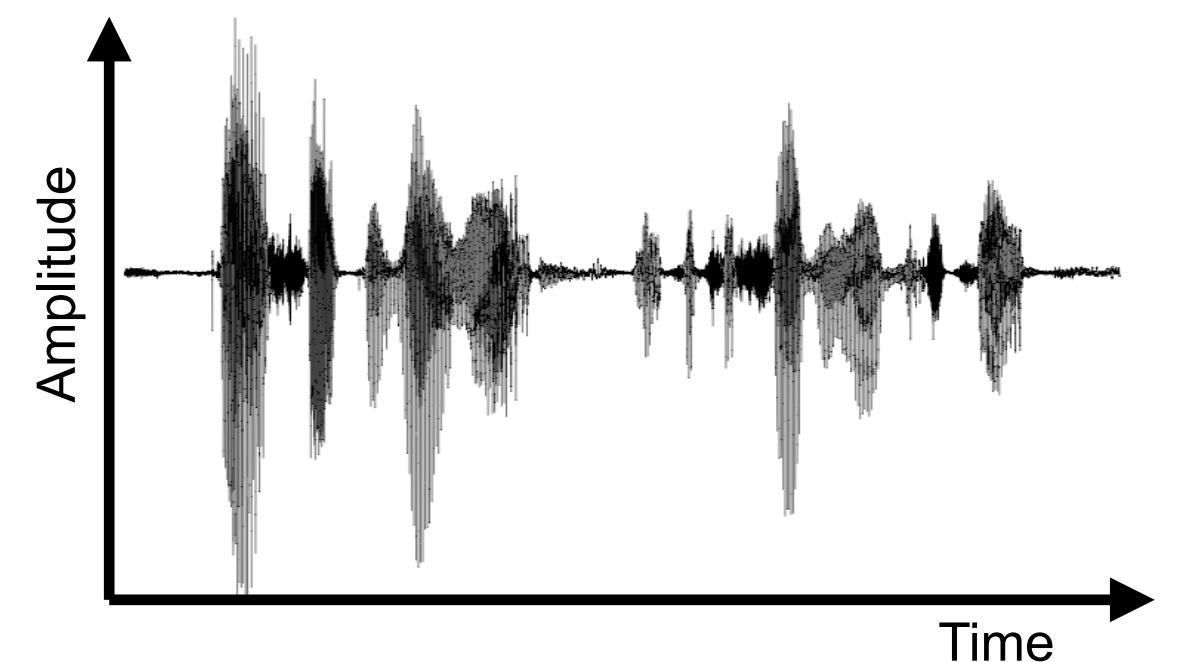
Multi-party video conferencing over IP, 1998



CU-SeeME (source: Wikipedia)

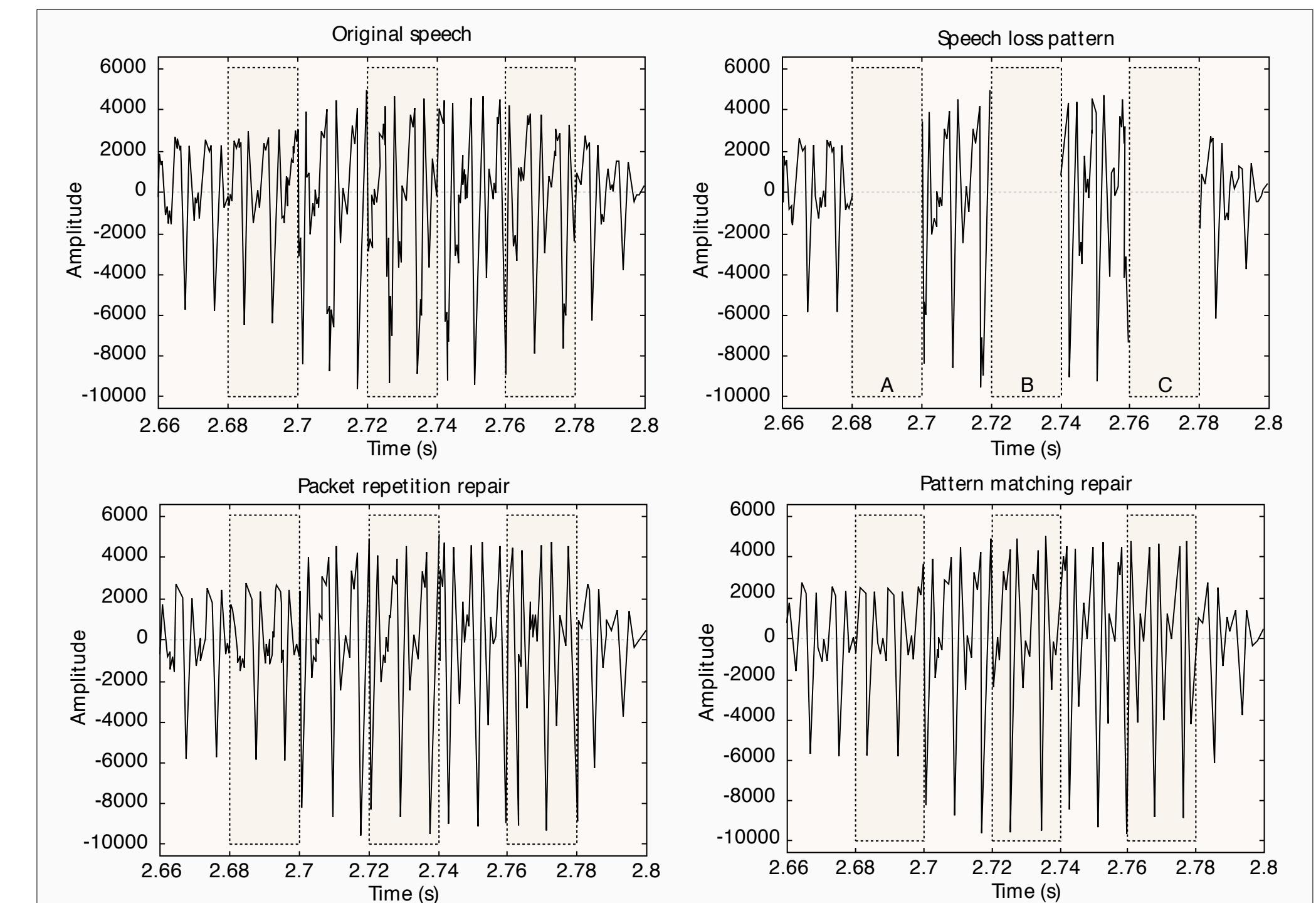
# Requirements on Timing and Data Rate

- Interactive conferencing has tight latency bounds
  - One-way mouth-to-ear delay ~150ms maximum for telephony
  - Video conferences want to lip-sync audio and video
    - Audio should be no more than 15ms ahead, or 45ms behind, video
  - User experience degrades gracefully
- Data rates depend on media type and encoding options
  - Speech coding typically operates on 20ms packets
    - Captures one frame of speech every 20ms, encodes, transmits
      - Data rate ~tens of kilobits per second
      - Background noise during quiet periods encoded at lower quality, packets sent less often
  - Video frame rate and resolution highly variable
    - High definition video encoding using H.264 is around 2-4Mbps
    - Frame rates between 25 and 60fps commonly used
    - I-frames → tens of packets; P-frames → single or few packets



# Reliability Requirements

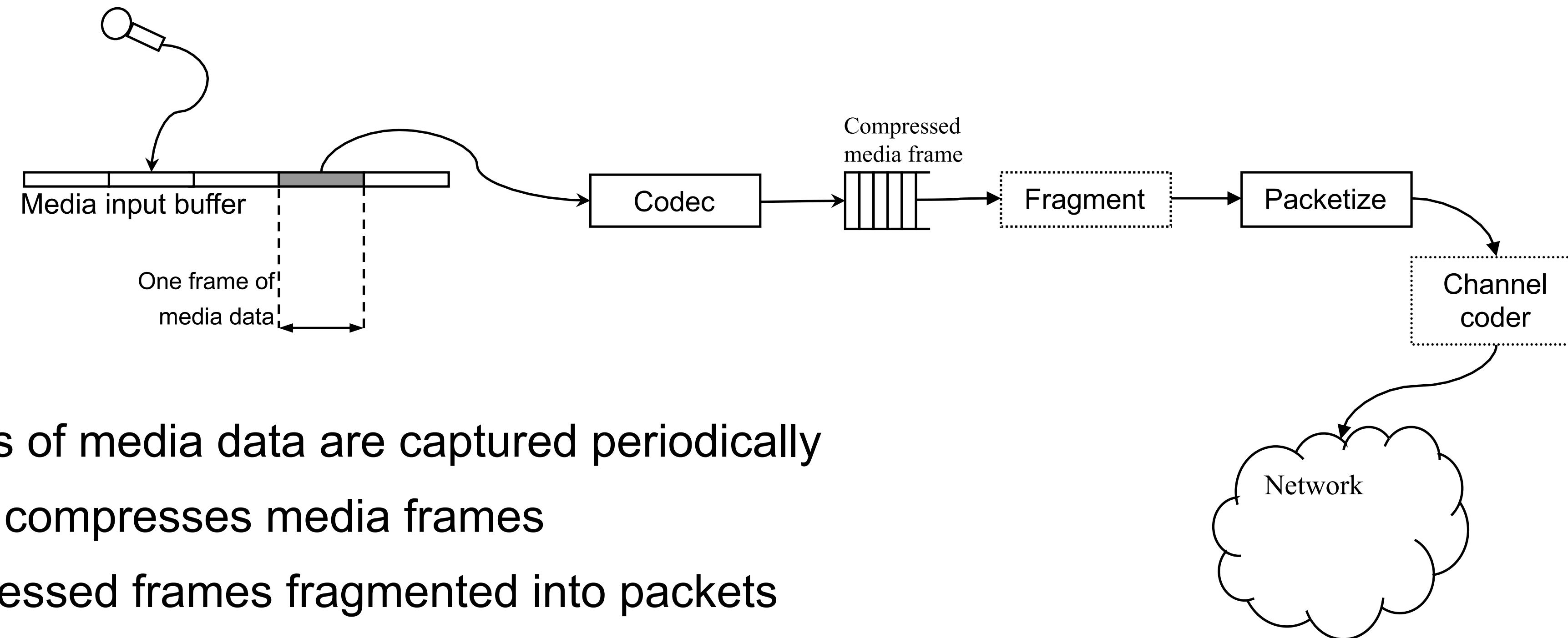
- Speech data is highly loss tolerant
  - Loss concealment can hide 10-20% random packet loss without noticeable loss in quality
  - Burst losses are less well concealed
- Video packet loss hard to conceal
  - No scene changes to reset decoder state to known good value
  - Retransmissions possible in some cases; forward error correction more typical



■ Figure 9. (a) Sample error concealment techniques: original audio signal; (b) sample error concealment techniques: the loss pattern; (c) sample error concealment techniques; packet repetition; (d) sample error concealment techniques: one sided waveform substitution.

C. S. Perkins, O. Hodson, and V. Hardman, [A Survey of Packet Loss Recovery Techniques for Streaming Media](#), IEEE Network Magazine, September 1998. DOI:[10.1109/65.730750](https://doi.org/10.1109/65.730750)

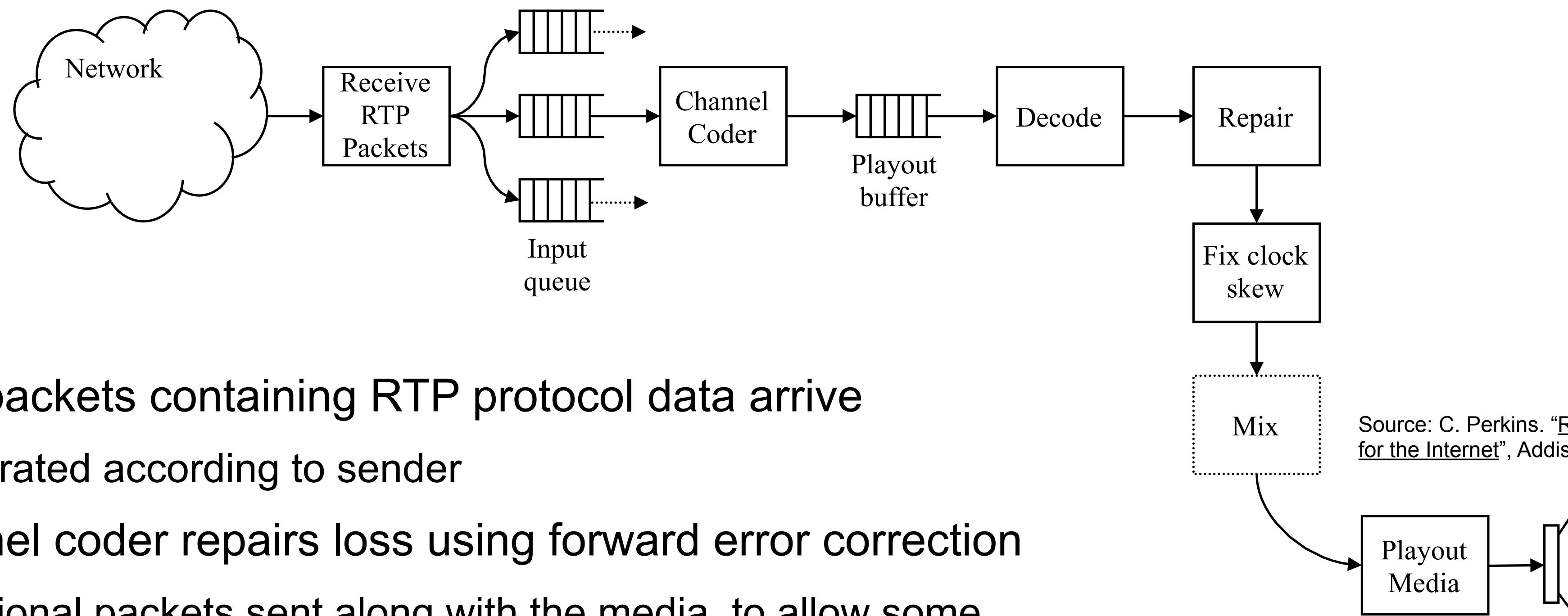
# Interactive Applications: Media Transmission Path



- Frames of media data are captured periodically
- Codec compresses media frames
- Compressed frames fragmented into packets
  - Transmitted using RTP inside UDP packets
  - RTP protocol adds timing and sequencing, source identification, payload identification
- Transmitted over the network

Source: C. Perkins. "[RTP: Audio and Video for the Internet](#)", Addison-Wesley, 2003

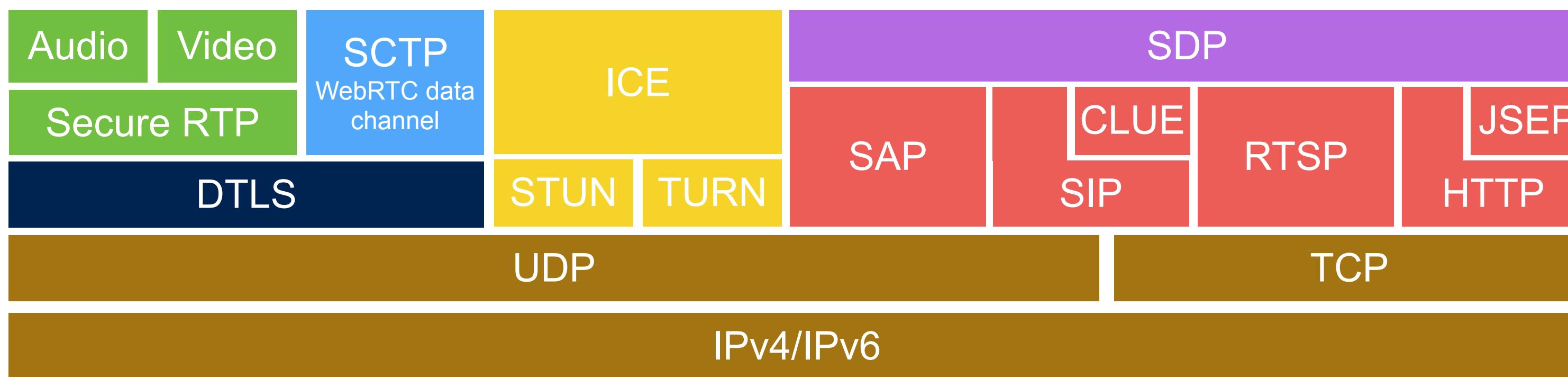
# Interactive Applications: Media Reception Path



- UDP packets containing RTP protocol data arrive
  - Separated according to sender
- Channel coder repairs loss using forward error correction
  - Additional packets sent along with the media, to allow some repair without needed retransmission
- Playout buffer used to reconstruct order, smooth timing
- Media is decompressed, packet loss concealed, and clock skew corrected
- Recovered media is rendered to user

Source: C. Perkins. "[RTP: Audio and Video for the Internet](#)", Addison-Wesley, 2003

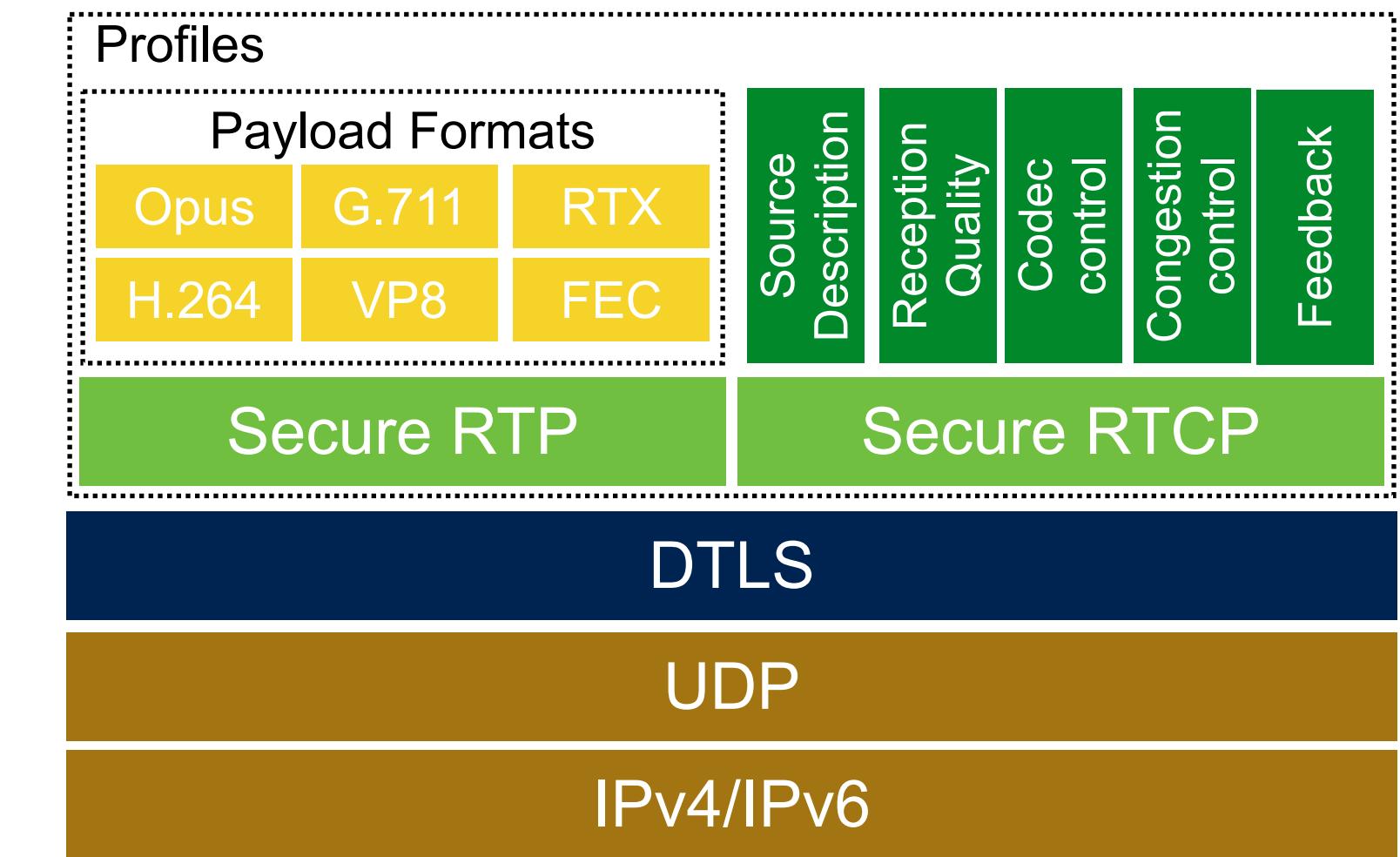
# Internet Multimedia Standards



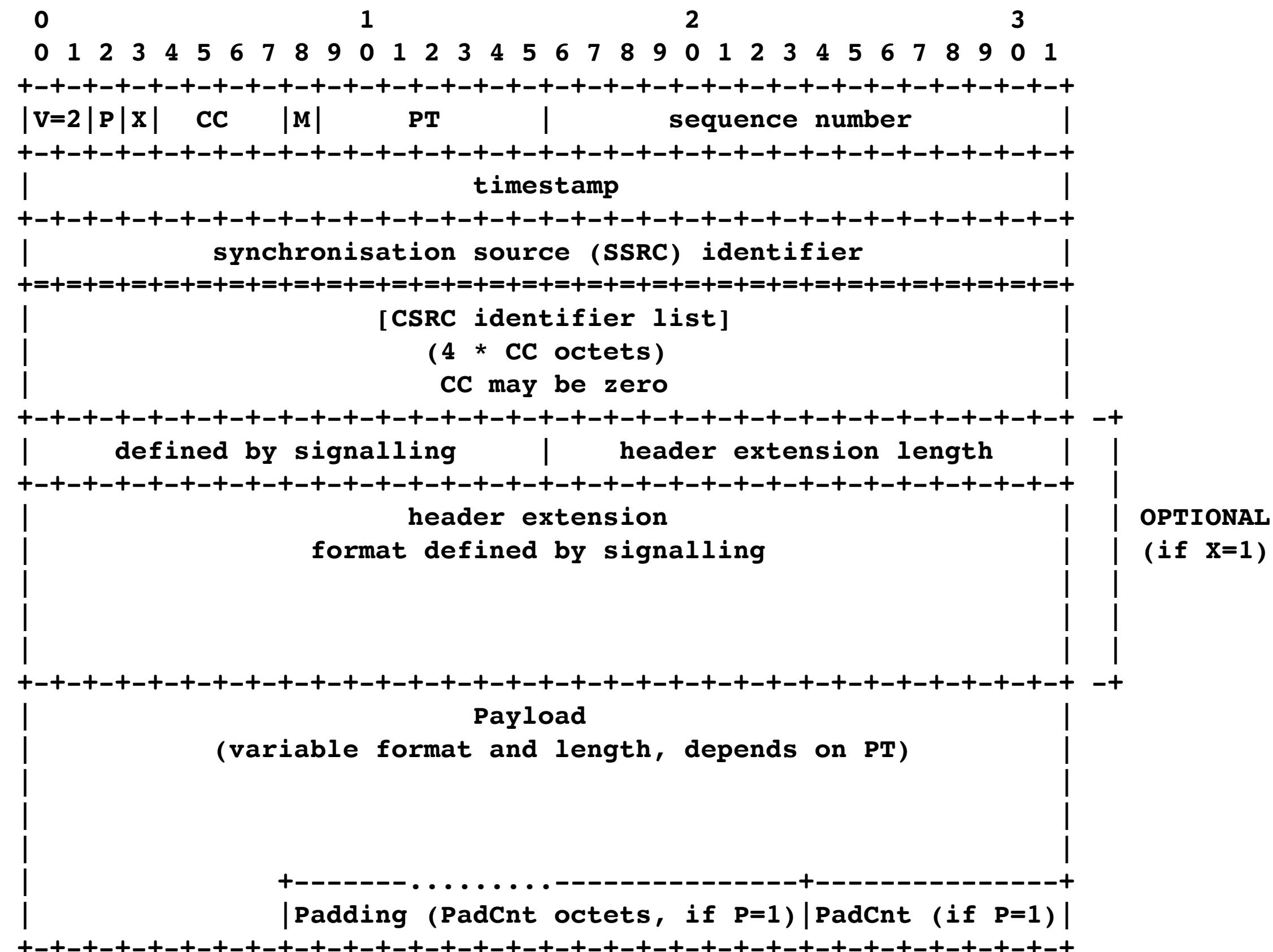
- Media: secure RTP, WebRTC data channel
- Path discovery and NAT traversal: ICE, STUN, TURN → Lecture 2
- Session descriptions: SDP
  - Announcing multicast sessions: SAP – *obsolete*
  - Control of streaming media: RTSP
  - Control of interactive conferencing: SIP
  - Control of telepresence: CLUE – *not widely used*
  - Control of web-based interactive media: JSEP (WebRTC)

# Media Transport: RTP

- Separate data and control channels
  - RTP – media payload formats
  - RTP Control Protocol (RTCP)
    - Source description and caller identity, reception quality, codec control
- Payload formats
  - Codec-specific packet formats; application level framing; robust, but complex
  - Each frame packetised for independent use for low latency
- Datagram TLS handshake – usual TLS handshake but within UDP packets
- Extensions
  - Reception quality and user experience monitoring
  - Codec control and other feedback
  - Circuit breakers and congestion control



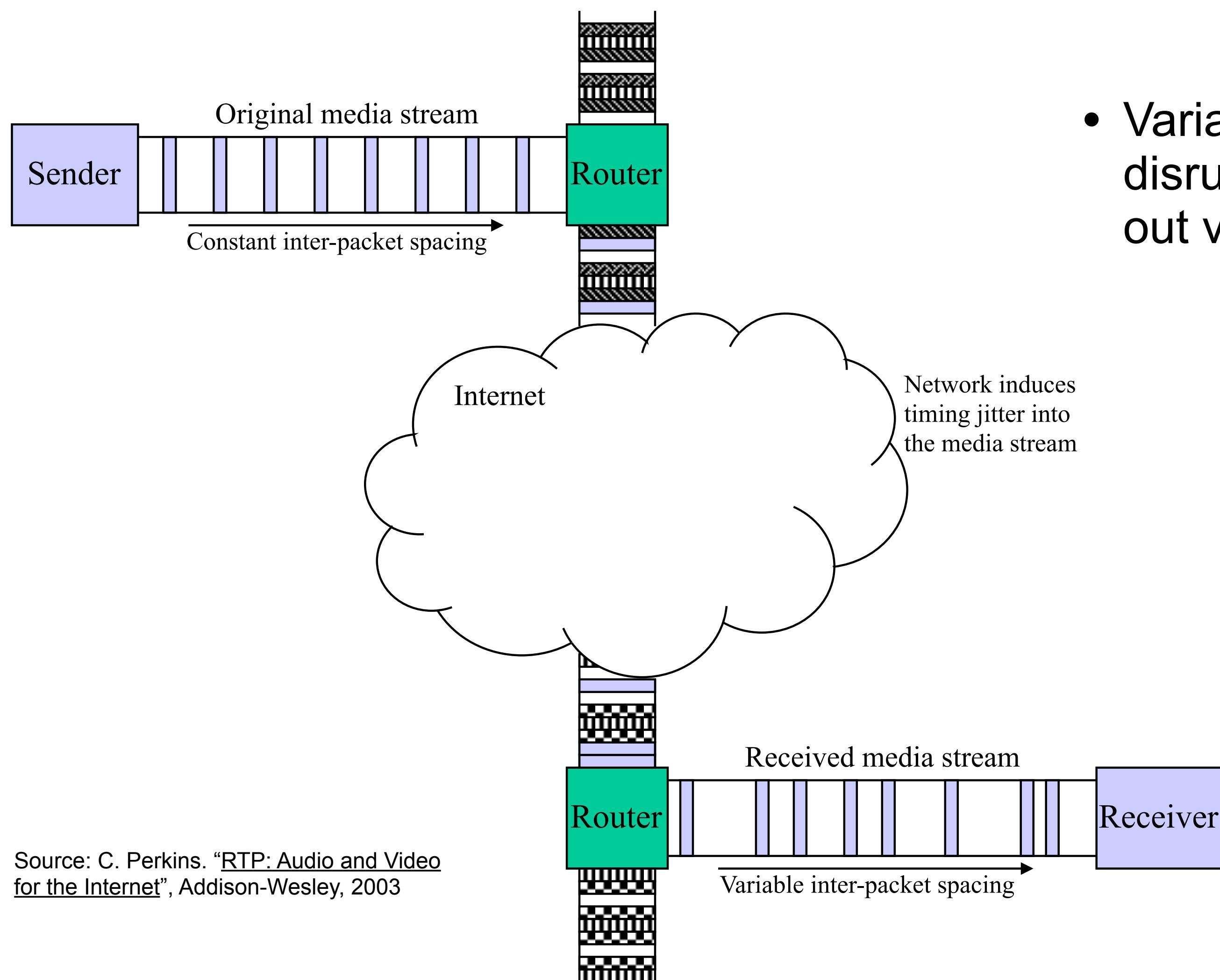
# Media Transport: RTP



- RTP data packets carried within UDP
- Header information carries:
  - Sequence number and timestamp
    - Allows receiver to reconstruct ordering and timing
  - Source identifiers
    - Who sent this packet – needed for multiparty calls
  - Payload format identifier
    - Does the packet contain audio or video?
    - What compression algorithm is used?

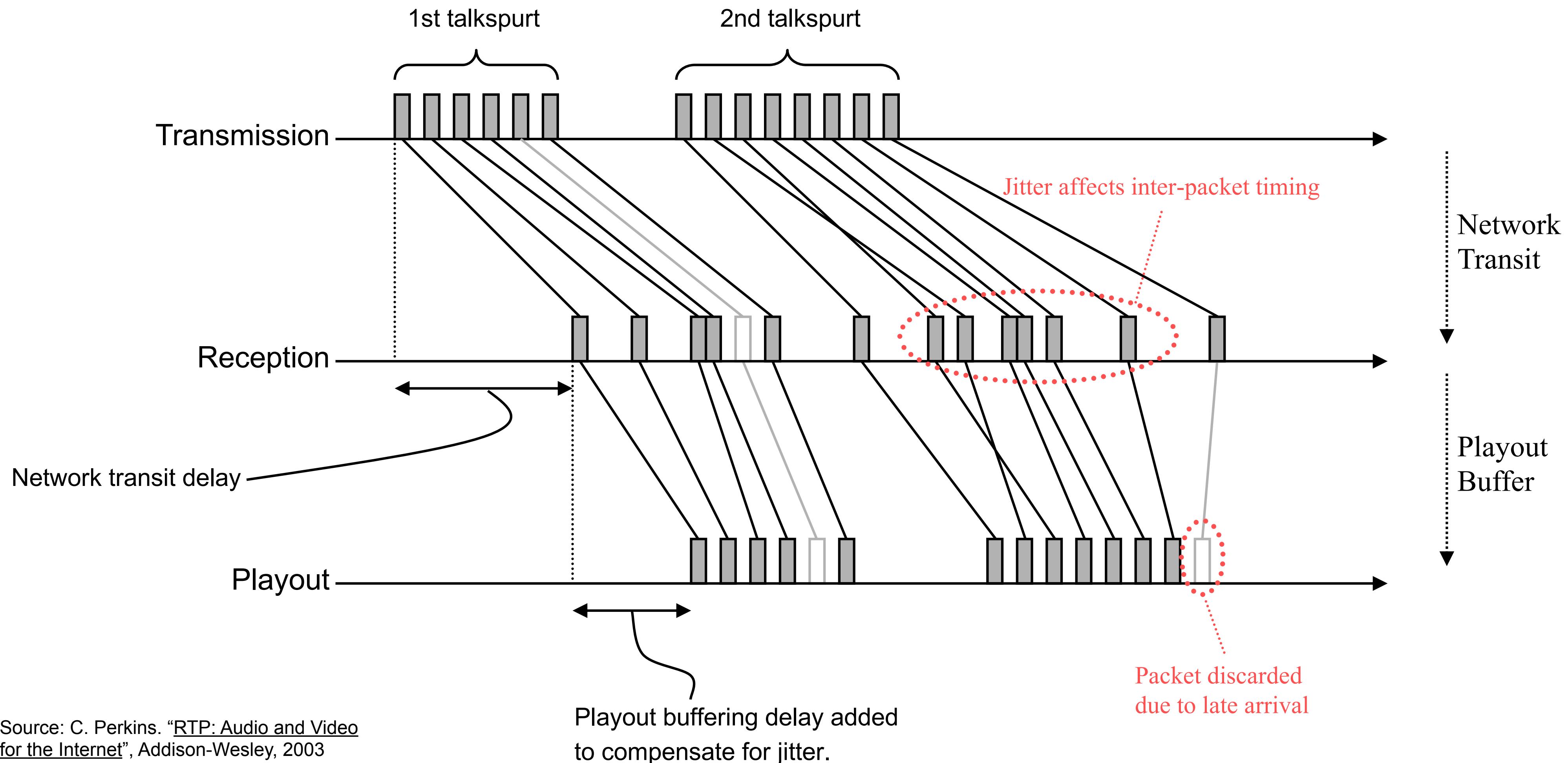
<https://datatracker.ietf.org/doc/rfc3550/>

# Media Transport: Timing Recovery (1/3)



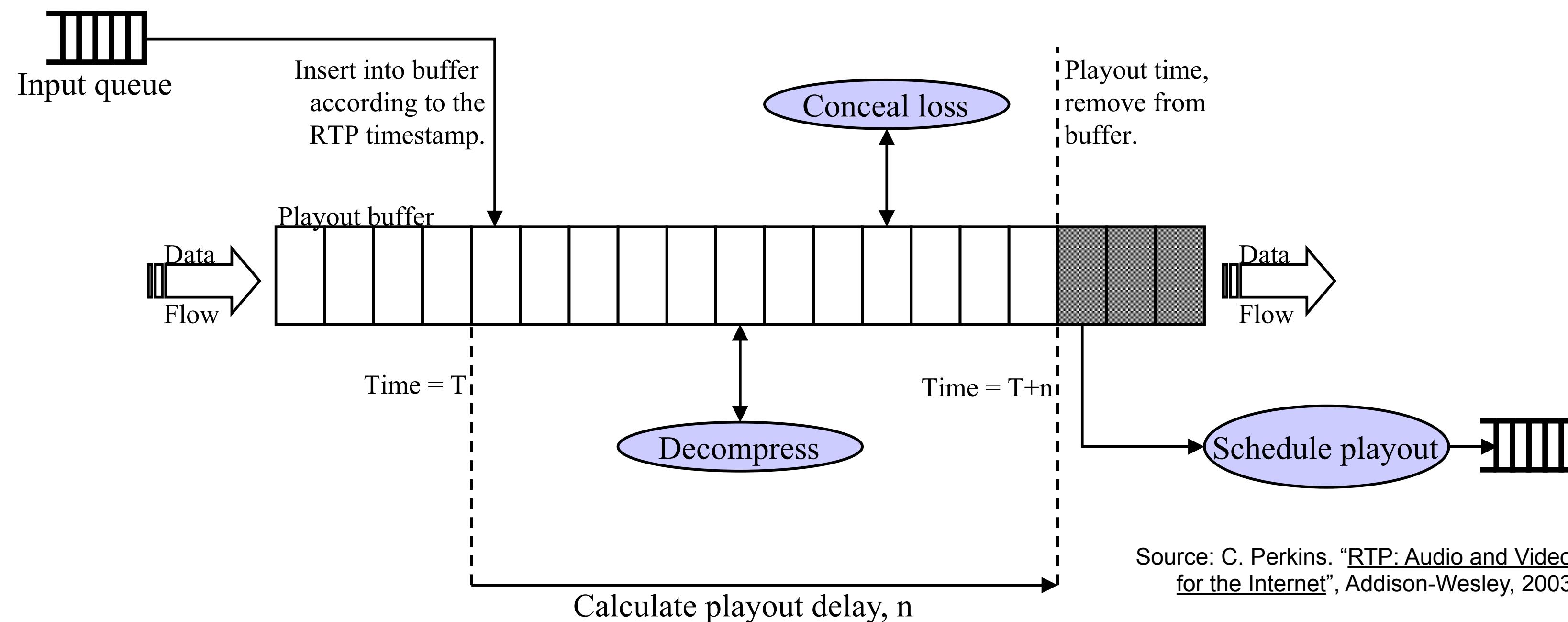
- Variable queueing delays in the network disrupt timing – receiver buffers to smooth out variation

# Media Transport: Timing Recovery (2/3)



# Media Transport: Timing Recovery (3/3)

- If packets played out immediately on arrival, variation in timing leads to gaps
- Delay playout by more than typical variation in inter-arrival time, to allow smooth play back



# Media Transport: Application Level Framing

- Packet loss is possible, so receivers must make best use of packets that do arrive
- RTP **payload formats** define how compressed audio/visual data is formatted into RTP packets
  - Goal: **each packet should be independently usable**
  - If a packet arrives, it should be possible to decode all the data it contains – not always possible, but desirable
  - Naïve packetisation can lead to inter-packet dependencies where a packet arrives but can't be decoded because some previous packet, on which it depends, was lost

Architectural Considerations for a New Generation of Protocols  
David D. Clark and David L. Tennenhouse  
*Laboratory for Computer Science, M. I. T.*

**Abstract**

The current generation of protocol architectures, such as TCP/IP or the ISO suite, seem successful at meeting the demands of today's networks. However, a number of new requirements have been proposed for the networks of tomorrow, and some changes in protocol structures may be necessary. In this paper, we review some key requirements for tomorrow's networks, and propose some architectural principles to structure a new generation of protocols. In particular, this paper identifies two new design principles, Application Level Framing and Integrated Layer Processing. Additionally, it identifies the presentation layer as a key aspect of overall protocol performance.

**1 Introduction**

Historically, there has been a sharp distinction between the network architectures used in universal access environments, such as the telephone network, and the architectures developed within the computer communications community. We believe that both communities would benefit from a unified architecture that addresses new communication requirements. We have identified three key requirements that will stress the existing protocol architectures. Future networks will have considerably greater capacity, will be based on a wide selection of technologies, and will support a broader range of services.

The principle design goal for many research projects, and some commercial products, is gigabit operation, both in aggregate over trunks and to individual end points. As networks proceed to higher speeds, there is some concern that existing protocols will represent a bottleneck, and various alternatives have been proposed, such as outboard protocol processors [1, 11] and new protocol designs [2, 14, 12]. Since

\*This research was supported by the Defense Advanced Research Projects Agency, monitored by the National Aeronautics and Space Administration under contract No. NAG2-582, and by the National Science Foundation under grant NCR-8814187.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1990 ACM 089791-405-8/90/0009/0200...\$1.50

**2 Structuring Principles for Protocol Architectures**

Most discussions of protocol design refer to the ISO reference model [9], or some other model that uses layering to decompose the protocol into functional modules. There is, however, a peril in using only a layered model to structure protocols into components, which is that layering may not be the most effective modularity for implementation.

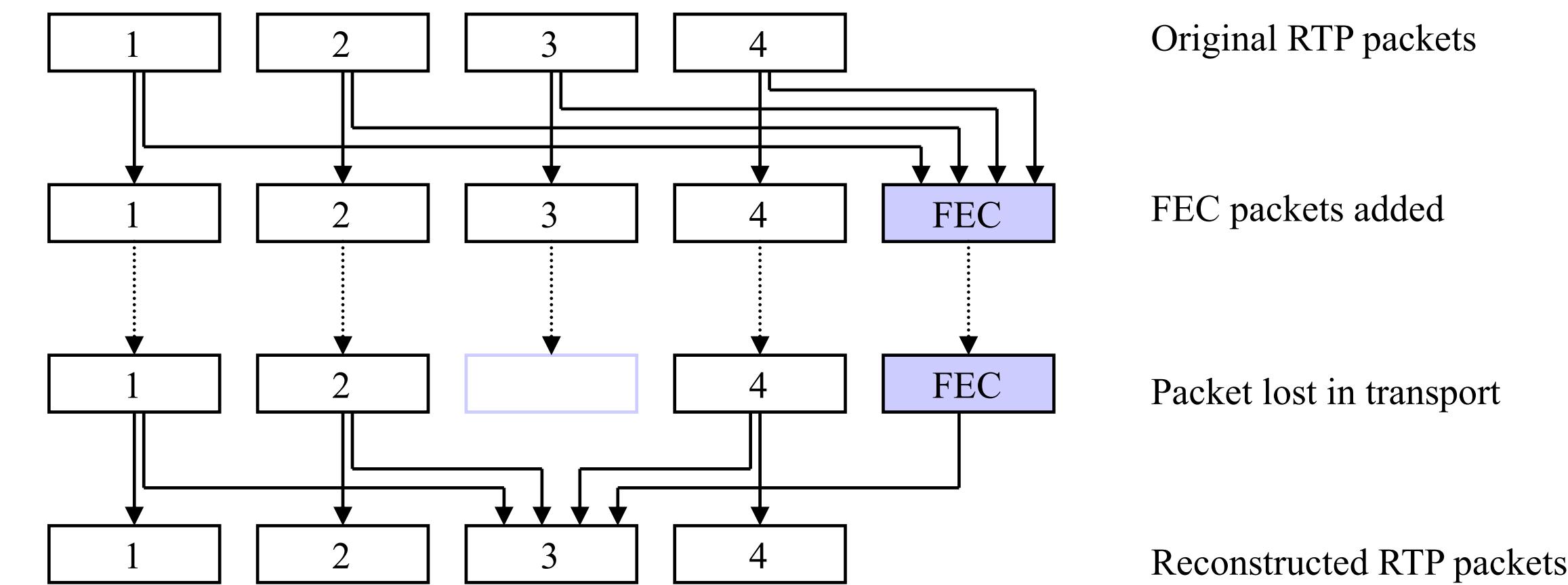
It is important to distinguish between the architecture of a protocol suite and the engineering of a specific end system or relay node. The architecture specifies the decomposition into functional modules, the semantics of the individual modules and, the syntax used to effect the protocol. There should be *a priori* requirements that the decomposition of a system design of a given system correspond to the architectural decomposition of the protocol. In the case of layered architectures, practical experience [4, 17] provides strong support for alternative engineering designs. Unfortunately, the structure of the protocol architecture may unac-

200

D. Clark and D. Tennenhouse, "Architectural considerations for a new generation of protocols", ACM SIGCOMM Conference, September 1990.  
<https://dx.doi.org/10.1145/99508.99553/>

# Media Transport: FEC vs. Retransmission

- Retransmission possible, but often takes too long – packet should have been played out before retransmission arrives
- Forward error correction (FEC) often used instead



- Additional FEC packets are sent along with the original data
  - Contain error correcting codes
  - e.g., the Exclusive-OR (XOR) of the original packets – many different FEC schemes
  - If some original packets are lost but the FEC packets arrive, original data can be reconstructed

# Interactive Applications

- Architecture for video conferencing
- Multimedia standards
- Media transport

# Interactive Applications

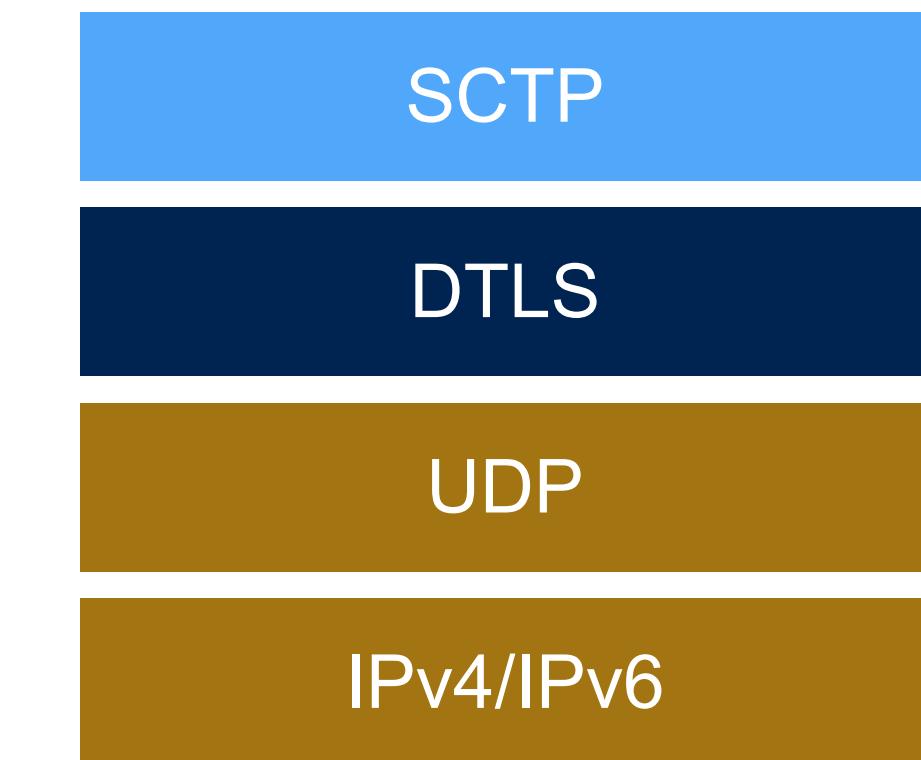
- Data Channel
- Signalling Protocols: SIP → WebRTC

# Media: WebRTC Data Channel

- In addition to audio-visual media, WebRTC provides a peer-to-peer **data channel**
  - For peer-to-peer file transfer
  - To support a chat session
  - To support reactions, raise hand, etc.

# Media: WebRTC Data Channel

- WebRTC data channel using SCTP in a secure UDP tunnel:
  - Transparent data channel
    - Message-oriented abstraction
    - Multiple sub-streams
    - Congestion controlled
  - Makes it straight-forward to build P2P applications with WebRTC
- Why not use QUIC? Because WebRTC pre-dates the development of QUIC
  - Future versions of WebRTC will likely migrate to using QUIC
  - QUIC is more flexible and more highly optimised



# Signalling and Session Descriptions

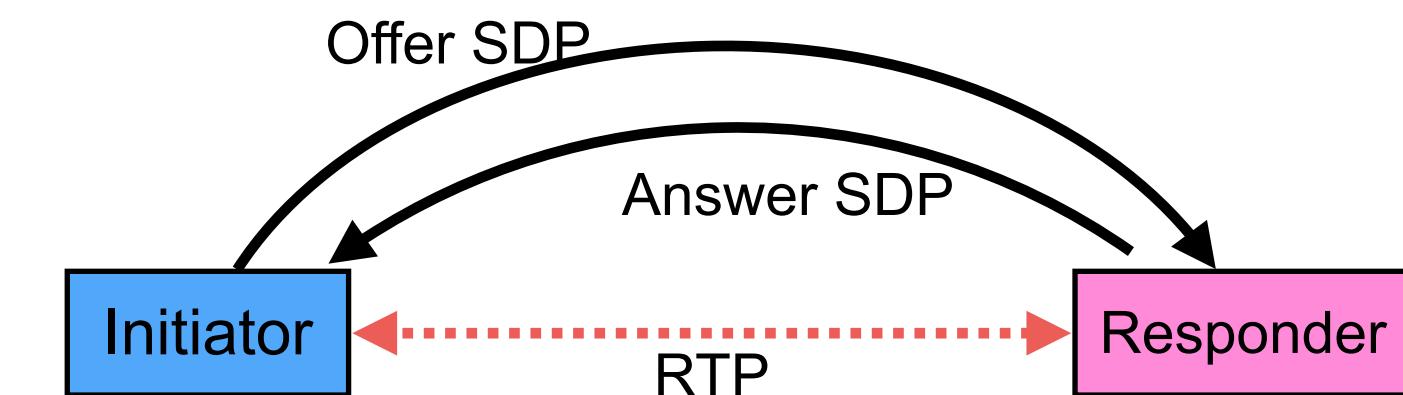
- Media transport flows peer-to-peer for low latency
- A **signalling protocol** is needed to find the peer and establish the paths
- The control protocol needs to describe the communication session expected:
  - Media transport connections required
  - Media formats and compression algorithms
  - IP addresses and ports to use
  - Originator and purpose of session
  - Options and parameters
- **Session description protocol (SDP)** provides a standard format for such data

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol”, IETF, RFC 4566, July 2006. <https://datatracker.ietf.org/doc/rfc4566/>

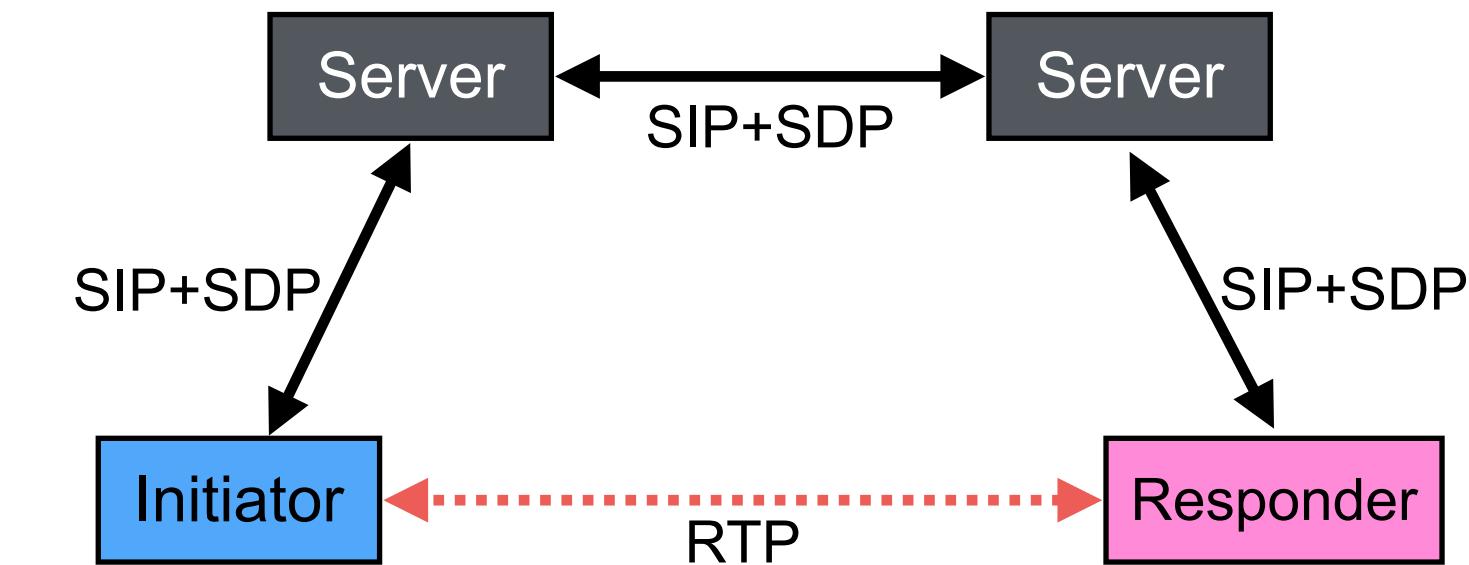
# Session Descriptions: SDP Offer/Answer

- Interactive sessions require negotiation
  - An **offer** to communicate: lists codecs, options and addressing details, identity of caller
  - The **answer** subsets codecs and options to those mutually acceptable, supplies addressing details, and confirms willingness to communicate
  - ICE algorithm (→ Lecture 2) probes NAT bindings, establishes path
  - Audio and video data flows
- SDP used as the negotiation format
  - SDP **was not** designed to express options and alternatives
  - Insufficient structure in syntax, semantic overloading
  - Complex → but complexity not initially visible; now too entrenched for alternatives to take off



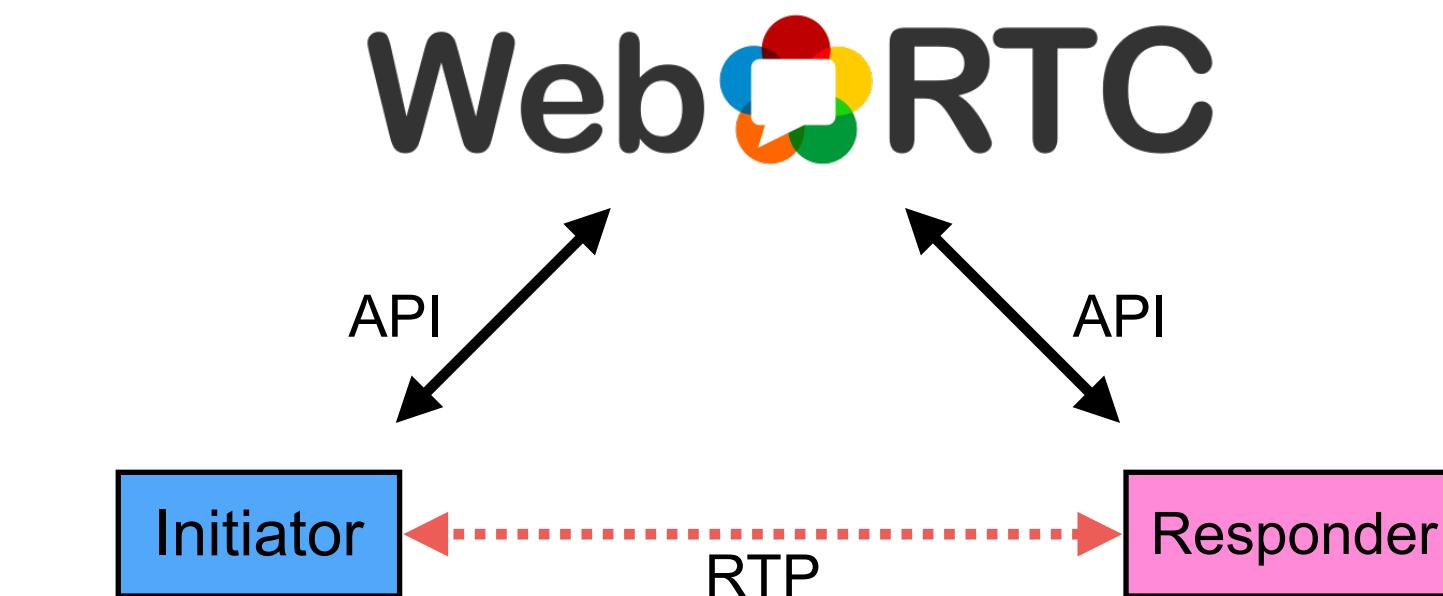
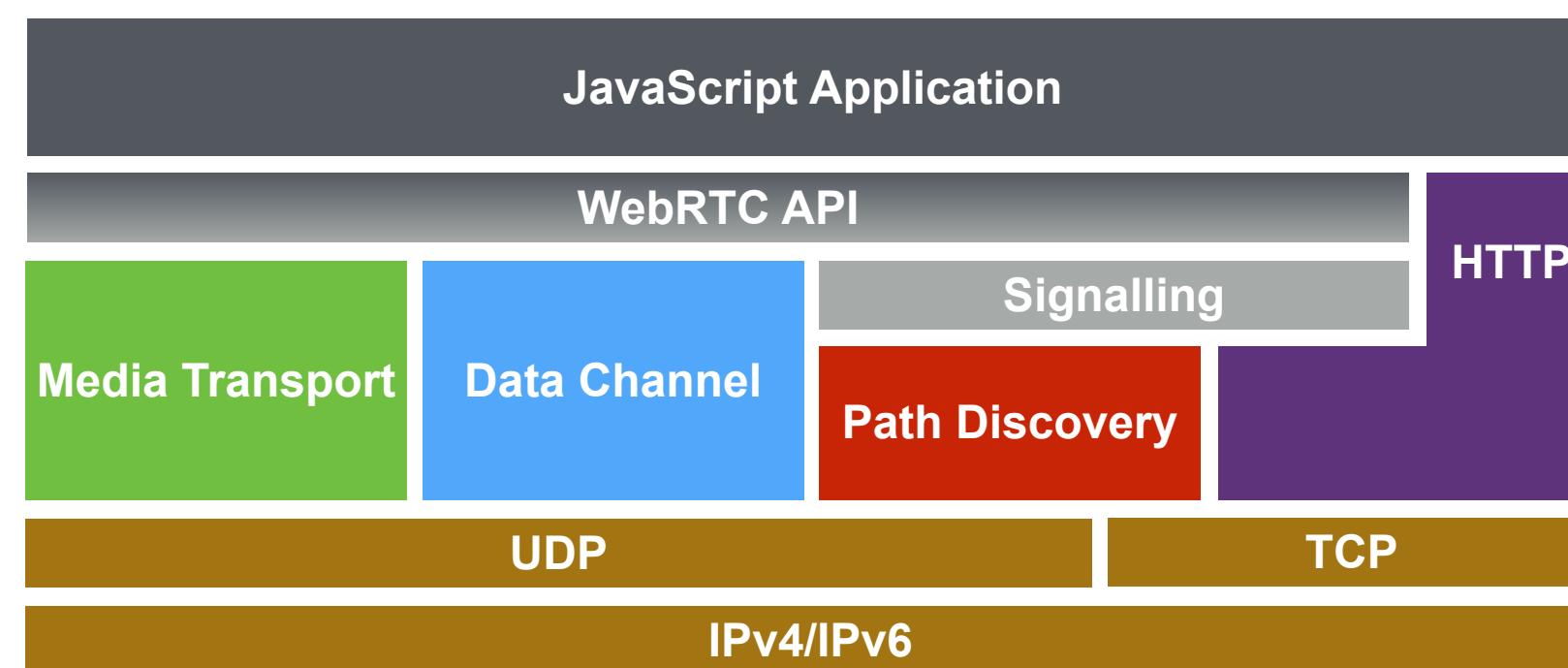
# Control of Interactive Conferencing: SIP

- The signalling protocol sets up the call
  - **Session Initiation Protocol (SIP)** for telephony and video conferencing
  - Conferencing servers with public IP addresses handle calls for a domain
  - SIP messages sent from initiator to responder via servers
    - Determine location of responder
    - Discover NAT bindings
    - Negotiate parameters and options for the call
    - An offer/answer exchange using SDP to describe the session being created
  - Alert the responding user – make phone ring! – and agree to setup a call
    - NAT binding discovery and connection probing takes place while alerting
  - Media then flows over direct peer-to-peer path



# Browser-based Conferencing: WebRTC

- WebRTC is an alternative signalling protocol
  - Implemented in web browsers
  - Exposes standard JavaScript API



- Signalling messages delivered via HTTP to web server controlling the call
  - Offer/answer exchanges using SDP
- Media transport uses RTP – same as SIP
- Adds peer-to-peer data channel
- Designed to integrate video conferencing into web browsers and web applications

# Future Directions for Interactive Applications

- New media types – holographic, tactile, augmented and virtual reality
  - Every stricter requirements on quality and latency
  - All can fit within the basic framework described
- Media over QUIC?
  - Active research and standardisation – expect rapid developments in this space

# Interactive Applications

- Architecture for video conferencing
- Signalling and media traffic
- SIP, SDP, RTP → WebRTC

# Streaming Video

- HTTP Adaptive Streaming

# Streaming Video Applications

- RTP **should** be usable for streaming video
  - Real-time Streaming Protocol (RTSP) developed by Real Networks for Internet TV in late 1990s
  - Uses unidirectional RTP for video delivery
  - Very low latency, loss tolerant
- **Most** video-on-demand, TV, and movies are delivered instead over HTTPS
  - Performance is significantly worse, very high latency, but startup latency is less critical
  - Integrates better with content distribution networks

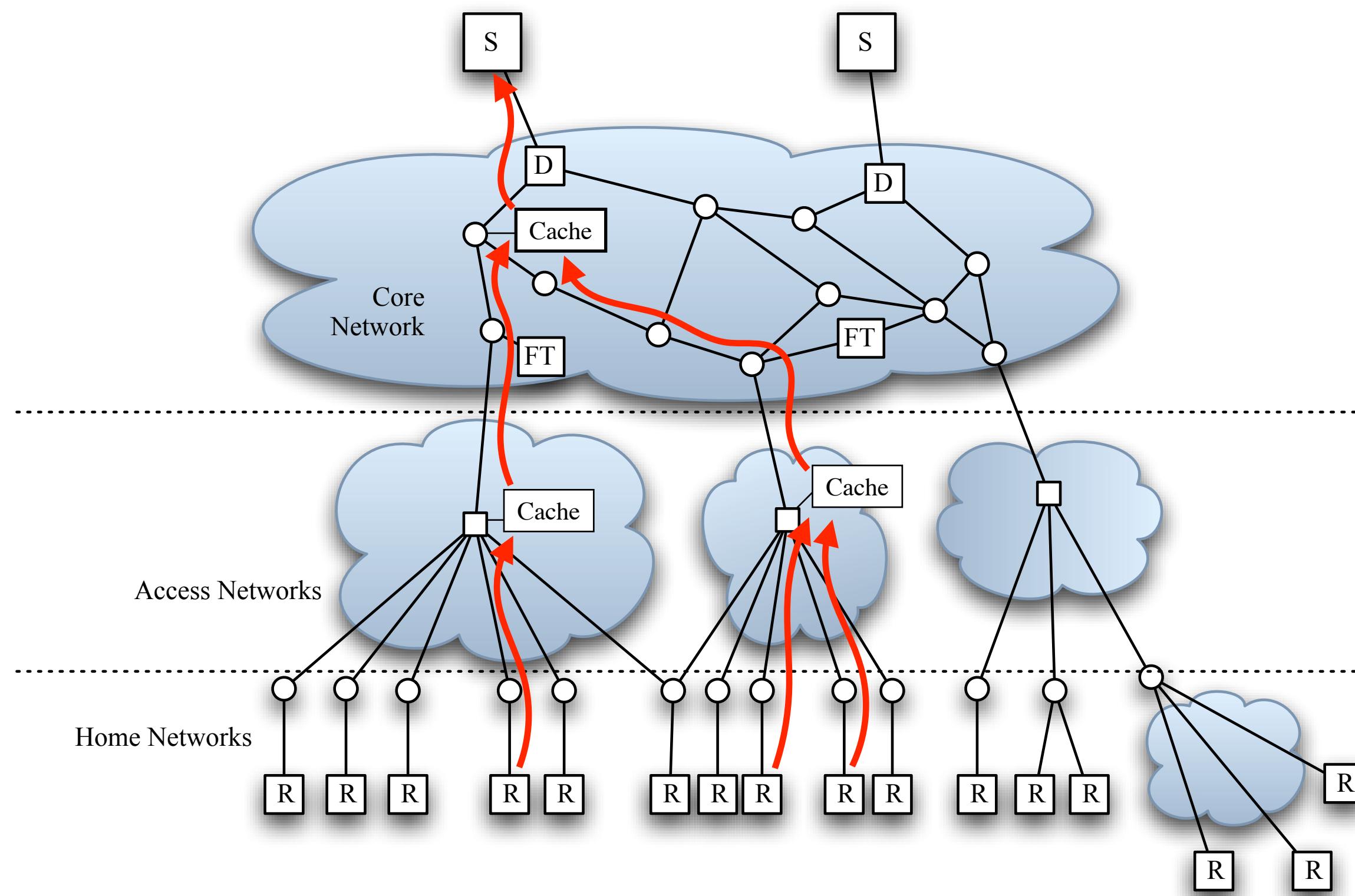


NETFLIX

BBC iPlayer

YouTube

# HTTP Content Distribution Networks

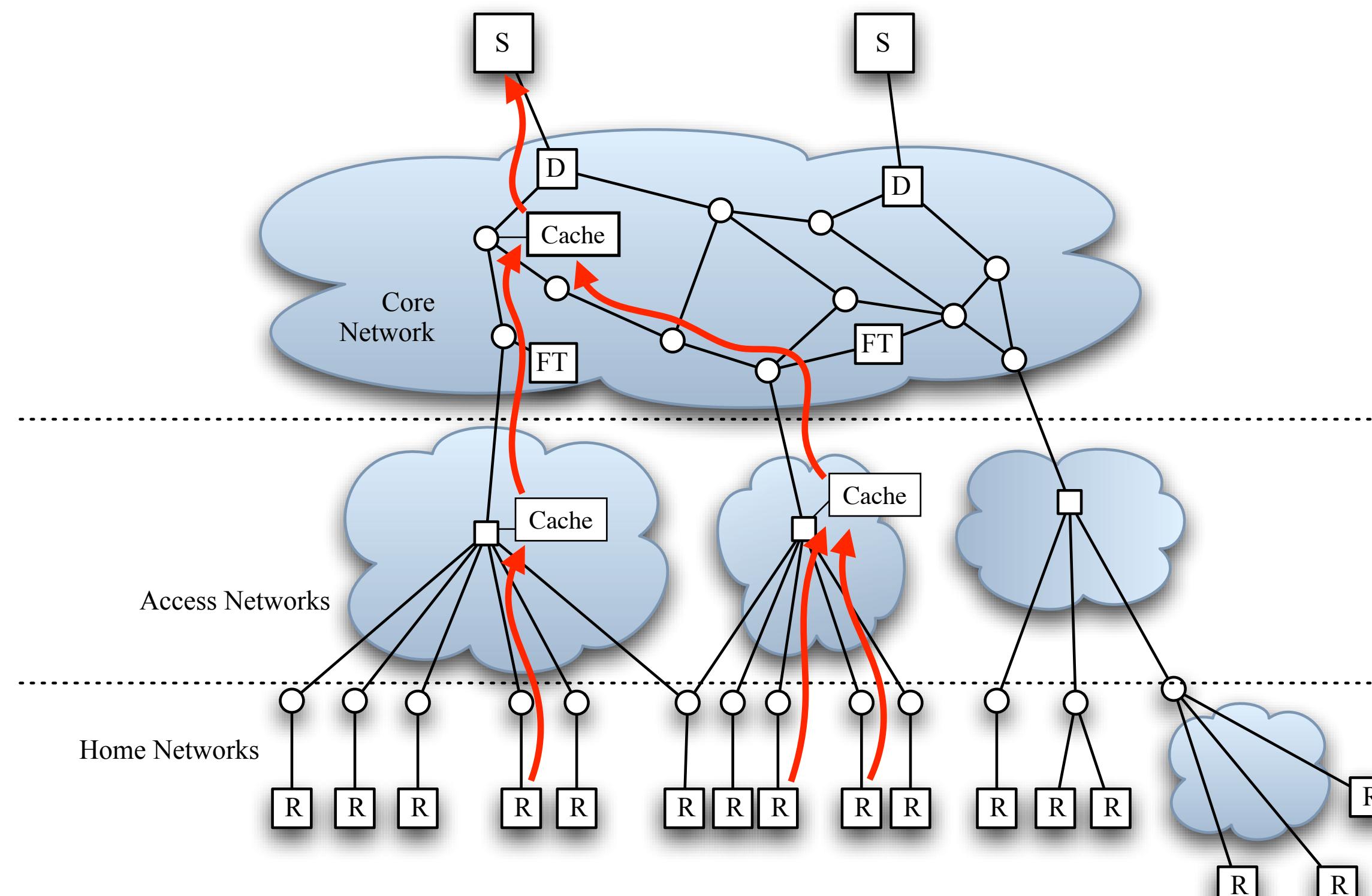


- There is an extensive, well-developed, content distribution infrastructure



- Highly effective for delivering and caching HTTP content
- Global deployment – agreements with most large ISPs to host caching proxy servers
- **Only works with HTTP-based content** – does not support RTP-based streaming

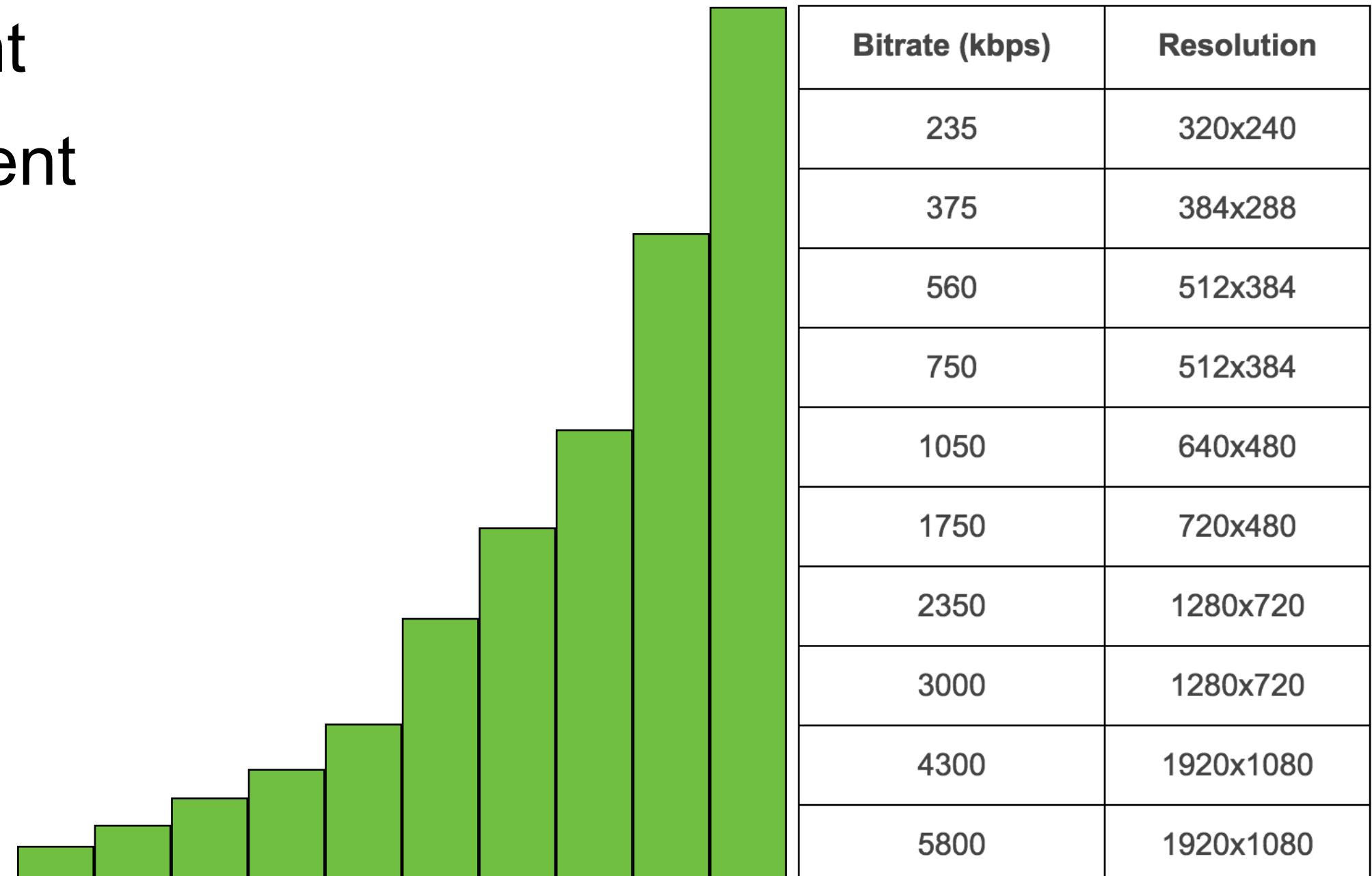
# HTTP Adaptive Streaming (a.k.a. MPEG DASH)



- **Deliver video over HTTPS** to make use of content distribution networks
  - Video content encoded in multiple **chunks**
    - Each chunk encodes ~10 seconds of video
    - Each chunk independently decodable
    - Each chunk made available in many different versions at different encoding rates
  - A **manifest** file provides an index for what chunks are available
  - Clients fetch manifest, download chunks in turn
    - Standard HTTPS downloads
    - But monitor download rate, and choose what encoding rate to fetch next – adaptive bitrate
  - Playout chunks in turn, as they download

# DASH: Chunked Media Delivery and Rate Adaptation

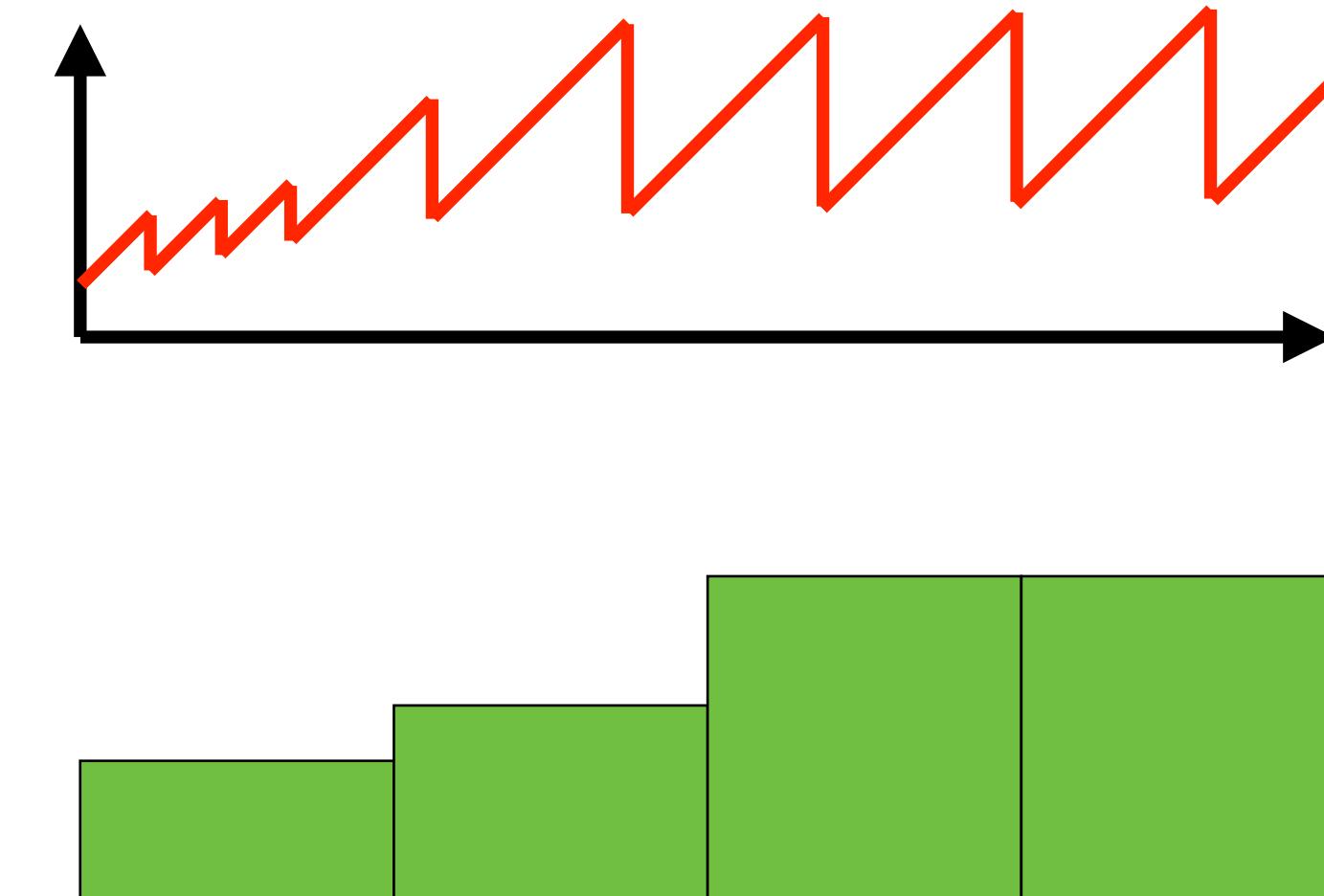
- Each chunk encodes ~10 seconds of video content
- Each chunk is compressed multiple times at different encoding rates (i.e., different sizes)
  - e.g., Netflix suggests encoding at ten different rates, as show on the right
- Receiver fetches manifest, containing the list of all versions of all each chunk
- Receiver fetches each chunk in turn
  - Measures download rate and compares to the encoding rate
  - If download rate slower than encoding rate, then switch to lower encoding rate for next chunk
  - If download rate faster than encoding rate, consider fetching higher rate for next chunk
- Chooses encoding rate to fetch based on TCP download rate



Source: Netflix

# DASH and TCP Congestion Control

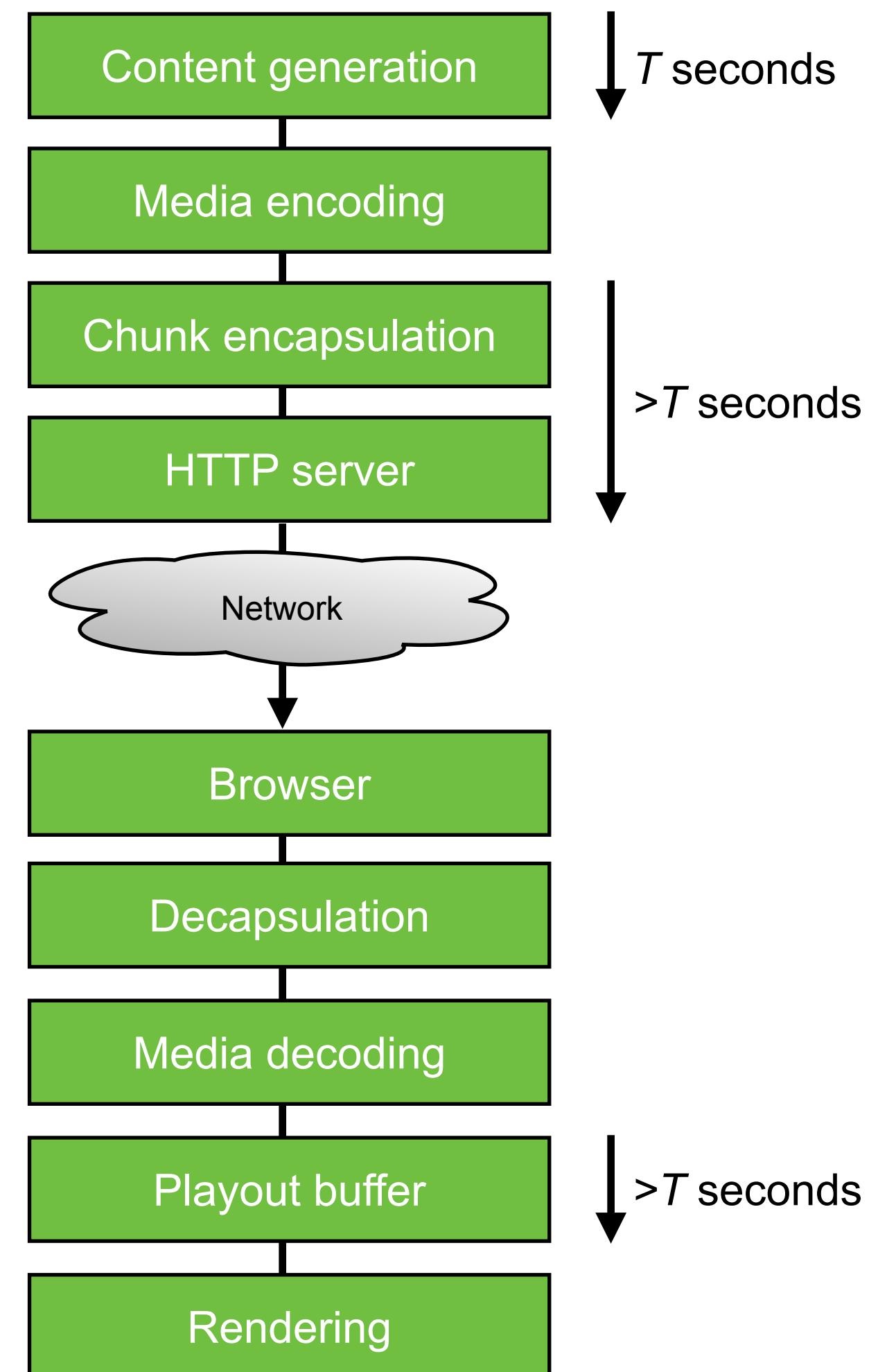
- DASH rate adaptation and TCP congestion control work at different time scales
  - TCP adjusts congestion window once per RTT
    - TCP Reno or Cubic
    - AIMD algorithm
  - DASH receiver measures **average download speed** of TCP connection over ~10 seconds
    - Selects size of next chunk to download based on average TCP download speed



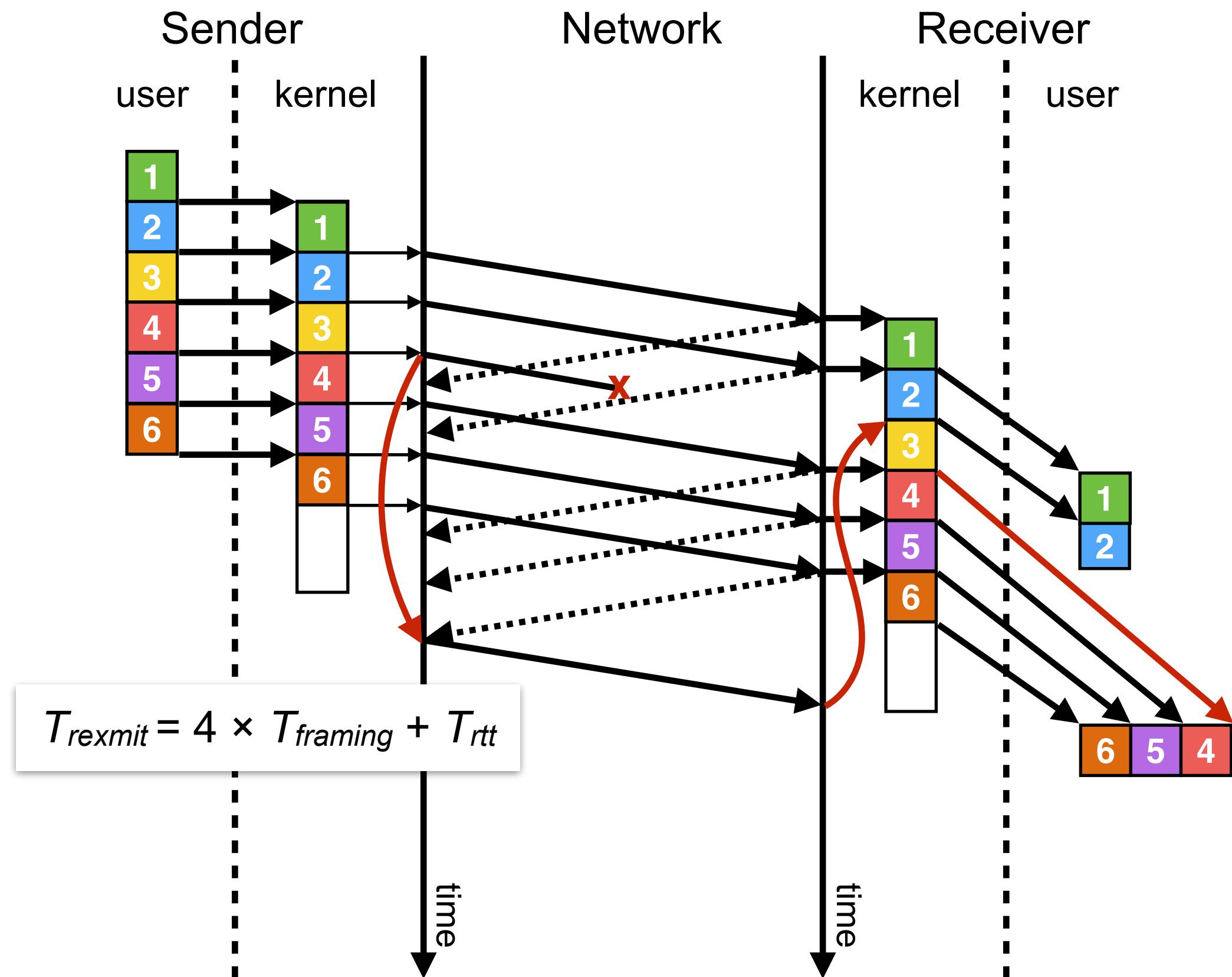
- Videos played using DASH often have relatively poor quality for first few seconds
  - Receiver picked a conservative download rate – relatively poor quality, small size – to start
  - Uses that to measure connection download speed, then switches to more appropriate rate

# DASH Latency

- Multi-second playout latency common, due to:
  - Chunk duration
  - Playout buffering
  - Encoding delays for live streaming
- **Chunk duration is a key** – must download complete chunk before starting playout
  - Download and decompress chunk  $n$
  - Start playing chunk  $n$  and while playing download chunk  $n+1$
  - No packets lost → latency equals chunk duration



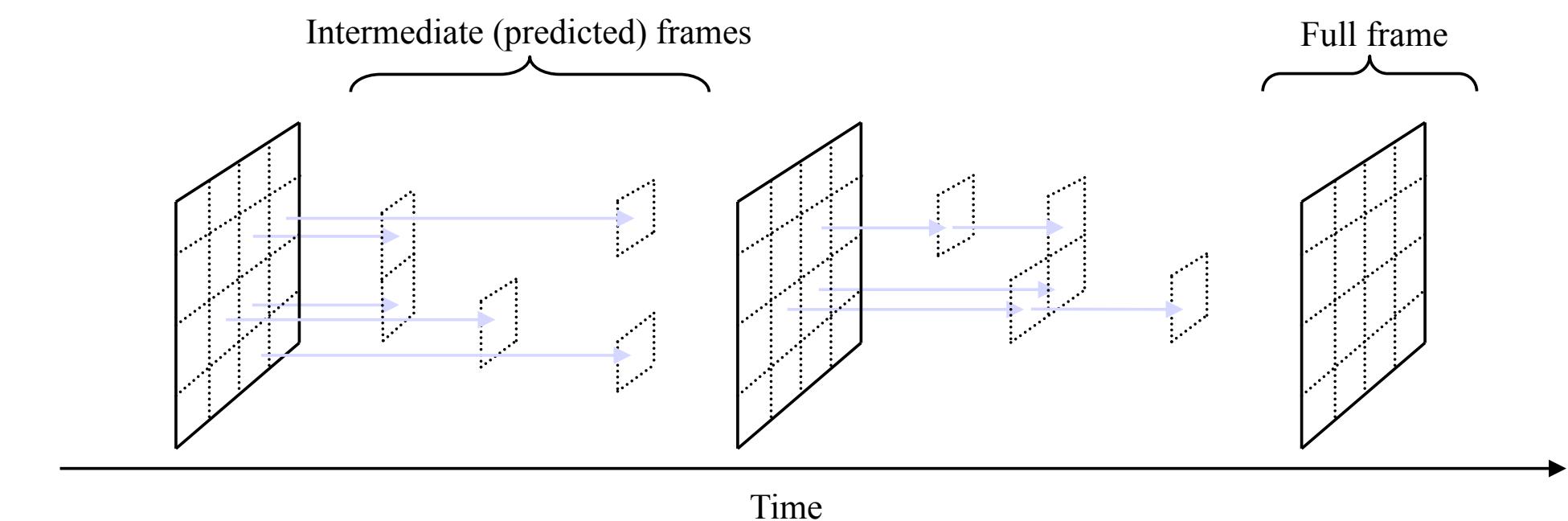
# Sources of Latency: TCP



- If a packet is lost, TCP retransmits after triple duplicate ACK
  - This takes time, and delays download of the chunk:
    - $4 \times$  packet transmission latency (the lost packet, plus three following that generate duplicate ACKs)
    - $1 \times$  network RTT to retransmit packet
  - Each packet loss and retransmission will increase chunk download time – and causes TCP to reduce its window
- Both add latency if using TCP

# Sources of Latency: Chunks and Video Compression

- Each chunk of video is independently decodable
  - Can't compress based on previous chunk, because previous chunk depends on the network capacity → unpredictable
  - Each chunk therefore starts with an I-frame, followed by P-frames
    - I-frames are large: often 20× size of P-frames
    - If chunks are smaller duration → contain fewer P-frames compared to number of I-frames; video compression ratio goes down
  - Trade-off between chunk size and compression overhead
    - Large chunks require less data, but add latency
    - Small chunks reduce latency, but need more data
    - Overheads become excessive for chunk duration  $\leq 2$  seconds



# Future Directions for Streaming Video

- Streaming video over WebRTC?
  - Web browsers now all support WebRTC
  - Intended for interactive conferencing, but entirely usable for RTP-based video streaming to browser
  - Much **lower latency** than DASH
  - Will this encourage CDNs to support RTP?
- Streaming video over QUIC?
  - HTTP/3 is a drop-in replacement for existing HTTP stack, and delivers requests over QUIC
  - YouTube already delivers video over QUIC this way – expect to see much more deployment and special purpose QUIC extensions for video



# Real-time and Interactive Applications

- Real-time traffic and its constraints
- Interactive applications → WebRTC
- Streaming applications → DASH
- Custom infrastructure for low latency; CDN and HTTP if latency less critical



University  
ofGlasgow

# Naming and the Tussle for Control

Networked Systems (H)  
Lecture 8



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Lecture Outline

- What is the DNS and DNS resolution
- DNS naming structure and organisation
- Methods for DNS resolution
- The politics of names
- Associated reading: “Tussle in cyberspace: Defining tomorrow’s Internet” discusses the issues of control over the network, and how protocol design influences this – the DNS is one area where we see this tussle

## Tussle in Cyberspace: Defining Tomorrow’s Internet

David D. Clark  
MIT Lab for Computer Science  
ddc@lcs.mit.edu  
  
Karen R. Sollins  
MIT Lab for Computer Science  
sollins@lcs.mit.edu

John Wroclawski  
MIT Lab for Computer Science  
jtw@lcs.mit.edu  
  
Robert Braden  
USC Information Sciences Institute  
braden@isi.edu

### Abstract

The architecture of the Internet is based on a number of principles, including the self-describing datagram packet, the end to end arguments, diversity in technology and global addressing. As the Internet has moved from a research curiosity to a recognized component of mainstream society, new requirements have emerged that suggest new design principles, and perhaps suggest that we jettison some old ones. This paper explores one important reality that surrounds the Internet today: different stakeholders that are part of the Internet milieu have interests that may be adverse to each other. These entities each vie to favor their particular interests. We call this process “the tussle”. Our position is that accommodating this tussle is crucial to the evolution of the network’s technical architecture. We discuss some examples of tussle, and offer some technical design principles that take it into account.

**Categories and Subject Descriptors**  
C.2.1 [Computer Systems Organization]: Computer Communications Networks—Network Architecture and Design; H.1 [Information Systems]: Models and Principles; K.4.1 [Computing Milieux]: Computers and Society; Public Policy Issues

**General Terms**  
Design, Economics, Legal Aspects, Security, Standardization

**Keywords**  
Tussle, Network Architecture, Trust, Economics, Design Principles, Competition

Work sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-02-0553 at MIT, and agreement number F30602-02-0554 at USC. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon.  
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post to servers or to redistribute to lists, requires prior specific permission and/or a fee.  
*SIGCOMM’02*, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.  
Copyright 2002 ACM 1-58113-570-X/02/0008 \$5.00.

### 1. INTRODUCTION

The Internet was created in simpler times. Its creators and early users shared a common goal—they wanted to build a network infrastructure to hook all the computers in the world together so that as yet unknown applications could be invented to run there. All the players, whether designers, users or operators, shared a consistent vision and a common sense of purpose.

Perhaps the most important consequence of the Internet’s success is that the common purpose that launched and nurtured it no longer prevails. There are, and have been for some time, important and powerful players that make up the Internet milieu with interests directly at odds with each other.

Some examples are very current. Music lovers of a certain bent want to exchange recordings with each other, but the rights holders want to stop them. People want to talk in private, and the government wants to tap their conversations. Some examples are so obvious that they are almost overlooked. For the Internet to provide universal interconnection, ISPs must interconnect, but ISPs are sometimes fierce competitors. It is not at all clear what interests are being served, to whose advantage, to what degree, when ISPs negotiate terms of connection. It is not a single happy family of people dedicated to universal packet carriage.

We suggest that this development imposes new requirements on the Internet’s technical architecture. These new requirements, in turn, motivate new design strategies to accommodate the growing tussle among and between different Internet players. The purpose of this paper is to explore what these requirements and strategies might be.

We begin by briefly discussing the Internet landscape – some fundamental differences between the mechanisms of engineering and society, and the players that populate our field. We then outline some proposed design principles intended to accommodate within the Internet mechanisms of society as well as those of engineering. We believe this accommodation is central to designing an Internet that is resilient to the challenges of society as well as those of technology. We conclude by discussing some tussle spaces, ways in which our principles might guide the technical response to these spaces, and specific technical research that may be of value in accommodating these tussles.

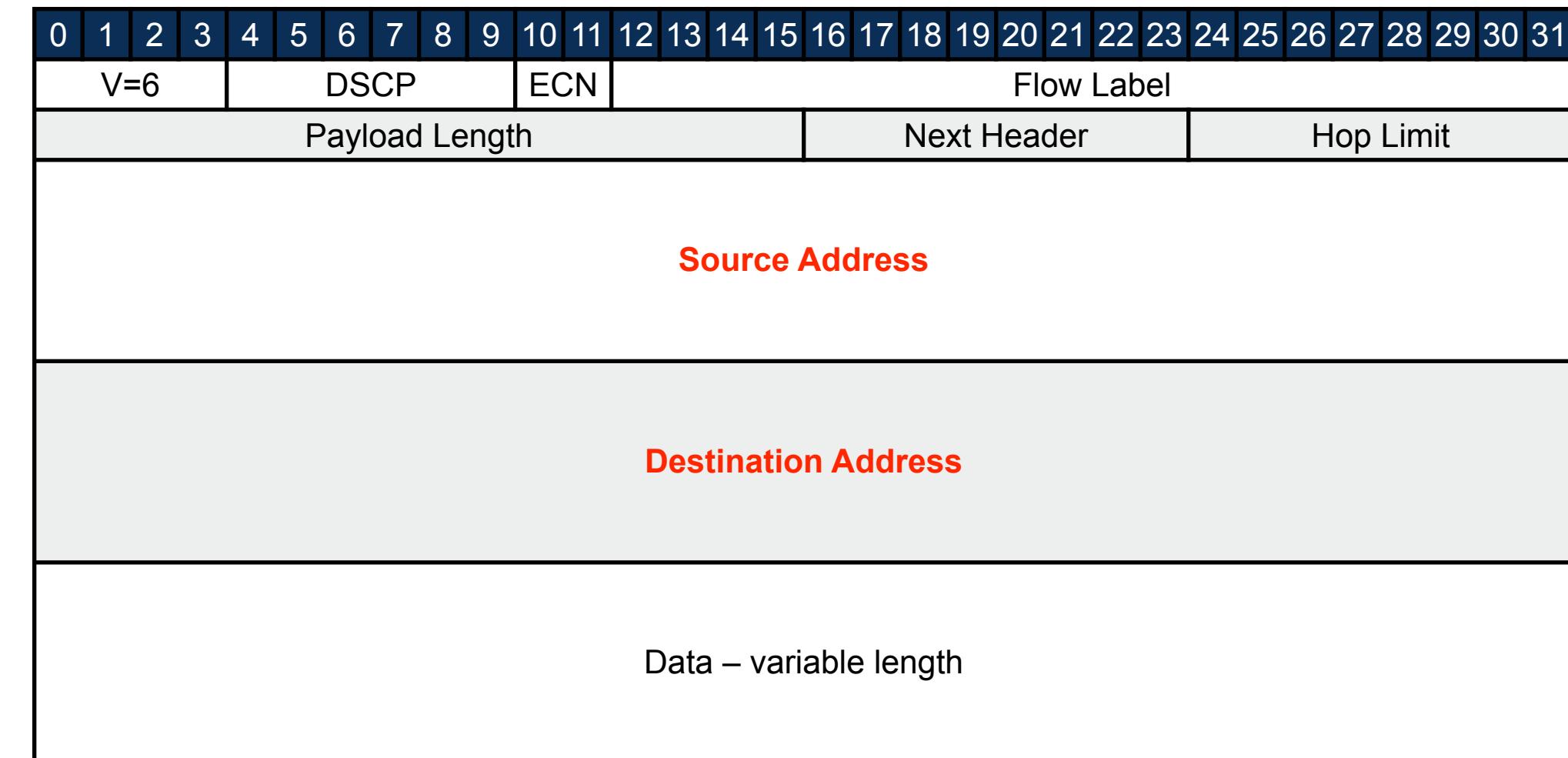
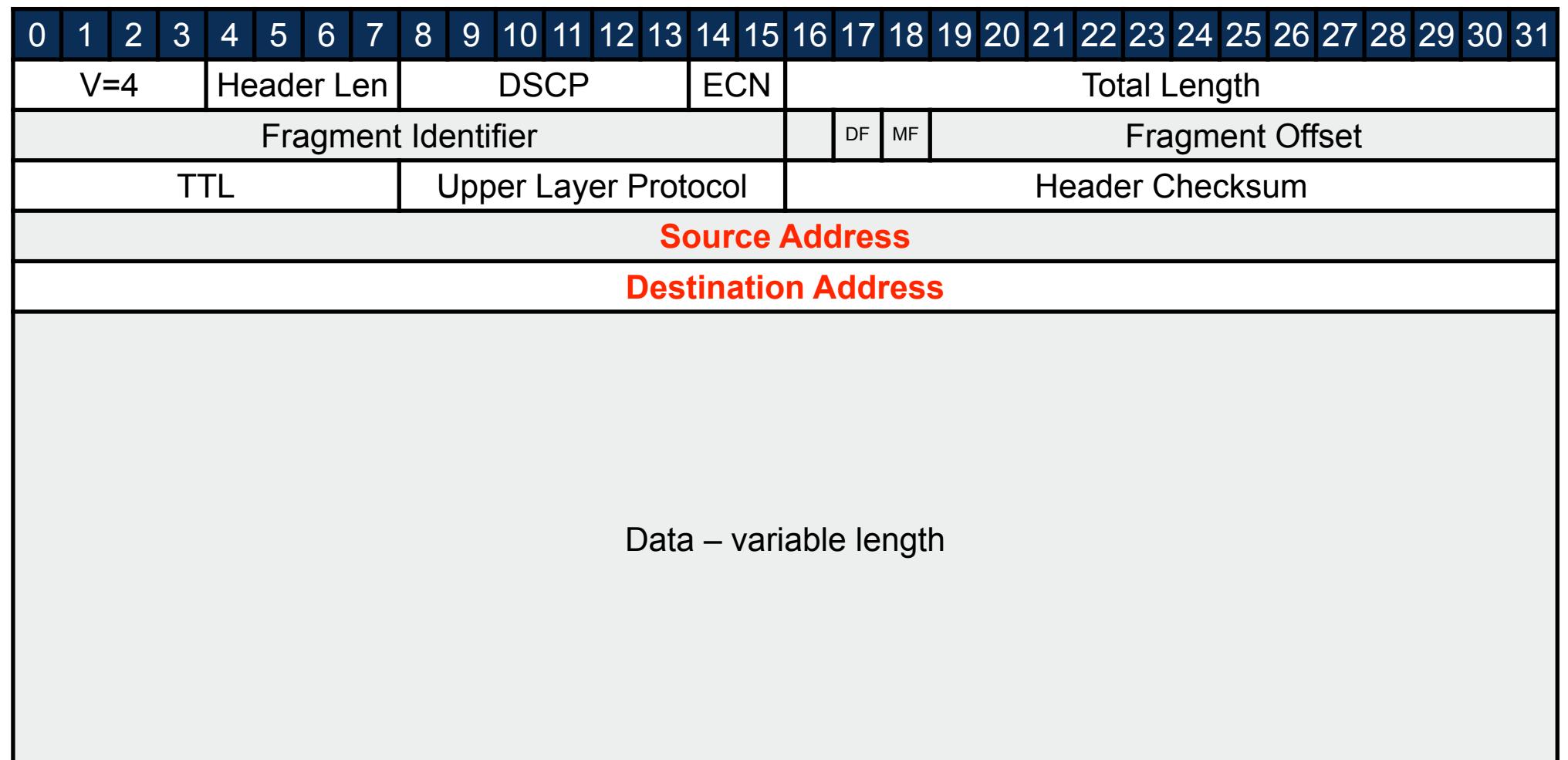
**1.1 The natures of engineering and society**  
Engineers attempt to solve problems by designing mech-

D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden,  
“Tussle in cyberspace: Defining tomorrow’s Internet”,  
ACM SIGCOMM Conference, August 2002.  
<https://dx.doi.org/10.1145/633025.633059>

# DNS Name Resolution

- What is the DNS?
- Structure of names
- Name resolution

# What is the DNS? (1/2)



- IP packets contain addresses rather than names
  - Designed for efficient processing by routers determining where to forward the packet
  - Not human readable – people prefer names, not addresses
  - Domain name system (DNS) is a distributed database; maps names to IP addresses

# What is the DNS? (2/2)

<https://www.csperkins.org/teaching/>

**Uniform Resource Locator (URL)**

www.csperkins.org

**Domain Name**

DNS

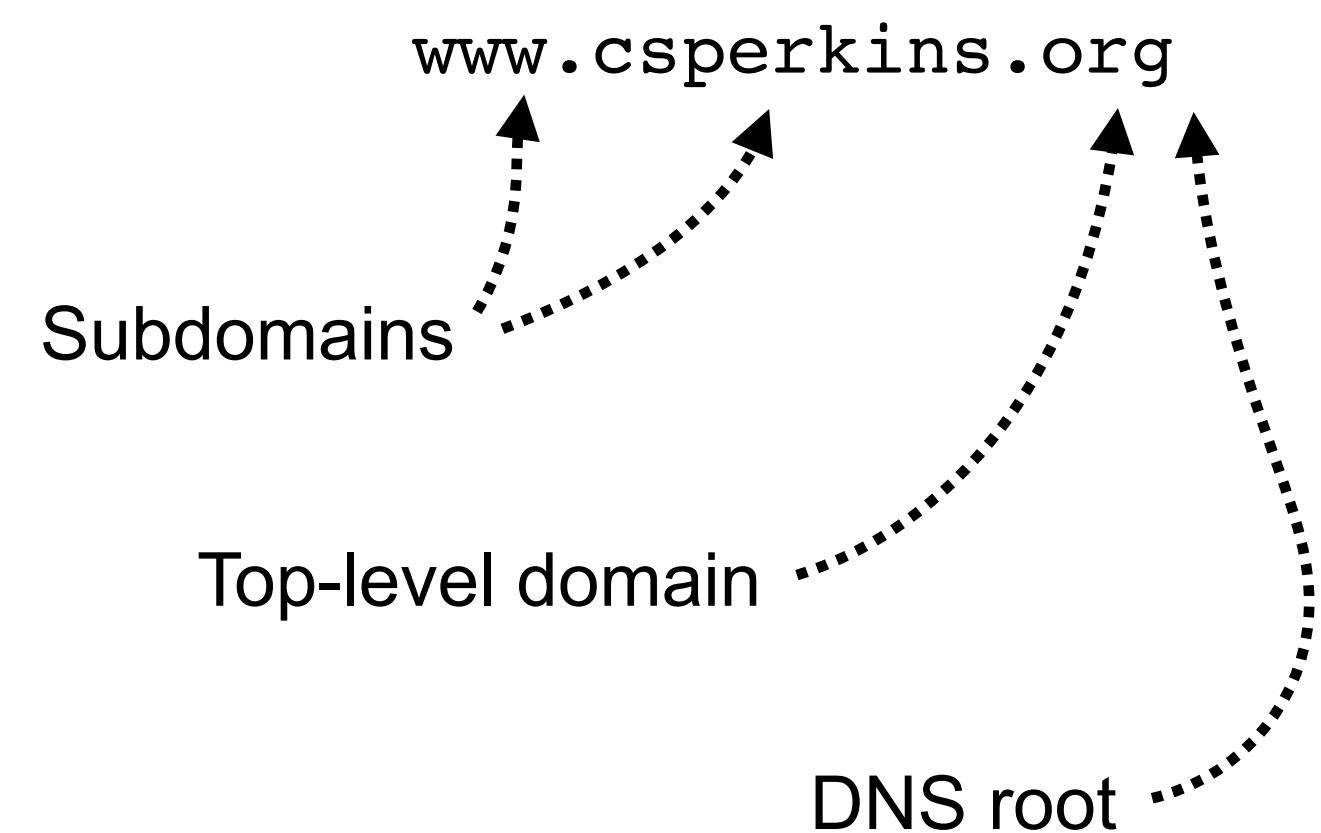
People and applications

Internet routing and forwarding

IPv6 2a00:1098:0:86:1000::10  
IPv4 93.93.131.127

**IP Addresses**

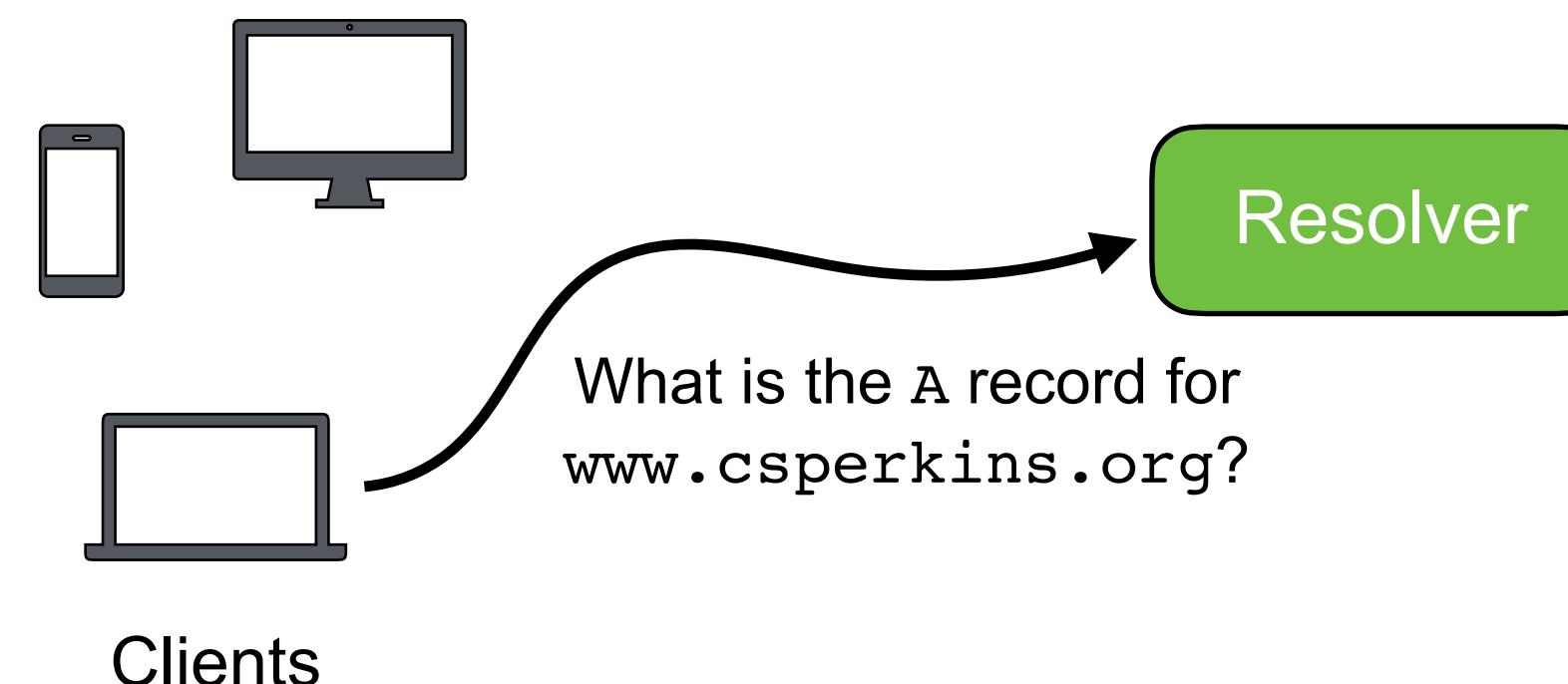
# Structure of DNS Names



- Naming is hierarchical
  - Sub-domains first
  - Concludes with a **top-level domain (TLD)**
    - Country-code top-level domains (ccTLDs)
      - .uk, .de, .cn, .io, .ly, ...
    - Generic top-level domains (gTLDs)
      - .com, .org, .net, ...
  - Top-level domains live within the **DNS root**
    - The **root servers** advertise the top-level domains
    - They have well-known, fixed, IP addresses – new DNS resolvers need to reach them to find the TLDs before they can answer DNS queries
- Each level is independently administered and operated
  - DNS is a distributed database – in authority and implementation
  - Each level in the hierarchy controls its own data

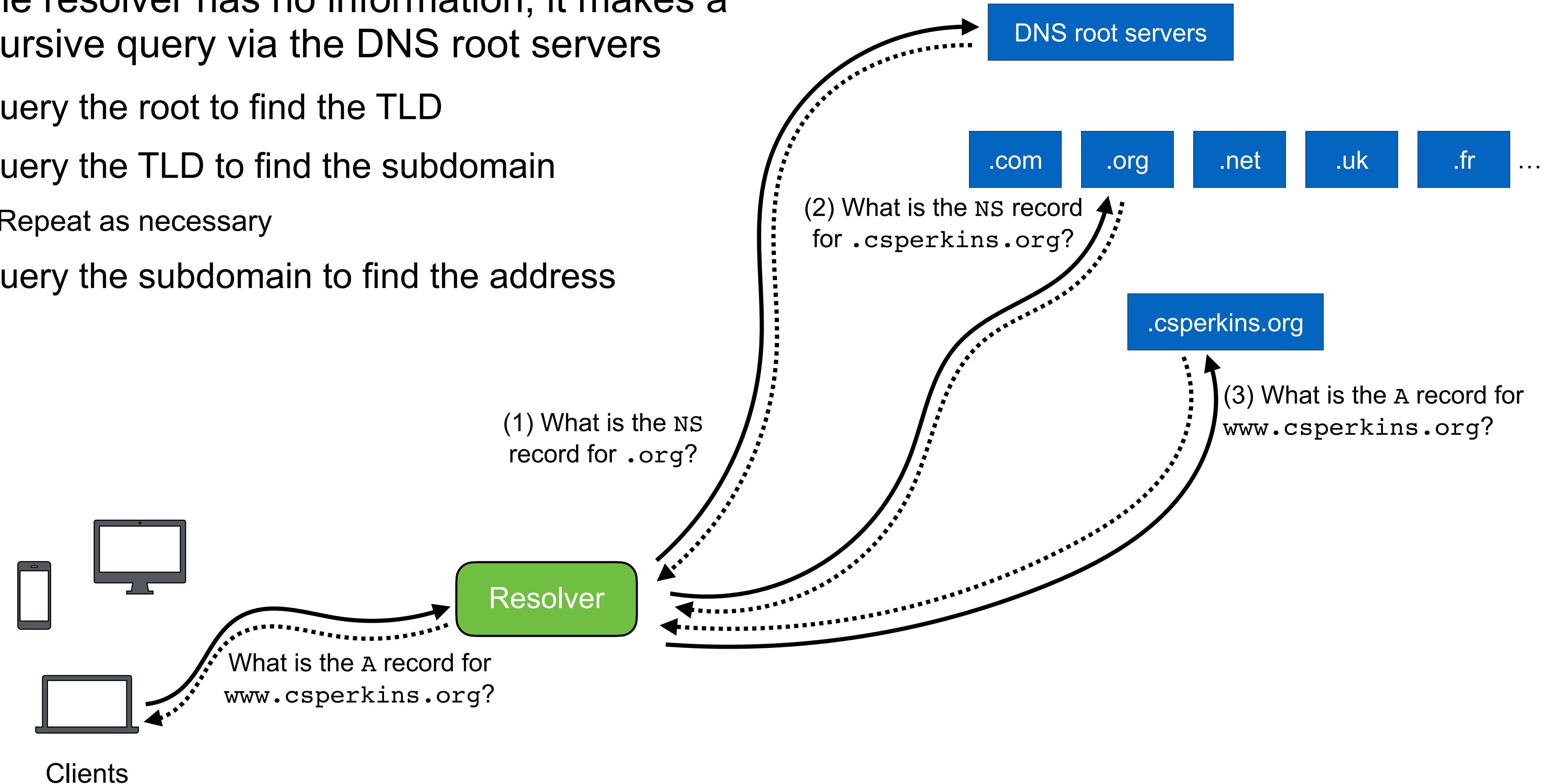
# DNS Name Resolution (1/3)

- The DNS is used for **name resolution**
  - Given a name, lookup a particular type of **record** relating to that name
  - Many different types of record: A, AAAA, CNAME, MX, NS, SRV, ...
  - Most common are A and AAAA records, that map hostnames to IPv4 and IPv6 addresses, and NS records that identify the **name server** for a domain
- A DNS **client** asks its **resolver** to perform the lookup by calling `getaddrinfo()`
  - The resolver could be a process running on the client, it more commonly runs on a machine provided by the network operator



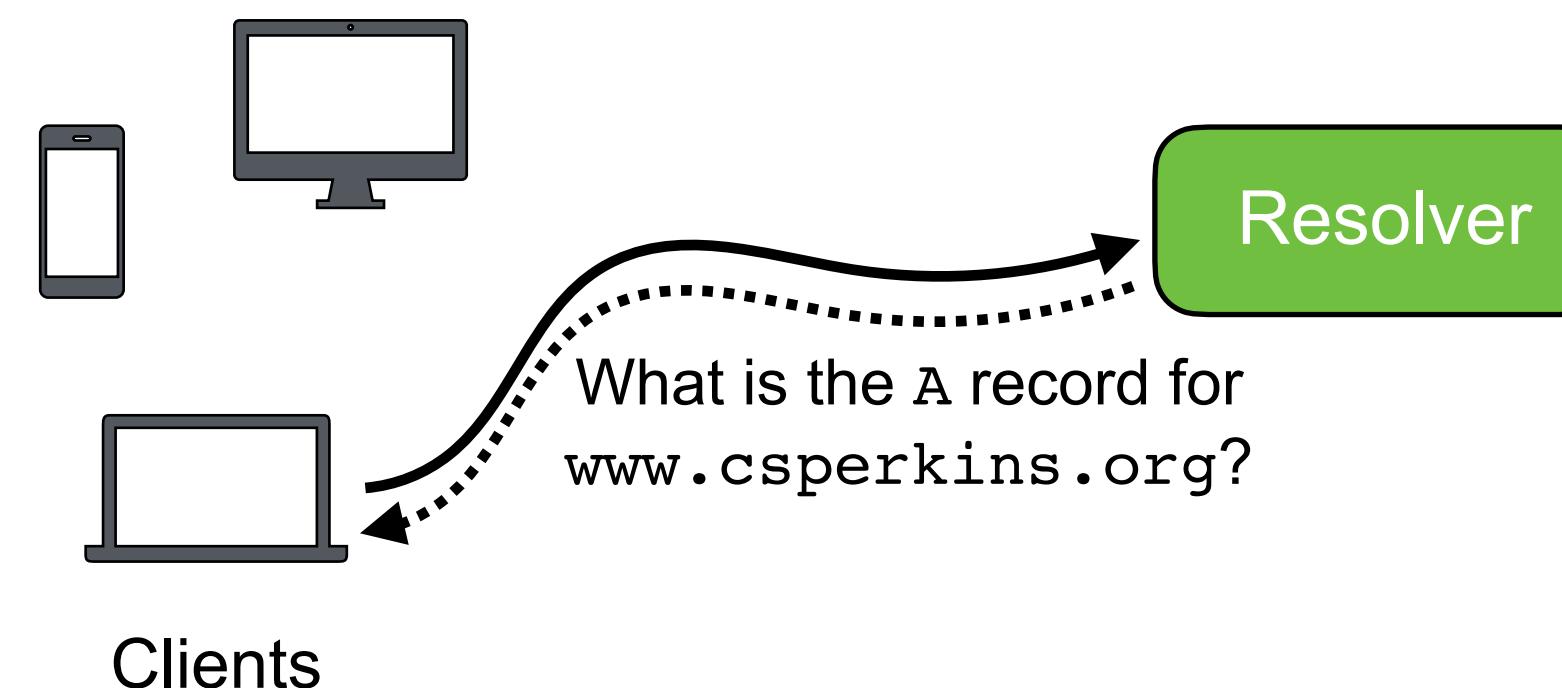
# DNS Name Resolution (2/3)

- If the resolver has no information, it makes a recursive query via the DNS root servers
  - Query the root to find the TLD
  - Query the TLD to find the subdomain
    - Repeat as necessary
  - Query the subdomain to find the address



# DNS Name Resolution (3/3)

- Responses include a **time to live** that allows a resolver to cache the value for a certain time period
  - Subdomains can set a short TTL and give different answers each time a particular name is requested → load balancing; CDNs directing queries to local caches
  - Subsequent queries are answered from the cache, where possible
    - If the cached entry times it, it's refreshed from the next level up in the hierarchy
    - Eventually reaching the DNS root
  - The IP addresses for the root servers are well known, and never time out



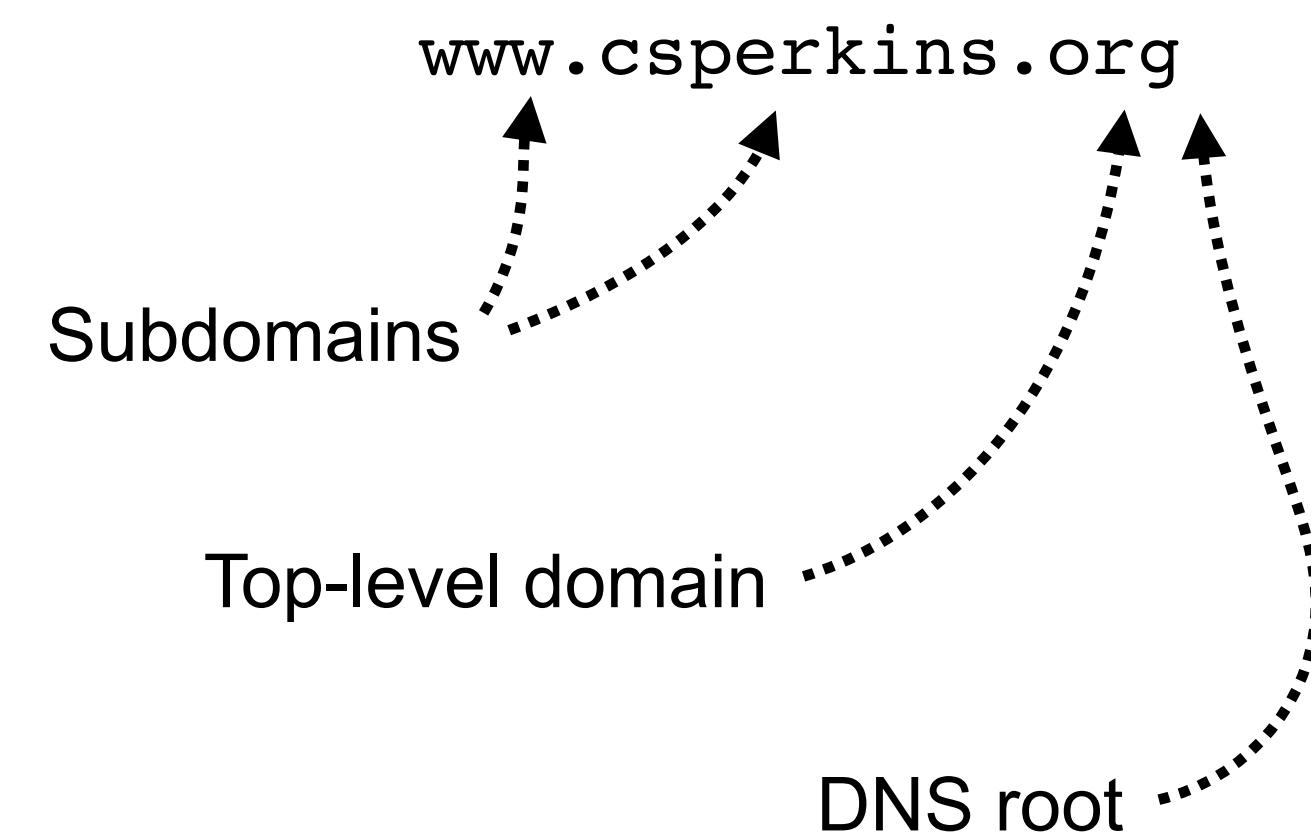
# DNS Name Resolution

- What is the DNS?
- Structure of names
- Name resolution

# DNS Names

- Who controls the DNS?
- Top-level domains
- Internationalised DNS
- The DNS root

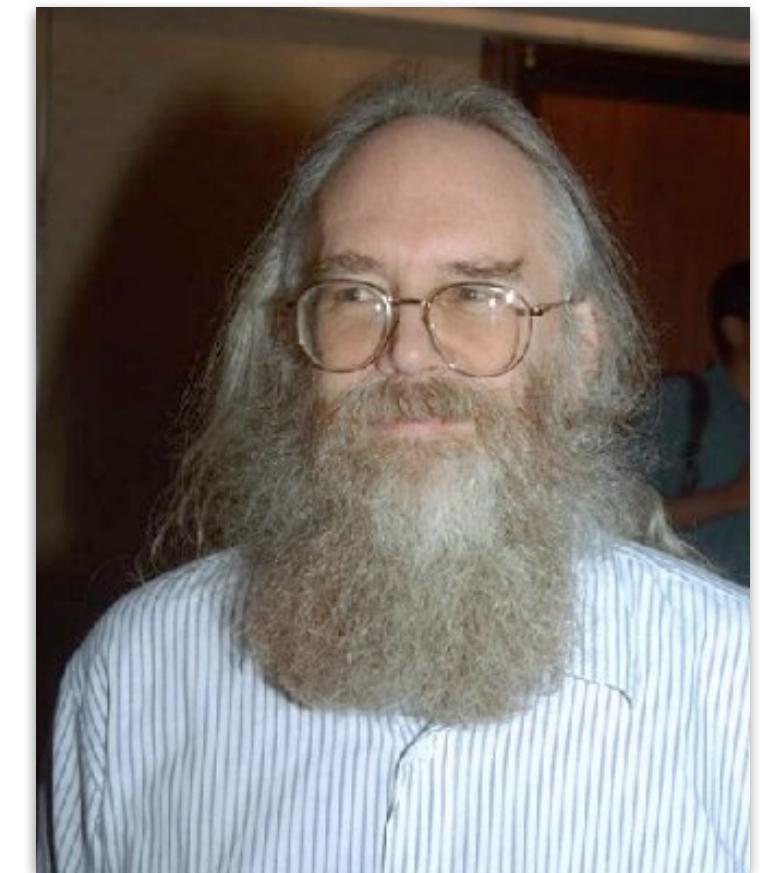
# Structure of DNS Names



- DNS names are assigned in a hierarchy – subdomains delegated from a top-level domain delegated from the DNS root
  - What top-level domains exist?
  - What policy does each top-level domain have for allocating sub-domains?
  - Who decides when to add new top-level domains?
  - Who controls the DNS root?

# From IANA to ICANN (1/2)

- The set of top-level domains is controlled by the Internet Corporation for Assigned Names and Numbers (ICANN)
- ICANN has a complex history
  - ARPANET project – US government funded research, 1966-1990
    - Developed initial versions of Internet protocols, TCP/IP, etc.
    - ARPANET needed protocol specifications and a parameter registry – Jon Postel volunteered as RFC Editor and IANA, originally as a graduate student at UCLA, later at USC/ISI
  - Postel handled domain name allocation, as IANA, funded by US Government
  - Informal, part-time, role – gradually formalised in the late 1990s, leading to formation of ICANN
  - <http://www.ietf.org/rfc/rfc2468.txt> – “I remember IANA”
- ICANN formed, incorporating IANA, September 1998
  - Dedicated organisation to manage domain names in the public interest, as a global multi-stakeholder forum, as the Internet started to become widely deployed and commercialised
  - A US not-for-profit corporation based in Los Angeles
  - US Government contractual control of ICANN ended on October 2016



Jon Postel

# From IANA to ICANN (2/2)

- ICANN has a complex governance model:
  - Board of governors; generic names supporting organisation – *gTLDs*; country code names supporting organisation – *ccTLDs*; address supporting organisation – *regional Internet registries*; governmental advisory committee – *representatives from each of the 112 UN-recognised countries*; at-large advisory committee; root server advisory committee; security and stability advisory committee; and technical liaison group
  - Regular public meetings, 3x per year; annual budget ~\$140M
- ICANN is **political** – many countries and organisations want to influence how domain names are managed and allocated



# Top-Level Domains (1/5)

- There are four types of top-level domain:
  - Country code top-level domains (ccTLDs)
  - Generic top-level domains (gTLDs)
  - Infrastructure top-level domains
  - Special-use top-level domains

# Top-Level Domains (2/5)

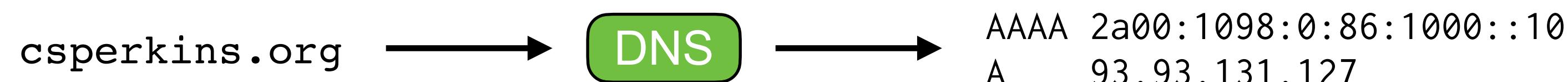
- DNS includes Country Code Top-Level Domains (ccTLDs)
  - ISO standard 3166-1 defines two-letter country names
    - Member states of the United Nations, UN special agencies (ITU, IMF, UNESCO, WHO, ...), parties to the International Court of Justice
    - Every code included in ISO 3166-1 is added to the DNS root zone
      - .uk, .fr, .de, .cn, .us, .io, .ly, ...
        - Each country has its own policy for sub-domains of the CCTLD
        - .cs – JANET Name Resolution Systems and the “Czechoslovakia problem” (historical)
      - Exceptions:
        - .gb – The UK should use .gb to match ISO 3166-1 (.gov.uk used to be .hmg.gb, but .gb never widely used)
        - .su – The domain for the Soviet Union still exists and accepts new registrations
        - .eu – The EU is not an ISO 3166-1 country, but has a ccTLD
        - .oz – Australia, sadly, changed from .oz to .au

# Top-Level Domains (3/5)

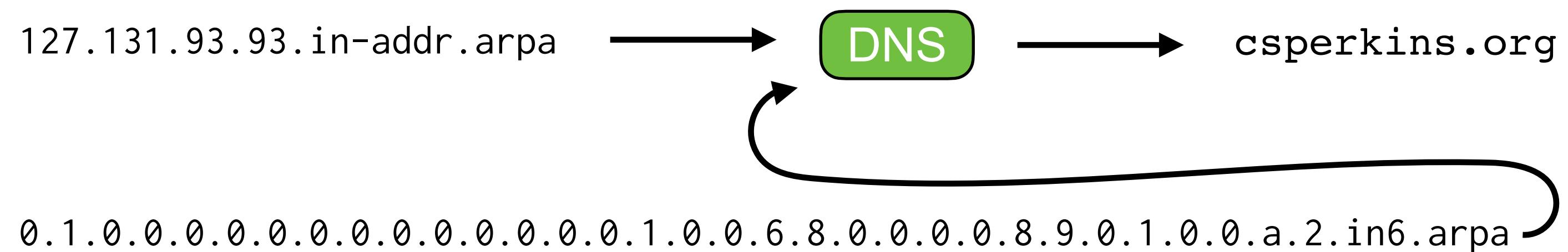
- DNS also includes Generic Top-level Domains (gTLDs)
  - Core set of gTLDs:
    - .com, .org, .net – unrestricted use
    - .edu – higher educational organisations (restricted use; primarily US-based)
    - .mil – US military
    - .gov – US government
    - .int – International Treaty Organisations (United Nations, Interpol, NATO, Red Cross, ...)
  - ICANN has since massively expanded the set of gTLD registrations
    - ~1,500 gTLDs registered
    - e.g., .scot is a gTLD

# Top-Level Domains (4/5)

- The infrastructure top-level domain, **.arpa**, is a historical relic
  - Used in the transition from the ARPANET – the precursor to the Internet
  - It has one current use: reverse DNS
- Forward DNS lookup:



- Reverse DNS lookup:



# Top-Level Domains (5/5)

- Six special-use domains exist:
  - **.example** – examples and documentation (**example.com**, **example.org**, etc., also exist)
  - **.invalid** – guaranteed never to exist
  - **.local** – represents the local network
  - **.localhost** – represents the local machine
  - **.onion** – gateway to Tor hidden services (see <https://datatracker.ietf.org/doc/rfc7686/>)
  - **.test** – for testing

# Internationalised DNS

- DNS names should be available in any language
  - Initial TLDs and sub-domains were in ASCII, but UTF-8 ought to be allowed
  - DNS should, **in principle**, work with UTF-8 name – in practice it doesn't, due to protocol ossification
- Internationalised DNS works around this by translating non-ASCII names into ASCII:
  - Punycode (see <https://datatracker.ietf.org/doc/rfc3492/>)
    - Encodes **any** unicode text as a sequence of ASCII letters, digits, and hyphens
    - München → Mnchen-3ya
    - Bahnhof München-Ost → Bahnhof Mnchen-Ost-u6b
    - Part after final hyphen is a base-36 (a-z0-9) encoded representation of a sequence of Unicode character and the locations where they should be inserted
  - Internationalised DNS names use Punycode, prefixed with `xn--`
    - e.g., `http://Яндекс.рф` translates to `http://xn--70akdum1a.xn--p1ai` (Yandex, a Russian search engine)

# The DNS Root (1/3)

- ICANN decides the set of legal top-level domains – the root servers then advertise the name servers for these domains
- What are the root servers?
  - The set of 13 servers that advertise the name servers for the top level domains
    - a.root-servers.net → m.root-servers.net
    - Also have well-known IPv4 and IPv6 addresses
  - Why 13 servers?
    - Want to be able to ask a DNS resolver to return a list of the root servers
    - DNS over UDP has a size limit on replies → 13 root servers is all that will fit into a single UDP packet

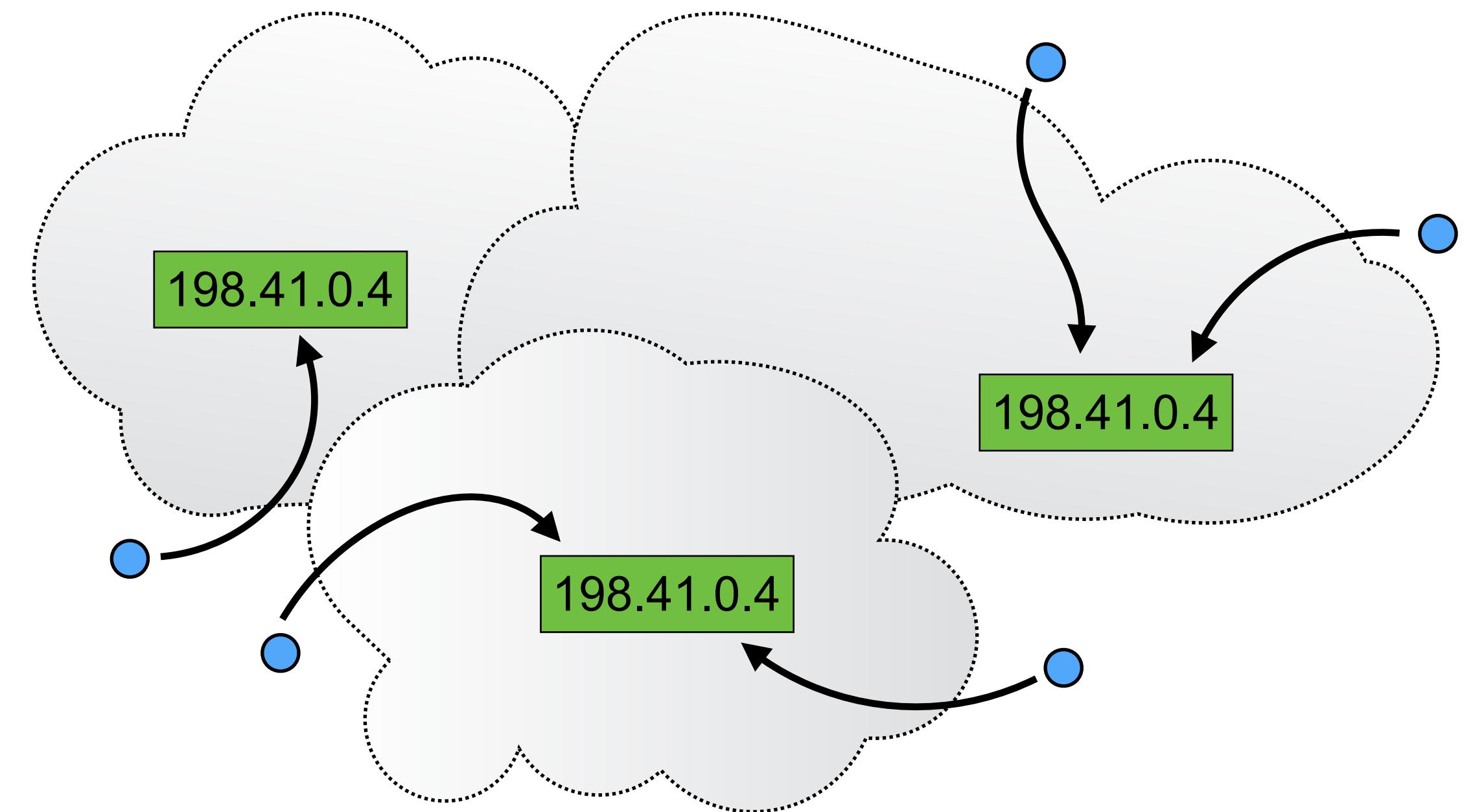
# The DNS Root (2/3)

Server	IPv4 Address	IPv6 Address	Operator
A	198.41.0.4	2001:503:ba3e::2:30	Verisign
B	199.9.14.201	2001:500:200::b	USC-ISI
C	192.33.4.12	2001:500:2::c	Cogent Communications
D	199.7.91.13	2001:500:2d::d	University of Maryland
E	192.203.230.10	2001:500:a8::e	NASA Ames Research Center
F	192.5.5.241	2001:500:2f::f	Internet Systems Consortium
G	192.112.36.4	2001:500:12::d0d	US Defense Information Systems
H	198.97.190.53	2001:500:1::53	US Army Research Lab
I	192.36.148.17	2001:7fe::53	Netnod
J	192.58.128.30	2001:503:c27::2:30	Verisign
K	193.0.14.129	2001:7fd::1	RIPE NCC
L	199.7.83.42	2001:500:9f::42	ICANN
M	202.12.27.33	2001:dc3::35	WIDE Project

- Heavily US-based, for historical reasons – discussed later
- The IP addresses of root servers cannot be changed – they're too widely known – who operates the root servers could change

# The DNS Root (3/3)

- There are not really 13 servers → **anycast routing** (lecture 9)
- Same IP address advertised from multiple places in the network – which replica you reach depends where you're located
- There are 13 IP addresses used by the root servers, but many more physical servers



# DNS Names

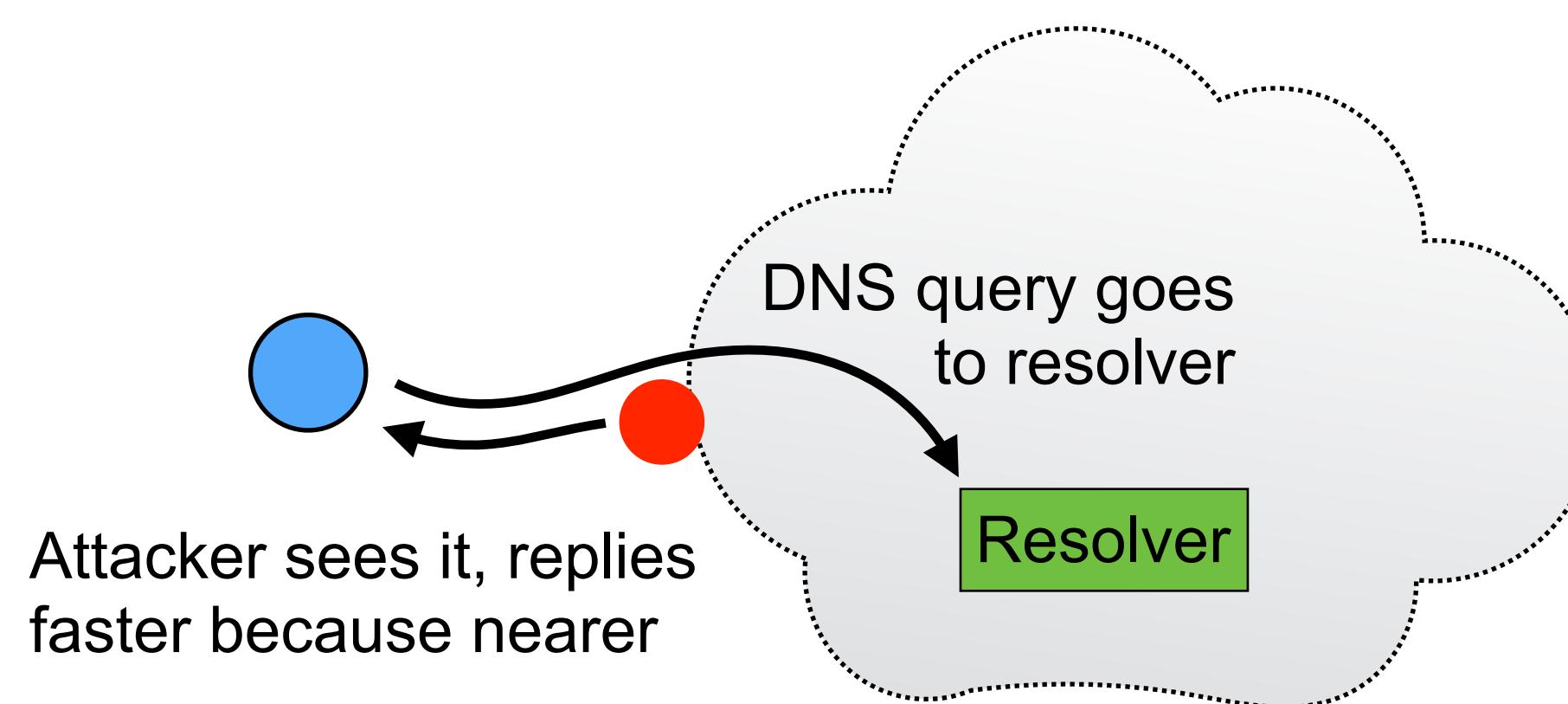
- Who controls the DNS?
- Top-level domains
- Internationalised DNS
- The DNS root

# Methods for DNS Resolution

- DNS security
- DNS resolution over UDP, TLS, HTTPS, and QUIC

# DNS Security (1/3)

- DNS has historically been completely insecure
  - Requests and responses delivered via unencrypted and unauthenticated protocol
  - Responses do not include a digital signature to verify authenticity of data
  - Trivial to eavesdrop on who is looking up what name
  - Trivial for on-path attackers, or malicious resolvers, to forge replies



e.g., possible if victim and attacker are in the same cafe, using insecure Wi-Fi

# DNS Security (2/3)

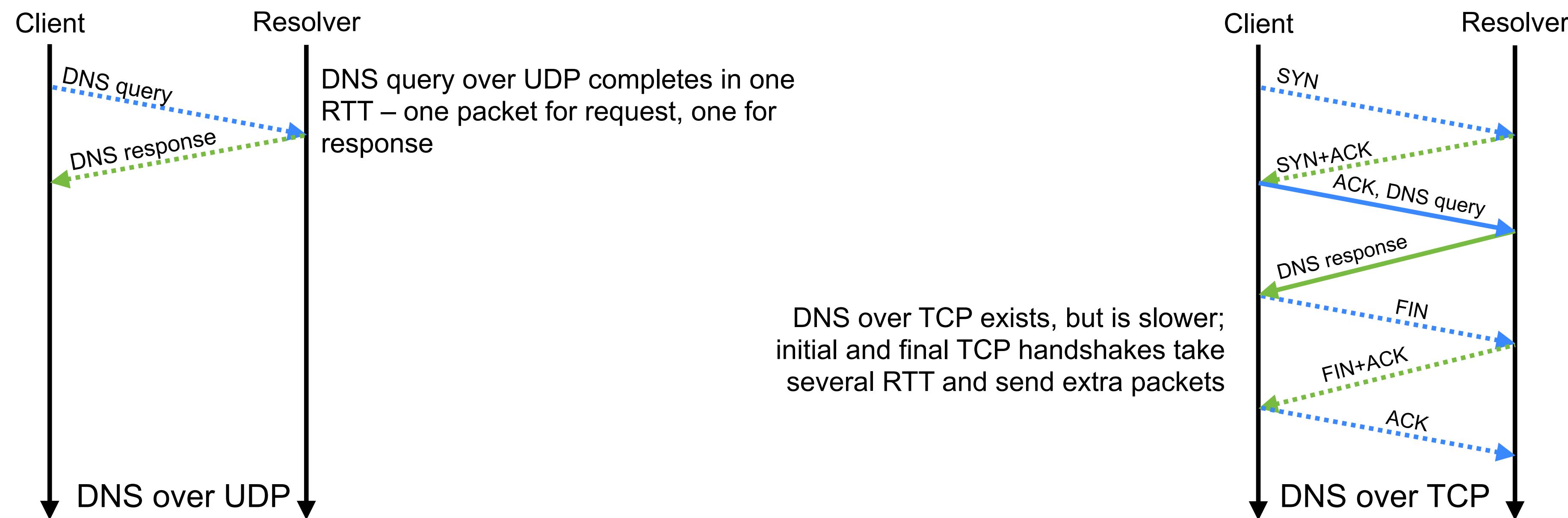
- Two approaches to securing DNS:
  - Transport security
    - Make DNS requests, and receive replies, over TLS (or some other secure channel)
    - Requests and responses are encrypted, so can't be understood or modified by attacker
    - If you trust the resolver, this protects against attack

# DNS Security (3/3)

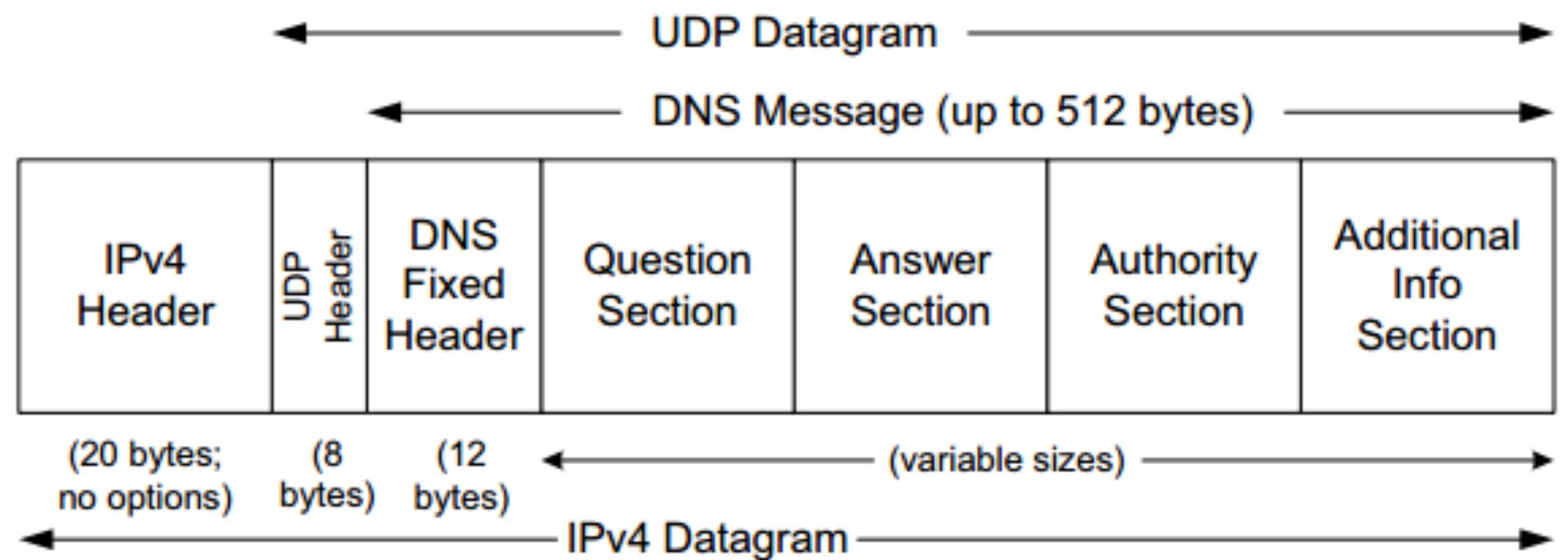
- Two approaches to securing DNS:
  - Transport security
  - Record security – DNSSEC
    - Add a digital signature to DNS responses that client can verify to check the data is valid
      - ICANN signs the root zone
      - Root servers sign information they provide about TLDs
      - TLDs sign information they provide about sub-domains
      - ...
    - Allows a client to verify signatures back to the root, providing a chain of trust to demonstrate ownership of a domain – protects against malicious resolvers
    - Makes extensive use of public key cryptographic techniques – details are complex
    - Implemented, but not widely used
  - **Need both transport and record security for fully secure DNS**

# DNS Over UDP (1/4)

- DNS queries generally made over UDP port 53
  - Requests and responses are generally small enough to fit into a single packet
  - TCP reliability isn't needed – if no answer, retransmit the request
  - Congestion control isn't needed – can't adjust the rate you send a single packet

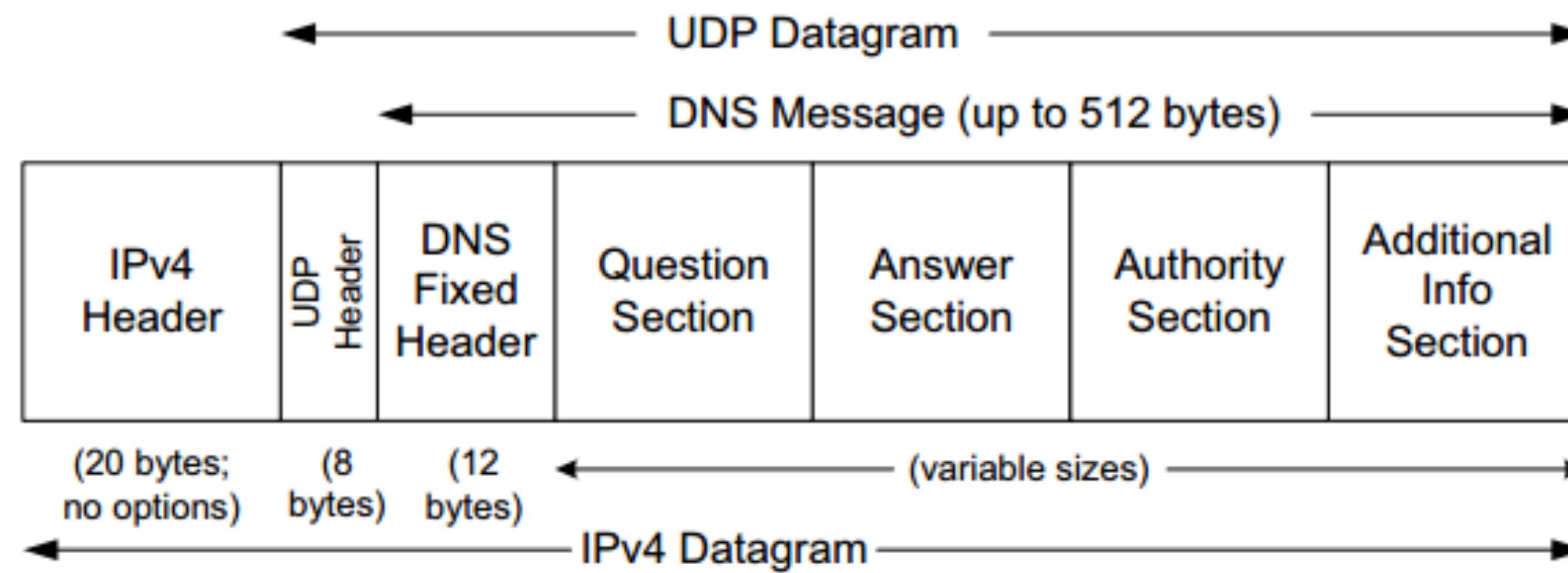


# DNS Over UDP (2/4)



- Question section:
  - List of domain names and requested record type
    - e.g., what is the AAAA record for domain `csperkins.org`
  - Can include more than one question

# DNS Over UDP (3/4)



- Answer, authority, and additional information sections:
  - List of domain names and record type, record data, and time-to-live
  - Answer section answers a question made in a previous request
    - e.g., the AAAA record for domain csperkins.org is 2a00:1098:0:86:1000::10 and it's valid for 1 hour
  - Authority describes where the answer came from

# DNS Over UDP (4/4)

```
[stlinux02] > dig csperkins.org

; <>> DiG 9.11.4-P2-RedHat-9.11.4-9.P2.el7 <>> csperkins.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51409
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;csperkins.org.      IN A

;; ANSWER SECTION:
csperkins.org.    2681  IN A   93.93.131.127

;; AUTHORITY SECTION:
csperkins.org.    78278  IN NS ns2.mythic-beasts.com.
csperkins.org.    78278  IN NS ns1.mythic-beasts.com.

;; ADDITIONAL SECTION:
ns1.mythic-beasts.com. 70870  IN   A   45.33.127.156
ns2.mythic-beasts.com. 157301  IN   A   93.93.128.67
ns1.mythic-beasts.com. 70870  IN   AAAA 2600:3c00:e000:19::1
ns2.mythic-beasts.com. 157301  IN   AAAA 2a00:1098:0:80:1000::10

;; Query time: 0 msec
;; SERVER: 130.209.244.1#53(130.209.244.1)
;; WHEN: Wed Mar  4 18:26:53 GMT 2020
;; MSG SIZE  rcvd: 199
```

The `dig` tool on Linux or macOS performs DNS queries

# DNS Over TLS (DoT)

- DNS over UDP is insecure
  - The packets are not encrypted or authenticated
  - Devices on the path between client and resolver can see DNS queries and responses – and can forge responses
- DNS over TLS solves this problem
  - DNS client opens a TCP connection to the resolver (port 853)
  - DNS client and resolver negotiate a TLS 1.3 session on the TCP connection
  - DNS client sends query, and receives response, over the TLS connection
    - DNS over TLS messages are formatted exactly the same as DNS over UDP, and contain exactly the same information – only difference is that they're sent over TLS not UDP
    - Slower and higher overhead than DNS over UDP – due to need to negotiate TCP and TLS –but more secure

# DNS over HTTPS (DoH)

- DoH allows a client to send queries to a resolver using HTTPS
  - Can use with either GET or POST methods in HTTPS

```
GET /dns-query?dns=AAABAAABAAAAAAA3d3dw1eGftcGx1A2NvbQAAAQAB HTTP/1.1
```

```
Accept: application/dns-message
```

Base-64 encoded version of the data that  
would be sent in a DNS-over-UDP request

Uses /dns-query as the URL path

```
POST /dns-query HTTP/1.1
```

```
Accept: application/dns-message
```

```
Content-type = application/dns-message
```

```
Content-length = 33
```

```
<33 bytes of UDP query, exactly as if sent in a UDP packet>
```

- HTTP response has Content-Type: application/dns-message and contains the exact same data that would be sent in a UDP-based DNS response

# DNS over QUIC (DoQ)

- Work in progress to define DNS over QUIC:
  - <https://datatracker.ietf.org/doc/draft-huitema-quic-dnsquic/>
  - Same principle as DNS over TLS:
    - Client opens a QUIC connection to the resolver
      - Negotiates TLS security as part of the connection setup
    - Sends the request and receives the response over that connection
      - Requests and response contain exactly the same data as DNS over UDP – just sent via QUIC

# Methods for DNS Resolution

- Increasingly many ways of making DNS queries:
  - DNS over UDP
  - DNS over TLS
  - DNS over HTTPS
  - DNS over QUIC
- The contents of the query and the response are **identical** in all cases
  - They change how the query is delivered to the resolver and how the response is returned, but not the contents of the messages
  - They change the security guarantees provided
  - They potentially gives clients more flexibility to query different resolvers

# Methods for DNS Resolution

- DNS security
- DNS resolution over UDP, TLS, HTTPS, and QUIC

# The Politics of Names

- Choice of DNS resolver
- Intellectual property and the DNS
- What domains should exist?
- Who controls the DNS root?

# Implications of Choice of DNS Resolver (1/4)

- **How is the DNS resolver chosen?**
- When connecting to network, hosts use DHCP (dynamic host configuration protocol) to discover network settings and configuration
  - DHCP tells the host what DNS resolver to use for the network
  - If a host has multiple network interfaces, it may use a different DNS resolver for each
    - The `getaddrinfo()` call takes a `hints` parameter, that can include the local IP address
    - e.g., consider a device connected to 4G cellular network and a private company Ethernet – the Ethernet might make available names of internal services that aren't accessible to the public
- Possible to configure the DNS resolver manually
  - e.g., to talk to Google public DNS resolver (IPv4 address 8.8.8.8)

# Implications of Choice of DNS Resolver (2/4)

- DNS resolution has typically been a system-wide service
  - Operating system implements a DNS resolution service; all DNS queries use that service
  - A consistent mapping of names to addresses
- **DoH is changing this**
  - JavaScript web applications can now easily perform DNS queries via any HTTP website
  - Each application may get different answers for the same query, depending on the server; it's no longer easily possible to enforce policy via the DNS

# Implications of Choice of DNS Resolver (3/4)

- Giving applications ability to securely access arbitrary DNS servers allows them to avoid local observation and/or filtering of DNS traffic
- Is this flexibility for each application to perform DNS queries differently a concern?

## No

Applications should have the ability to use a secure DNS server they trust to avoid phishing attacks, malware, monitoring, etc.

Why should network operators be able to see DNS queries and modify responses? This is a privacy and security risk

## Yes

Network operators filter DNS responses to block access to malicious sites and prevent malware spreading – allowing applications to bypass this is a security risk

Network operators filter DNS responses to enforce legal or societal constraints – e.g., Internet Watch Foundation DNS block list to stop UK-based access to sites hosting child sexual abuse material

# Implications of Choice of DNS Resolver (4/4)

- **Can a network restrict the choice of DNS resolver?**
- Firewalls can block access to DNS-over-UDP and DNS-over-TLS resolvers
  - Block access to UDP port 53
  - Block access to TCP port 853
  - For all destination IP addresses except those of allowed DNS resolvers
- Difficult to block DNS-over-HTTPS
  - Network cannot distinguish DNS-over-HTTPS from any other traffic over HTTPS
  - May be able to tell from the destination IP address
    - e.g., Google use IPv4 address 8.8.8.8 for public DoH services, but not other traffic
  - But if a web server handles a mix of web and DNS traffic over HTTPS, cannot block one without blocking the other
- Many ISPs and governments concerned that DoH prevents use of DNS as a control point

# Intellectual Property and the DNS

- Intellectual property is managed on a national basis
  - e.g., a company might own a trademark in the UK while a different company owns the same trademark in the Republic of Ireland
  - Which of those companies owns *trademark.ie* and which owns *trademark.co.uk* is a straightforward legal question
  - Which of those companies owns *trademark.com* is likely harder to decide
    - Especially since .com is operated by a US-based organisation, and a third company might own the trademark in the US
- A ccTLD clearly operates under legal regime of a particular country
- Use of a gTLD might lead to legal complications

# What Domains Should Exist?

- **Should a particular gTLD be allowed to exist?**
- For example, should .xxx exist to host “adult” content?
  - If so, who gets to decide what content should (must?) sit within that gTLD?
  - Different countries have very different norms and standards in this area
  - Who controls what TLDs ICANN permits? Who should control it?

# What Domains Should Exist?

- Should a particular subdomain be allowed to exist?
- Significant differences around freedom of speech and permissible topics in different parts of the world
- A ccTLD can enforce local conventions and rules
- What rules apply to a gTLD?
  - If a particular country/group finds a site objectionable, should it be taken down?
  - If country X decides particular content is illegal, but it is legal in country Y, should a gTLD operated out of country Y but accessible in country X permit such content?
  - e.g., Holocaust denial is illegal in Germany but not in the US – should .com, operating from the US, permit sites hosting such content?

# Who Controls the Root Servers?

- DNS root servers are mostly controlled by US-based organisations
- Is this a risk for other countries?
- Should the root servers be controlled by a broader mix of countries?
  - If so, who gets to decide – ICANN? The UN?
  - Is there a benefit in controlling a DNS root server?
  - Is there a benefit in controlling a gTLD server?  
Who gets to host **.com**, for example?

Server	IPv4 Address	IPv6 Address	Operator
A	198.41.0.4	2001:503:ba3e::2:30	Verisign
B	199.9.14.201	2001:500:200::b	USC-ISI
C	192.33.4.12	2001:500:2::c	Cogent Communications
D	199.7.91.13	2001:500:2d::d	University of Maryland
E	192.203.230.10	2001:500:a8::e	NASA Ames Research Center
F	192.5.5.241	2001:500:2f::f	Internet Systems Consortium
G	192.112.36.4	2001:500:12::d0d	US Defense Information Systems Agency
H	198.97.190.53	2001:500:1::53	US Army Research Lab
I	192.36.148.17	2001:7fe::53	Netnod
J	192.58.128.30	2001:503:c27::2:30	Verisign
K	193.0.14.129	2001:7fd::1	RIPE NCC
L	199.7.83.42	2001:500:9f::42	ICANN
M	202.12.27.33	2001:dc3::35	WIDE Project

# Should There Be a Single DNS Root?

- Should all TLDs be accessible from everywhere?
  - Should there be a single global DNS?
  - Should the same name always resolve to the same site?
    - With global content distribution networks, how can you tell?
  - Should different countries be allowed to filter DNS?
    - If so, how should such restrictions be implemented?
    - It is difficult to distinguish modifications to DNS responses made to conform to government-mandated filtering requirements from those made by malware, phishing attacks, etc. – is this a feature or a bug?

# Naming and the Tussle for Control

- What is the DNS?
- How are DNS queries made?
- Who controls the names?



University  
ofGlasgow

# Networks and Internet Routing

Networked Systems (H)  
Lecture 9



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Lecture Outline

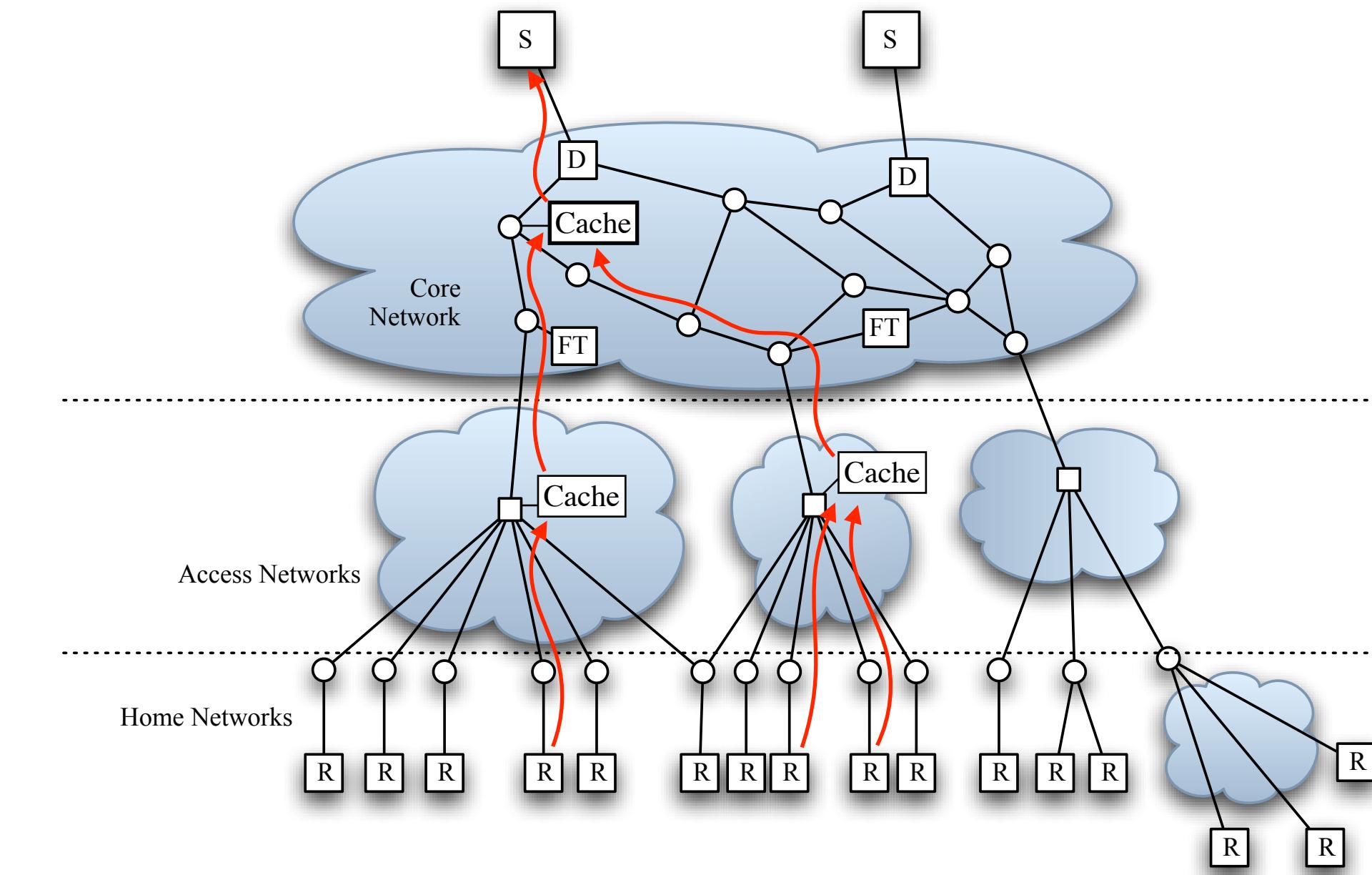
- Networks and Internet Routing:
  - Content distribution networks
  - Inter-domain routing
  - Routing security
  - Intra-domain routing
- Wrap-up

# Content Distribution Networks (CDNs)

- Load balancing and latency reduction
- Implementation using DNS
- Implementation using anycast routing

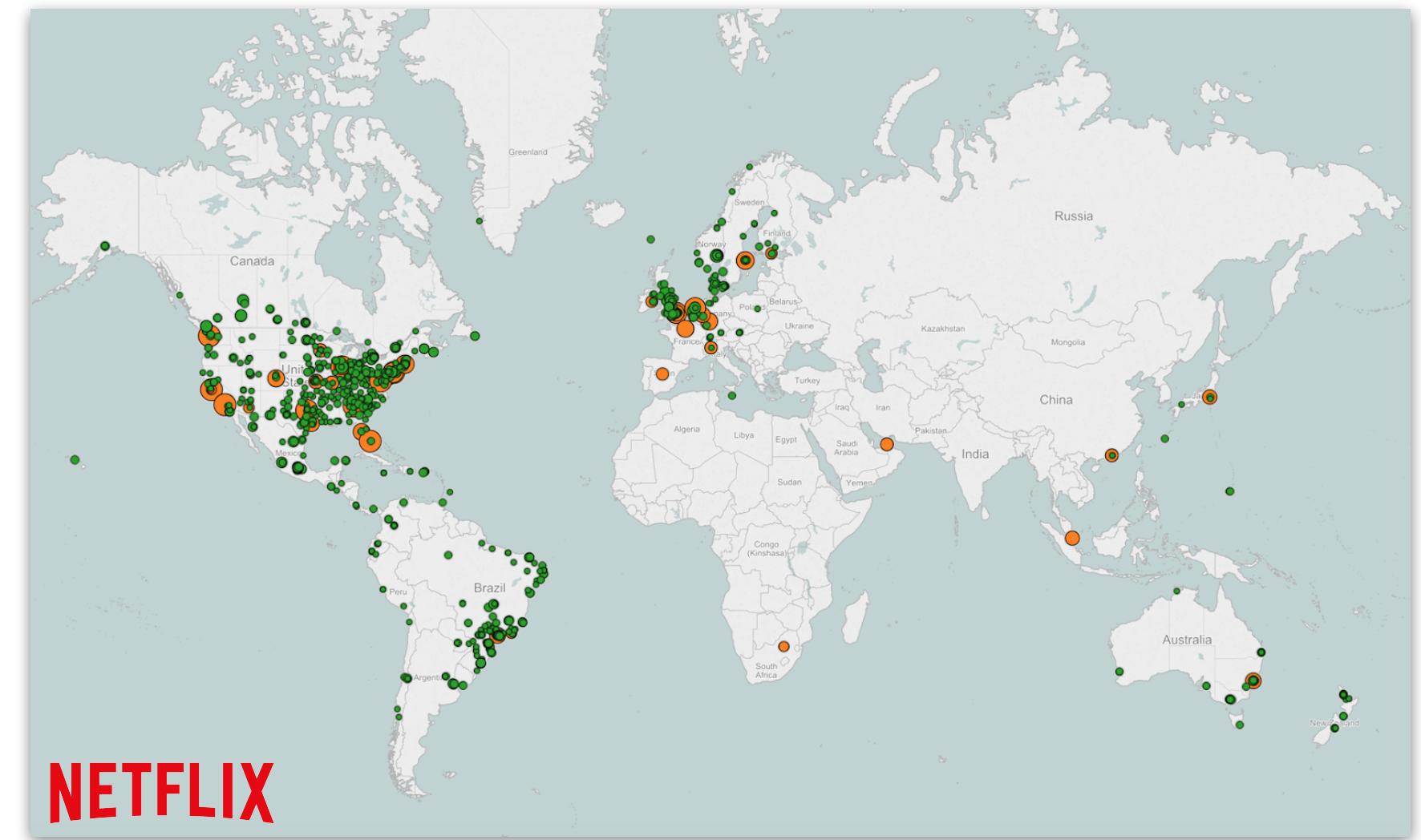
# Content Distribution Networks (CDNs)

- **Content distribution networks (CDNs)** provide scalability, load balancing, and low latency
  - Host content for their customers in web caches spread around the world in edge networks and data centres
  - Reduces load on the main servers – rather than keep a local copy of the files, they link to the CDN-hosted copy
  - Reduces latency to respond to requests
  - Reduces chances of successful denial-of-service attack
- Many commercial CDNs available
- Many large organisations run their own
  - **Hypergiants** such as Google, Facebook, Netflix, ...



# Using CDNs to Distribute Load

- CDNs distribute load by locating web caches around the world and answering most requests from a local cache
- Needs extensive investment, large-scale cooperation with ISPs, Internet Exchange Points, etc.
- Mutual benefit for an ISP to host the servers for a CDN
  - Reduces the load on the ISP's network
    - One copy of the content sent to cache; the cache then distributes many times
    - Avoids overload of link from ISP to outside world
    - e.g., Netflix distribute “tens of terabits per second” of video – not possible from a single data centre; needs local fanout
  - Increases the reach and robustness of the CDN



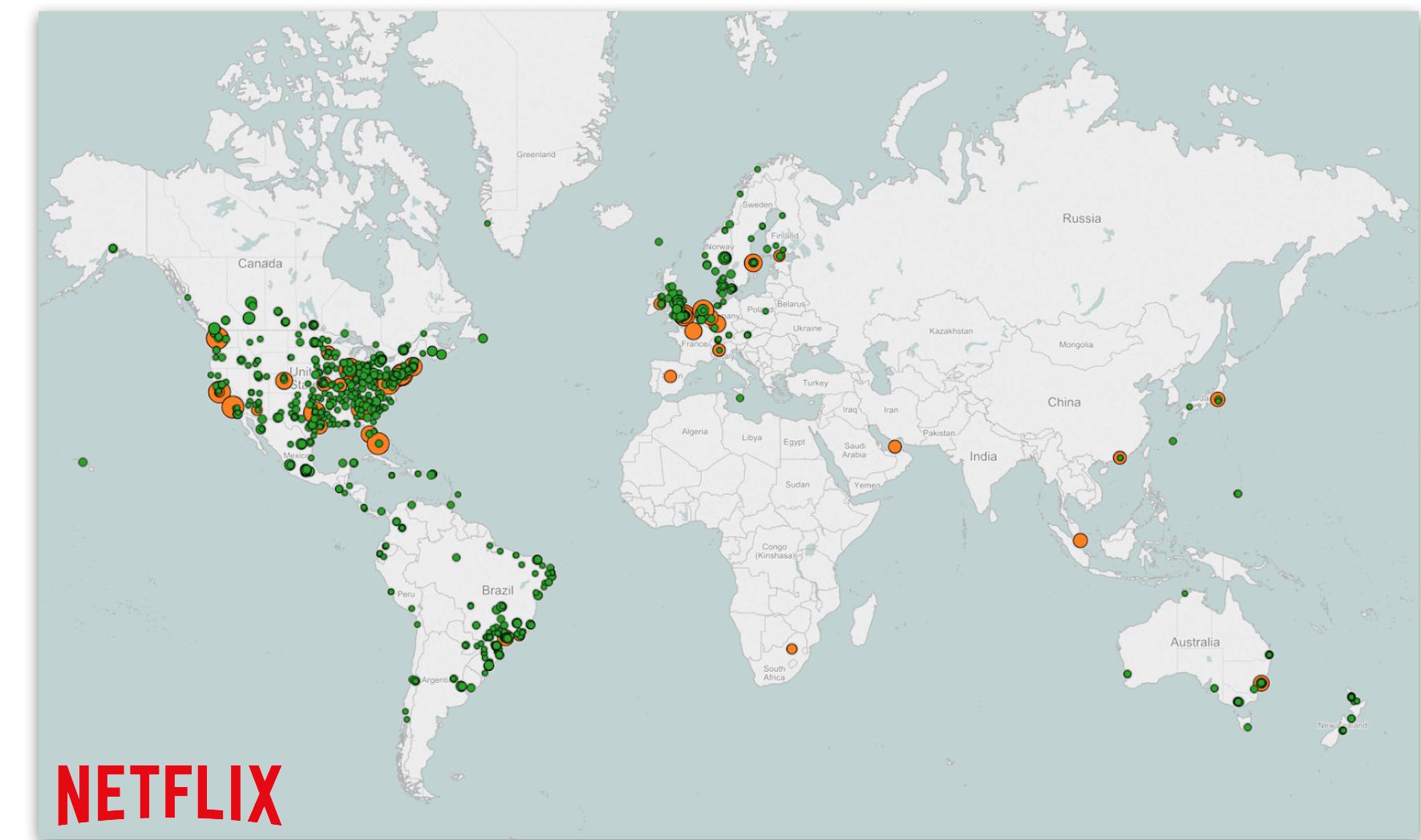
Netflix Open Connect CDN locations (source: <https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience>)



“more than 240,000 servers  
in over 130 countries”

# Using CDNs to reduce latency

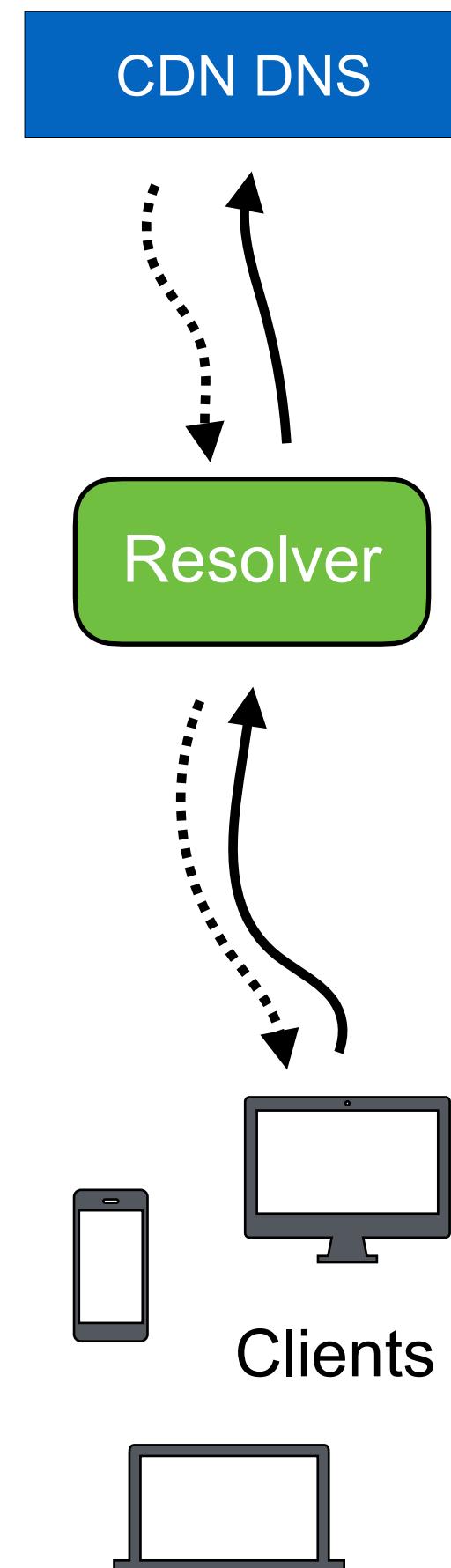
- Key CDN benefit is reducing latency → content is delivered from nearby caches
  - e.g., requests to a US-based service are answered from a CDN cache in Europe, rather than having to cross long-distance path to the US
- Requires global distribution of CDN proxy caches
  - Are CDNs effectively serving the entire world?
  - Providing effective Internet access to developing regions requires more than just connectivity; also needs data centres and infrastructure to host the CDN nodes
- Effective for cacheable static content – video, software updates, images – may need edge compute infrastructure to support other applications



Netflix Open Connect CDN locations (source: <https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience>)

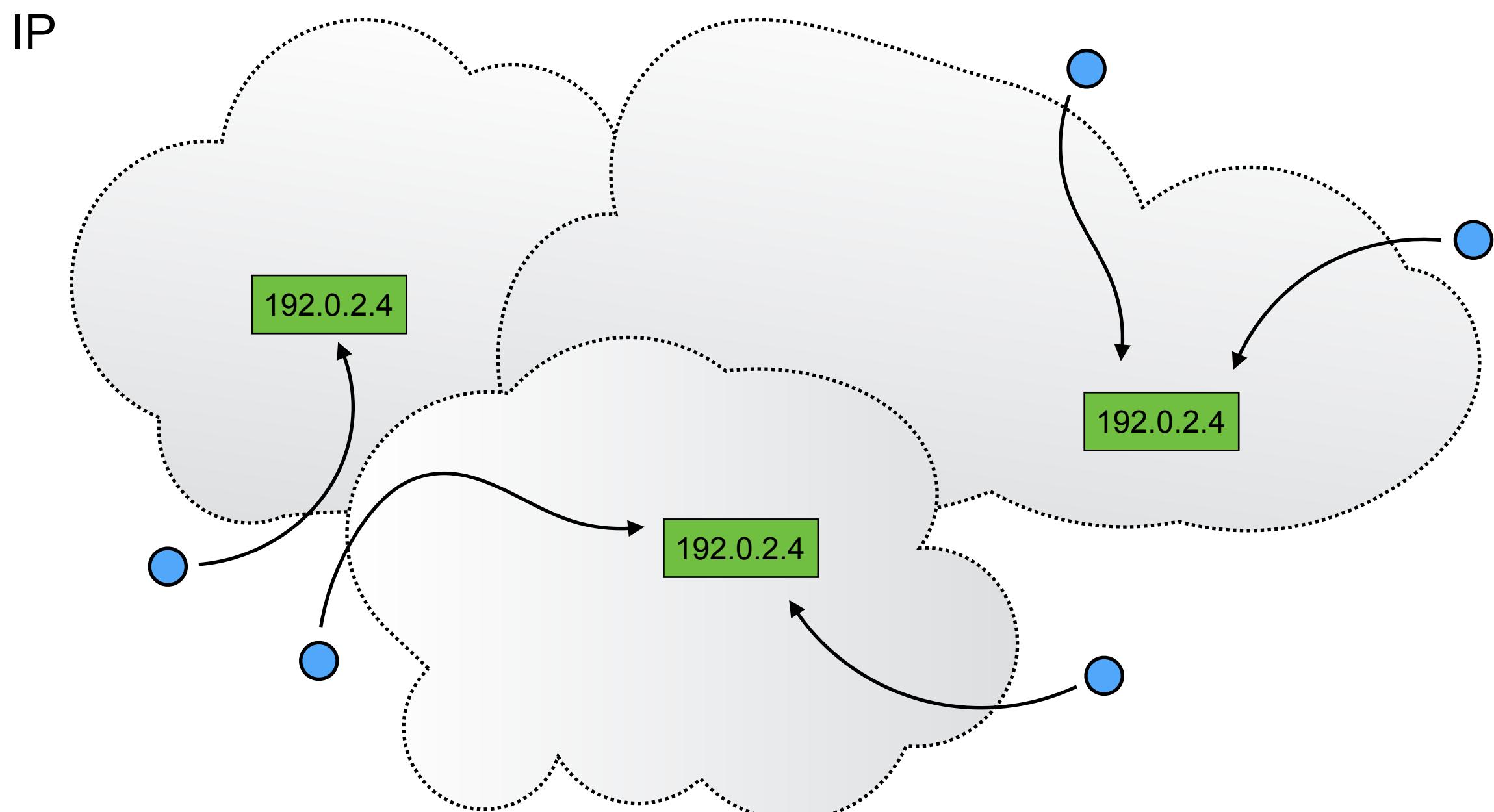
# Locating the Nearest CDN Node using DNS

- Locate nearest CDN node using the DNS
- Each resource hosted by the CDN has a unique DNS name – each resource is given a different domain name
  - e.g., the CDN hosts `https://example.com/images/kitten.jpg` as `https://9BC1C10B7947A890B9.cdn-example.com/kitten.jpg`
- DNS server for the CDN returns different A or AAAA records for a name, depending on where it's requested from, what CDN caches have the data
  - Directs to local cache based on IP address of resolver making the query
    - DNS client subnet extension [RFC7871] for remote resolvers
    - Doesn't need to be particularly accurate – trying to figure out if you're in UK rather than US, to direct you to data centre in London rather than New York
    - Allows very fine-grained control, but puts high load on DNS



# Locating the Nearest CDN Node using Anycast Routing

- Locate nearest CDN node using **anycast routing**
- Each resource hosted by the CDN has a unique filename
  - e.g., CDN hosts `https://example.com/images/kitten.jpg` as `https://cache.cdn-example.com/9BC1C10B7947A890B9.jpg`
  - The DNS name always maps to the same IP address – that IP address is a load balancer at the entrance to a data centre
- **The CDN uses multiple data centres – all using the same IP addresses**
  - Each data centre advertises its addresses via BGP
  - Inter-domain Internet routing will ensure traffic goes to the closest data centre to source



# Content Distribution Networks (CDNs)

- Load balancing and latency reduction
- Implementation using DNS
- Implementation using anycast routing

# Inter-domain Routing

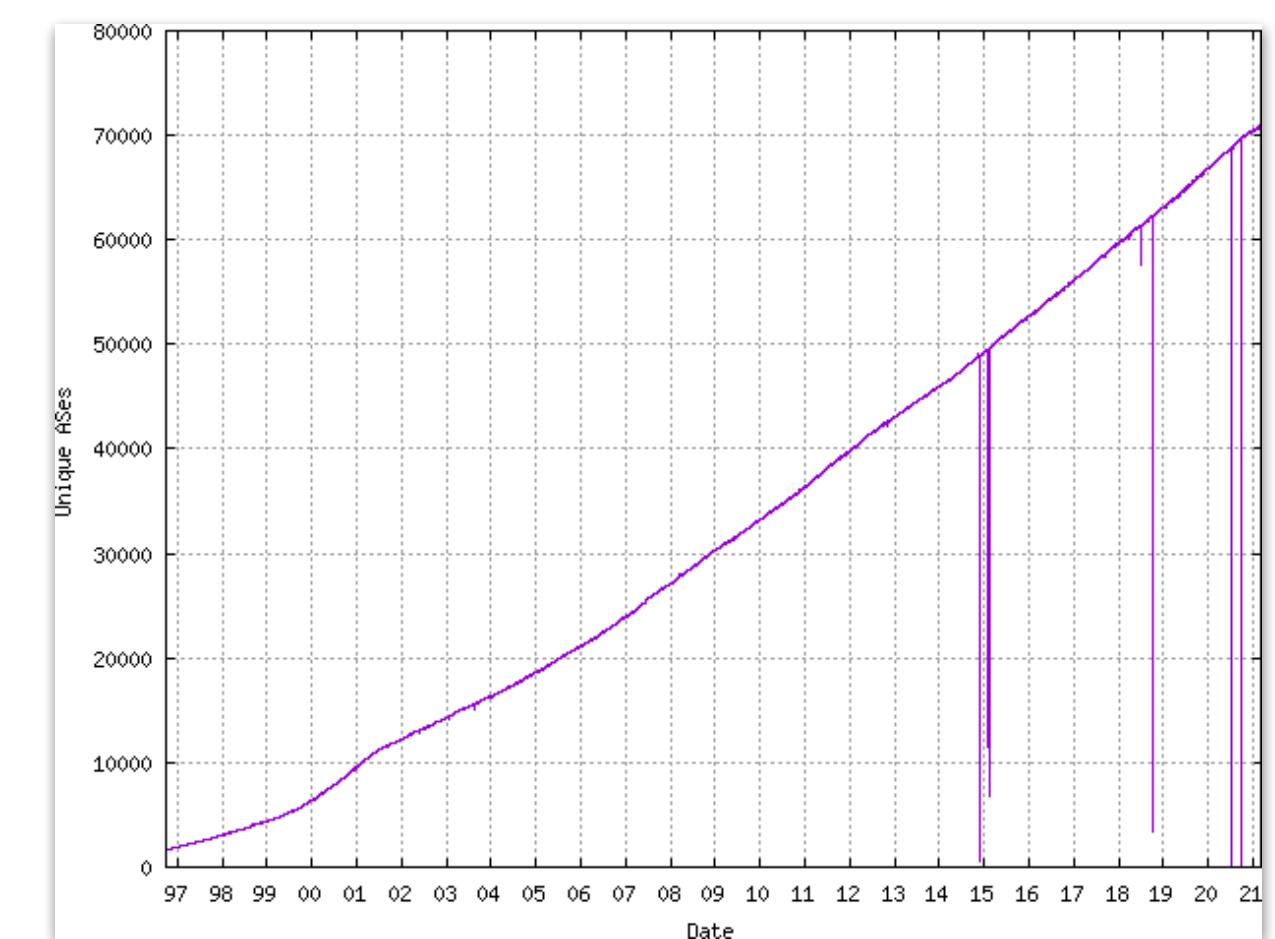
- Autonomous Systems and the AS graph
- Routing at the edge
- Routing in the core
- BGP

# Inter-domain Routing

- The Internet is a network of networks
  - Each network is an **Autonomous System (AS)**
  - Each network is a separate routing domain
- **Inter-domain routing** is the problem of finding the best path from the source network to the destination network
  - Treat each network as a node on the routing graph (the “AS topology graph”)
  - Treat connections between networks as edges in the graph
  - What is the best set of networks to choose to get from source to destination

# Autonomous Systems

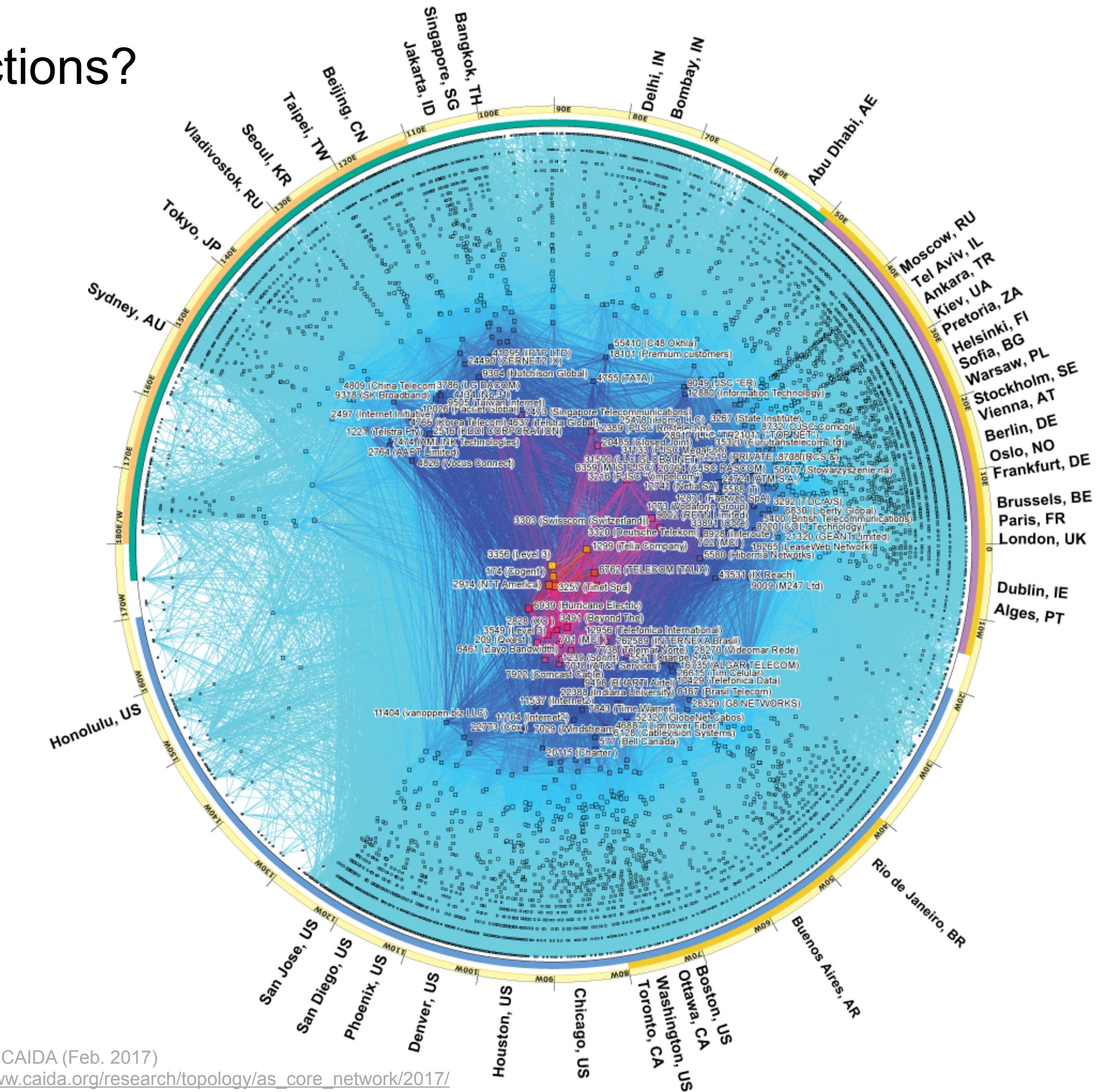
- An **Autonomous System (AS)** is an independently operated network
  - An Internet service provider, or other organisation, that operates a network and wants to participate in interdomain routing
  - Some organisations operate more than one AS
    - For ease of administration
    - Due to company mergers
    - ...
  - Each AS is identified by a unique number, allocated by the RIR
    - 114685 AS numbers allocated as of 6 March 2021; of these 70,926 are advertised in BGP
    - <http://www.potaroo.net/tools/asn32/> – current state of allocation of AS numbers
    - <http://bgp.potaroo.net/cidr/autnums.html> – complete list AS numbers and names, with links to details
    - <http://www.cidr-report.org/as2.0/> – classless interdomain routing (CIDR) report



Source: <http://www.cidr-report.org/as2.0/>

# AS-level Internet Topology Graph (IPv4)

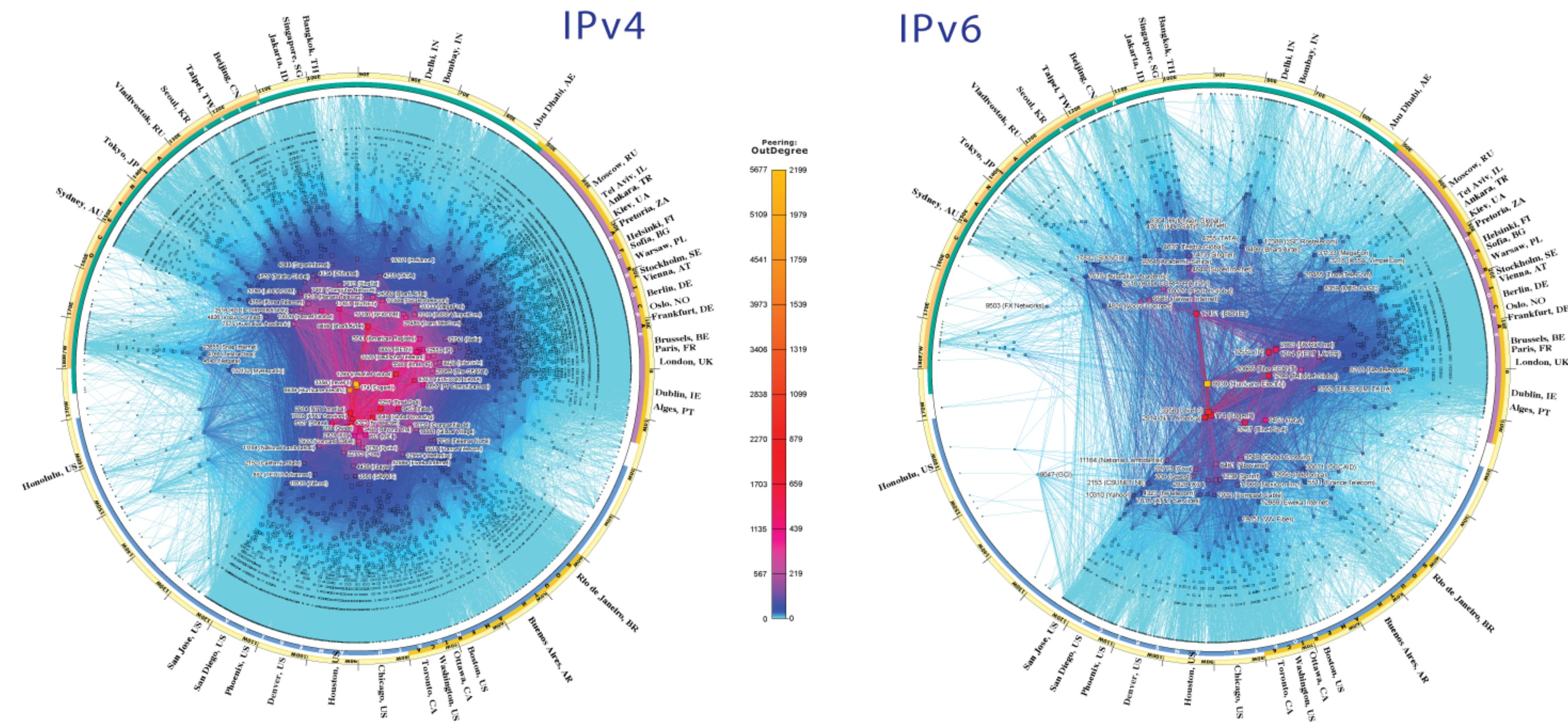
- AS-level topology – what are the inter-AS connections?
  - Each node is an autonomous system
    - Position around circle based on geographic location
    - Distance from centre based on number of links to other networks → more links, closer to the centre
  - Edges show links between autonomous systems
    - A **potential route** traffic can flow
    - Not all possible routes are in use



# AS-level Internet Topology Graph (IPv4 and IPv6)

## CAIDA's IPv4 & IPv6 AS Core AS-level INTERNET GRAPH

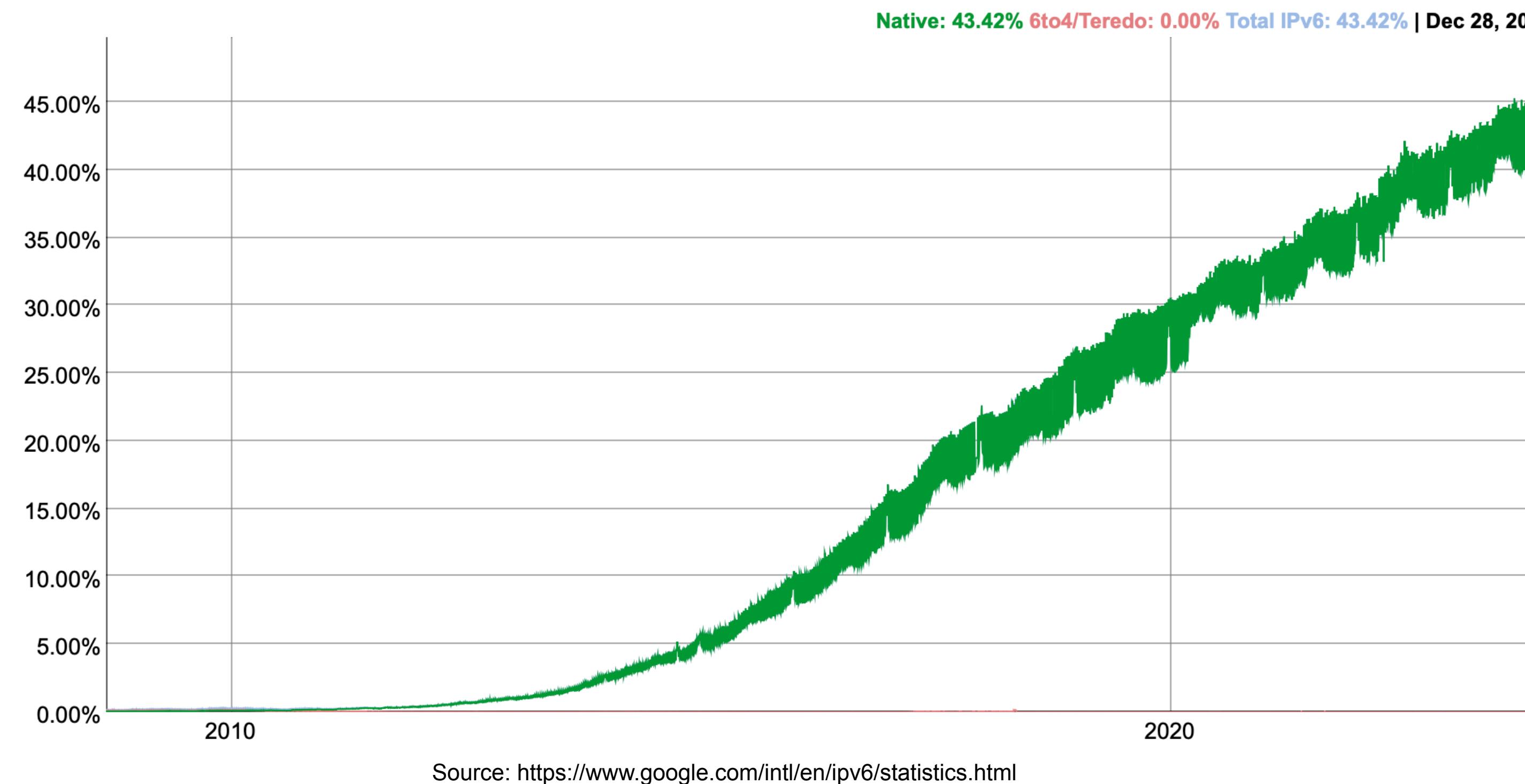
Archipelago January 2015



Copyright © 2015 UC Regents. All rights reserved.

Source: [http://www.caida.org/research/topology/as\\_core\\_network/2015/](http://www.caida.org/research/topology/as_core_network/2015/)

# Differences Between IPv4 & IPv6 AS-level Topologies

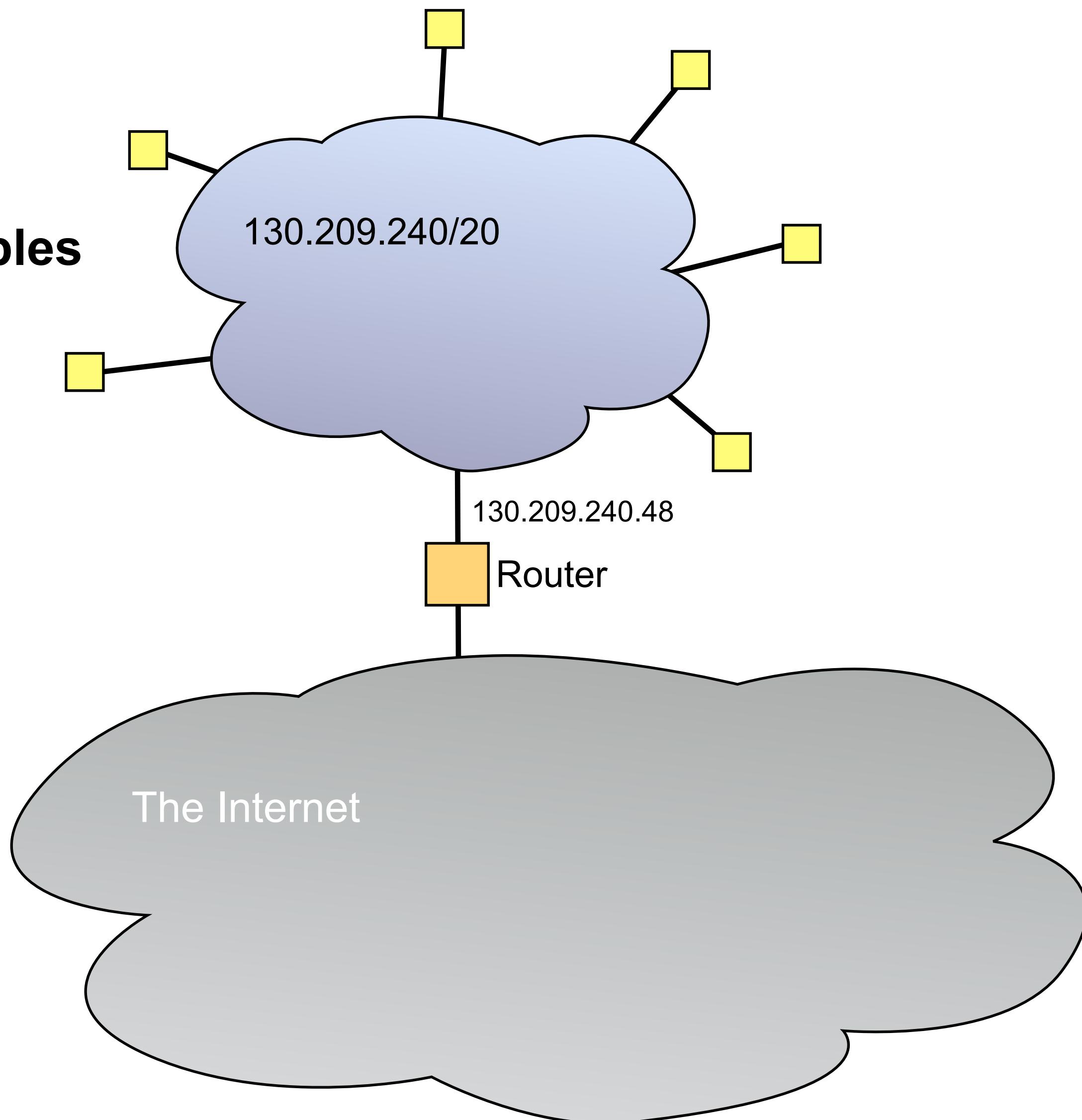


- IPv6 AS-level topology graph is sparse compared to IPv4, but there is significant IPv6 usage → IPv4 has 30 years head start on deployment; the topology can be expected to be more densely interconnected

# Routing at the Edge

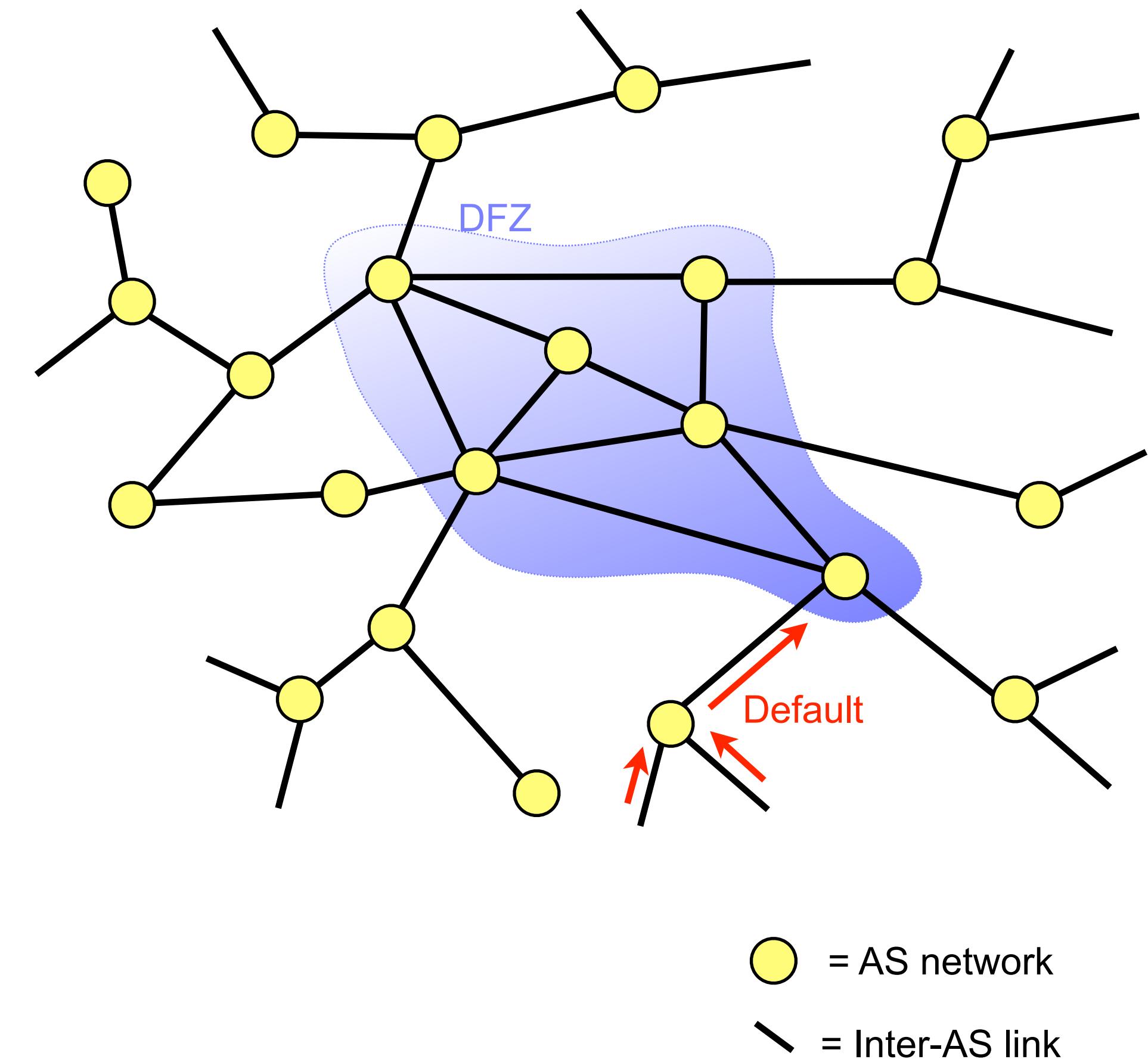
- Devices at the edge tend to have simple **routing tables**
  - The devices on the local network
  - A **default route** for everything else
  - e.g., routing table for hosts in Glasgow SoCS:

Network:	Netmask:	Gateway:
130.209.240.0	255.255.240.0	eth0
default	0.0.0.0	130.209.240.48

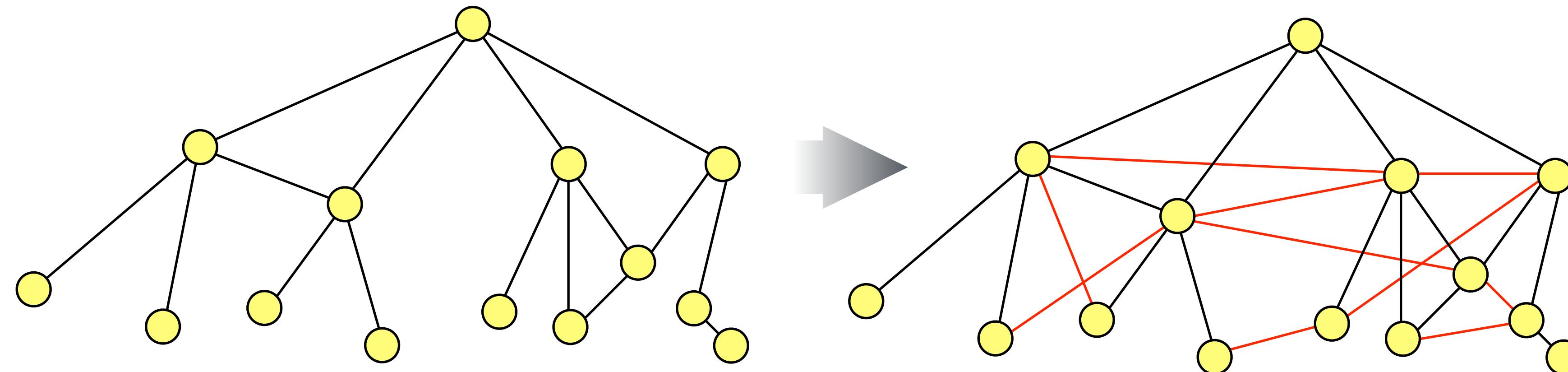


# Routing in the Core of the Internet (1/2)

- Networks near the edge can often use a **default route** for much of their traffic
- Core networks need full routing table
  - The **default free zone** (DFZ)
  - With enough interconnections, they must store routing information about the broad network topology
  - Many possible paths – need information to choose the most appropriate



# Routing in the Core of the Internet (2/2)



- AS-level topology flattening → increasingly rich interconnections
- **Internet Exchange Points (IXPs)** are becoming commonplace
  - Example: London Internet Exchange
  - >850 different networks interconnected
  - >4.3 Tbps of traffic



# Routing Policy

- Interdomain routing is between competitors
  - Each AS is independently operated and may compete for customers
  - Business, political, and economic relationships influence routing
- Routing **must** consider policy
  - Policy restrictions on who can determine the network topology
  - Policy restrictions on which route traffic should follow between a particular source and destination
  - Policy restrictions might prioritise non-shortest path routes
    - Do you prioritise cost, bandwidth, or latency when choosing a route?
    - Traffic between certain networks may be prohibiting from passing through other networks
    - Traffic between certain regions should avoid other regions

# Border Gateway Protocol (BGP)

- Internet routing uses the **Border Gateway Protocol (BGP)**
  - External BGP (**eBGP**) used to exchange routing information between ASes
    - An eBGP session is used by an AS to exchange information about the AS-level topology with a neighbouring AS
    - Runs over a TCP connection between two routers, one in each AS
    - Exchanges knowledge of the AS-level topology of the network, filtered according to policy
    - Used to compute appropriate **interdomain routes** to each destination AS
  - Internal BGP (**iBGP**) propagates this information to routers within an AS
    - An **intra-domain routing protocol** determines routes within the AS
    - iBGP distributes information to internal routers on how to reach external destinations

# Routing Information Exchanged in eBGP

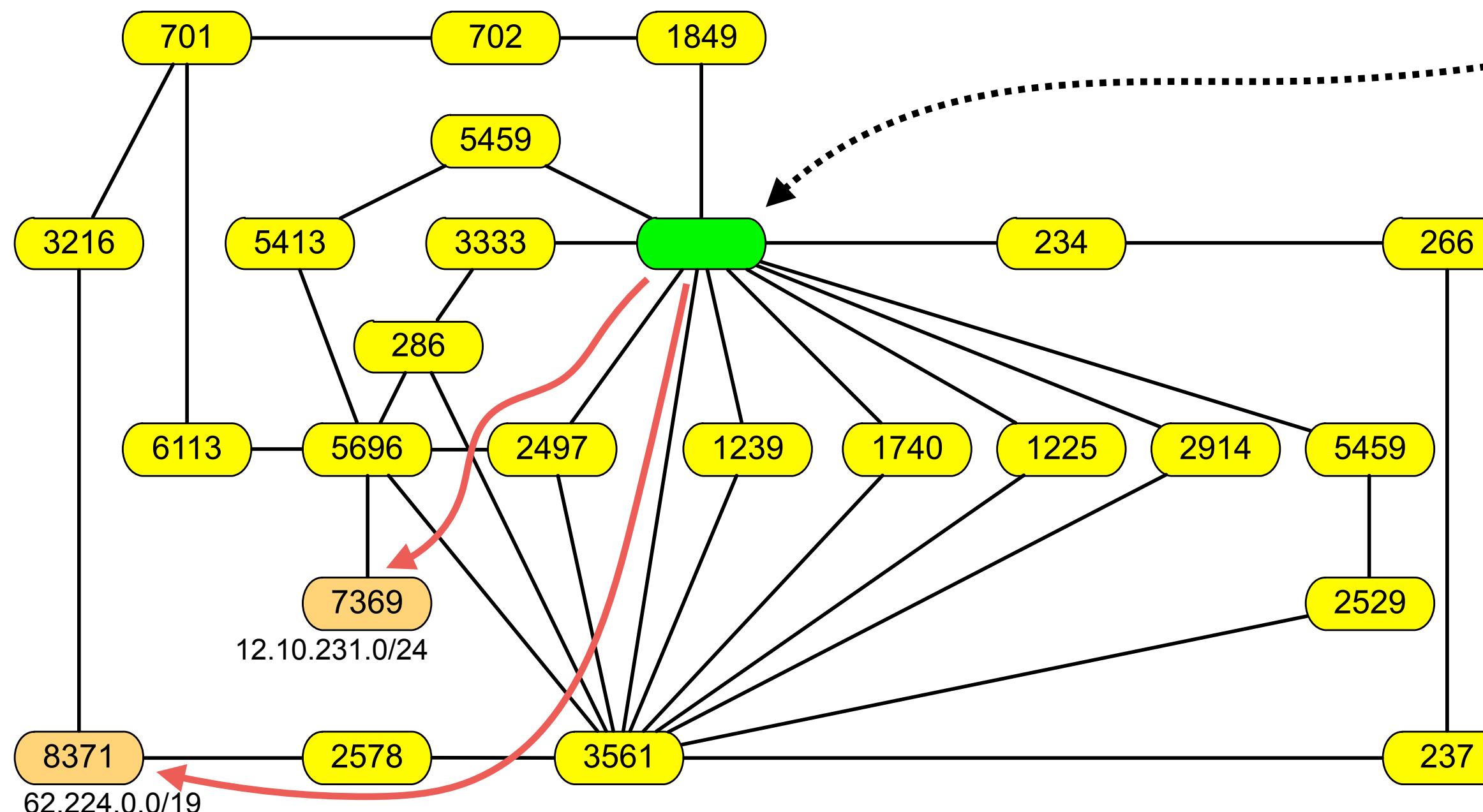
- eBGP routers advertise lists of IP address ranges (“prefixes”) and their associated AS-level paths
- Combined to form a routing table

Prefix	Next Hop	AS Path
...		
* 12.10.231.0/24	194.68.130.254	5459 5413 5696 7369 i
*	158.43.133.48	1849 702 701 6113 5696 7369 i
*	193.0.0.242	3333 286 5696 7369 i
*	204.212.44.128	234 266 237 3561 5696 7369 i
*>	202.232.1.8	2497 5696 7369 i
*	204.70.4.89	3561 5696 7369 i
*	131.103.20.49	1225 3561 5696 7369 i
* 62.224.0.0/19	134.24.127.3	1740 3561 2578 8371 i
*	194.68.130.254	5459 2529 3561 2578 8371 i
*	158.43.133.48	1849 702 701 3216 3216 3216 8371 8371 i
*	193.0.0.242	3333 286 3561 2578 8371 i
*	144.228.240.93	1239 3561 2578 8371 i
*	204.212.44.128	234 266 237 3561 2578 8371 i
*	202.232.1.8	2497 3561 2578 8371 i
*	205.238.48.3	2914 3561 2578 8371 i
*>	204.70.4.89	3561 2578 8371 i
*	131.103.20.49	1225 3561 2578 8371 i
...		

- Hosts with IP addresses in the range 12.10.231.0 - 12.10.231.255 are in AS 7369.
- That AS is best reached via AS 2497, then AS 5696.
- Packets destined for those addresses should be sent to address 202.232.1.8 next, from where they will be forwarded.

# AS Topology Graph

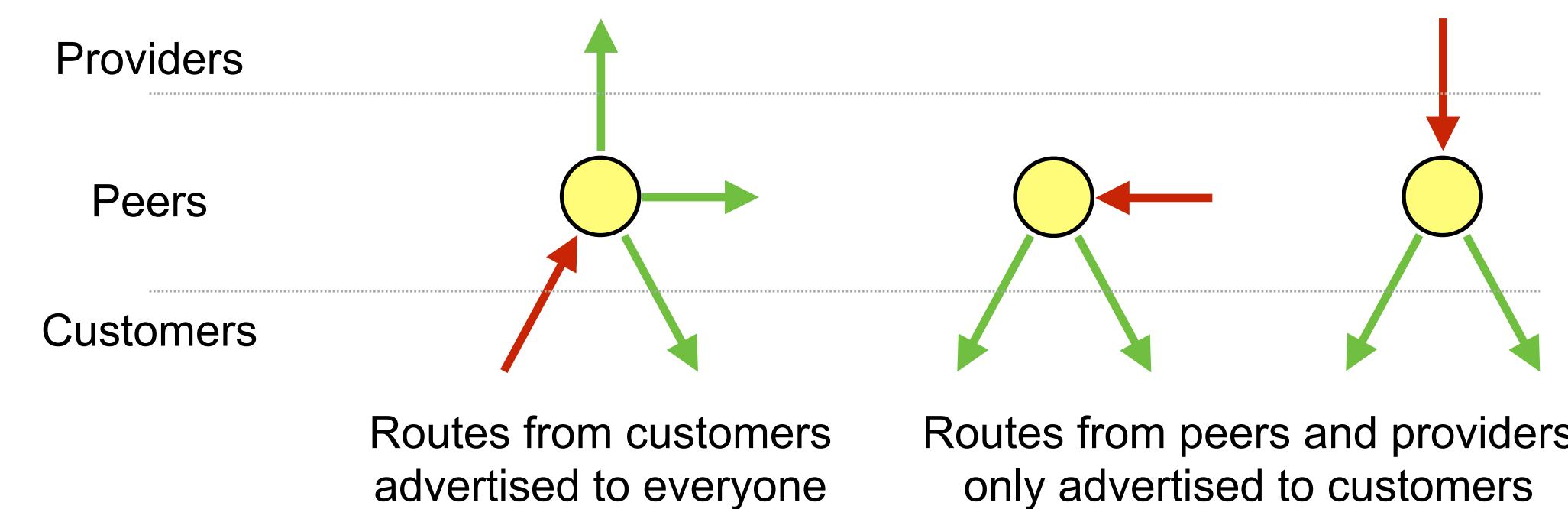
An example fragment of the AS topology graph:



Prefix	Next Hop	AS Path
...		
*	12.10.231.0/24	5459 5413 5696 7369 i
*	158.43.133.48	1849 702 701 6113 5696 7369 i
*	193.0.0.242	3333 286 5696 7369 i
*	204.212.44.128	234 266 237 3561 5696 7369 i
*>	202.232.1.8	2497 5696 7369 i
*	204.70.4.89	3561 5696 7369 i
*	131.103.20.49	1225 3561 5696 7369 i
*	62.224.0.0/19	134.24.127.3
*	134.24.127.3	1740 3561 2578 8371 i
*	194.68.130.254	5459 2529 3561 2578 8371 i
*	158.43.133.48	1849 702 701 3216 3216 3216 8371 8371 i
*	193.0.0.242	3333 286 3561 2578 8371 i
*	144.228.240.93	1239 3561 2578 8371 i
*	204.212.44.128	234 266 237 3561 2578 8371 i
*	202.232.1.8	2497 3561 2578 8371 i
*	205.238.48.3	2914 3561 2578 8371 i
*>	204.70.4.89	3561 2578 8371 i
*	131.103.20.49	1225 3561 2578 8371 i
...		

# Routing Policy in eBGP

- Each AS chooses what routes to advertise to its neighbours
- Doesn't need to advertise everything it receives
  - Usual to drop some routes from the advertisement – depends on the chosen routing policy
  - Common approach: the Gao-Rexford rules:



Lixin Gao



Jennifer Rexford

- Gao-Rexford rules specify **what routes are advertised**, not how traffic is forwarded – difference between **control plane** and **data plane**
- Resulting graph is valley-free DAG: traffic flows up to common provider, then down
- Designed to optimise **profit**, not routing efficiency – don't advertise expensive routes

# BGP Routing Decision Process (1/2)

- Each AS exchanges routing information with neighbours
  - AS-level topology information, filtered based on policies applied by each AS
  - Gives a partial view of AS-level network topology
- Each AS applies policy to choose what routes to use:
  - Choose what route to install in forwarding table for each destination prefix
  - BGP may not find a route, even if one exists, since may be prohibited by policy
  - Routes are often not the shortest AS path – shortest policy-compliant path
- Each AS applies policy to determine what routes to advertise to neighbours
  - Filter out routes the neighbour should not use (e.g., Gao-Rexford rules, and other reasons)
  - De-prioritise certain other routes
  - Tag routes for special processing where business agreements exist
  - Mapping business goals to BGP policies is poorly documented → many operational secrets

Table 2: Simplified BGP decision process [6, 24]. This table was also provided with the survey.

#	Criteria
1	Highest LocalPref
2	Lowest AS Path Length
3	Lowest origin type
4	Lowest MED
5	eBGP-learned over iBGP-learned
6	Lowest IGP cost to border router (hot-potato routing)
7	If both paths are external, prefer the path that was received first (i.e., the oldest path) [6]
8	Lowest router ID (to break ties)

Source: Phillipa Gill, Michael Schapira, and Sharon Goldberg, “A Survey of Interdomain Routing Policies”, ACM CCR, V44, N1, January 2014, p29-34

# BGP Routing Decision Process (2/2)

- BGP exchanges routes between autonomous systems
  - Policy, economic, and political concerns outweigh shortest path in many cases
  - Routing is between competitors – there is little trust and little public data
  - The BGP decision process is public – the input data is not, making it difficult to evaluate how routing decisions are made

# Inter-domain Routing

- Autonomous Systems and the AS graph
- Routing at the edge
- Routing in the core
- BGP

# Routing Security

- Internet Routing Security
- RPKI
- MANRS

# Internet Routing is not Secure

- Any AS participating in BGP routing can announce any address prefix – whether or not they own it
  - Sometimes this is accidental
  - Sometimes this is malicious
- Result is that traffic is misdirected: **BGP hijacking**
  - Security risk if traffic can be misdirected
  - Accidental hijacking is a serious stability problem for the network

The screenshot shows a web browser displaying the RIPE NCC website at [www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study](https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study). The page title is "YouTube Hijacking: A RIPE NCC RIS case study". The main content area discusses the event on Sunday, 24 February 2008, where Pakistan Telecom (AS17557) started an unauthorized announcement of the prefix 208.65.153.0/24. It details how the announcement was forwarded to the rest of the Internet, leading to YouTube traffic being redirected to Pakistan. The page also includes sections for "Introduction", "Event Timeline", and "Service Status".

<https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study>

# Resource Public Key Infrastructure (RPKI)

- RPKI is an attempt to secure Internet routing
- Lets an AS make a signed **Route Origin Authorisation (ROA)** – a digital signature, announced via BGP, proving the AS is authorised to announce a IP address prefix
  - Hierarchical delegation: RIR → provider AS → customer AS
  - Allows a router to validate whether a prefix announcement is authorised – hijacked prefixes will not have a valid signature
  - Becomes part of BGP policy: prefer signed prefixes
- As of 2019, about ~12% of IPv4 addresses are covered by prefixes with valid digital signatures, but growing rapidly
- More information:
  - <https://blog.cloudflare.com/rpki-2020-fall-update/>
  - <https://isbgpsafeyet.com/>

**RPKI is Coming of Age**  
A Longitudinal Study of RPKI Deployment and Invalid Route Origins

Taejoong Chung Rochester Institute of Technology	Emile Aben RIPE NCC	Tim Bruijnzeels NLNetLabs
Balakrishnan Chandrasekaran Max Planck Institute for Informatics	David Choffnes Northeastern University	Dave Levin University of Maryland
Bruce M. Maggs Duke University and Akamai Technologies	Alan Mislove Northeastern University	Roland van Rijswijk-Deij University of Twente and NLNetLabs
John Rula Akamai Technologies	Nick Sullivan Cloudflare	

**ABSTRACT**  
Despite its critical role in Internet connectivity, the Border Gateway Protocol (BGP) remains highly vulnerable to attacks such as prefix hijacking, where an Autonomous System (AS) announces routes for IP space it does not control. To address this issue, the Resource Public Key Infrastructure (RPKI) was developed starting in 2008, with deployment beginning in 2011. This paper performs the first comprehensive, longitudinal study of the deployment, coverage, and quality of RPKI.  
We use a unique dataset containing *all* RPKI Route Origin Authorizations (ROAs) from the moment RPKI was first deployed, more than 8 years ago. We combine this dataset with BGP announcements from more than 3,300 BGP collectors worldwide. Our analysis shows that after a gradual start, RPKI has seen a rapid increase in adoption over the past two years. We also show that although misconfigurations were rampant when RPKI was first deployed (causing many announcements to appear as invalid) they are quite rare today. We develop a taxonomy of invalid RPKI announcements, then quantify their prevalence. We further identify suspicious announcements indicative of prefix hijacking and present case studies of likely hijacks.  
Overall, we conclude that while misconfigurations still do occur, RPKI is “ready for the big screen,” and routing security can be increased by dropping invalid announcements. To foster reproducibility and further studies, we release all RPKI data and the tools we used to analyze it into the public domain.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.  
*IMC '19, October 21–23, 2019, Amsterdam, Netherlands*  
© 2019 Association for Computing Machinery.  
ACM ISBN 978-1-4503-6948-0/19/10..\$15.00  
<https://doi.org/10.1145/3355369.3355596>

1 INTRODUCTION  
The Border Gateway Protocol (BGP) is *the* mechanism that allows routers to construct routing tables across the Internet. Unfortunately, the original BGP protocol lacked many security features (e.g., authorization of IP prefix announcements), making BGP vulnerable to attacks such as prefix hijacking [3, 5, 7, 14] and route leaks [5]. To defend against these threats, the Resource Public Key Infrastructure (RPKI) was developed in April 2008 as part of the IETF in the SIDR Working Group [54]. Beta deployments followed in the years after, until all Regional Internet Registries (RIRs) started production deployment of RPKI in January 2011.  
At its core, RPKI is a hierarchical Public Key Infrastructure (PKI) that binds Internet Number Resources (INRs) such as Autonomous System Numbers (ASNs) and IP addresses to public keys via certificates. The corresponding private keys can be used by certificate holders to make attestations about these INRs—most importantly, Route Origin Authorization (ROA) objects. ROAs allow a certificate holder to authorize an ASN to announce certain IP prefixes, and are signed using the private key of a certificate covering the IP space.  
Each of the five RIRs operate their own RPKI *trust anchor* (equivalent to a root certificate in other PKIs), the private key of which is used to sign such certificates. The RIRs also offer hosted services to their members, enabling them to obtain RPKI certificates and generate ROAs.  
RPKI objects including certificates, ROAs, and supporting structures such as manifests and certificate revocation lists (CRLs) are published in so-called RPKI repositories. RPKI validation software—called Relying Party (RP) software—retrieves objects from these repositories and performs cryptographic validation of the content, ultimately producing a set of valid ROAs. A validating router can then use this set to verify incoming BGP announcements. If the router finds a BGP announcement to be in conflict with the set of valid ROAs, it should reject the announcement as (by definition) the origin AS is not authorized to announce the IP prefix(es).  
While RPKI sounds straightforward in practice it can be complex, creating many opportunities for mistakes. For example, an AS may sub-allocate an IP prefix to a customer AS without updating its ROAs or mistakenly include the wrong range of IP prefixes in a

Taejoong Chung, et al., “RPKI is coming of age: A longitudinal study of RPKI deployment and invalid route origins”, ACM Internet Measurement Conference, Amsterdam, October 2019. <https://dx.doi.org/10.1145/3355369.3355596>

# Mutually Agreed Norms for Routing Security (MANRS)



MANRS



Internet  
Society

- **Mutually Agreed Norms for Routing Security (MANRS)** project aims to improve routing security (<https://www.manrs.org>)
  - Filtering
  - Anti-spoofing
  - Coordination
  - Global validation via RKPI
  - Combination of signed route advertisements via RKPI to catch malicious attacks, with best practises for route filtering, anti-spoofing, and coordination between ISPs to protect against misconfigurations

# Routing Security

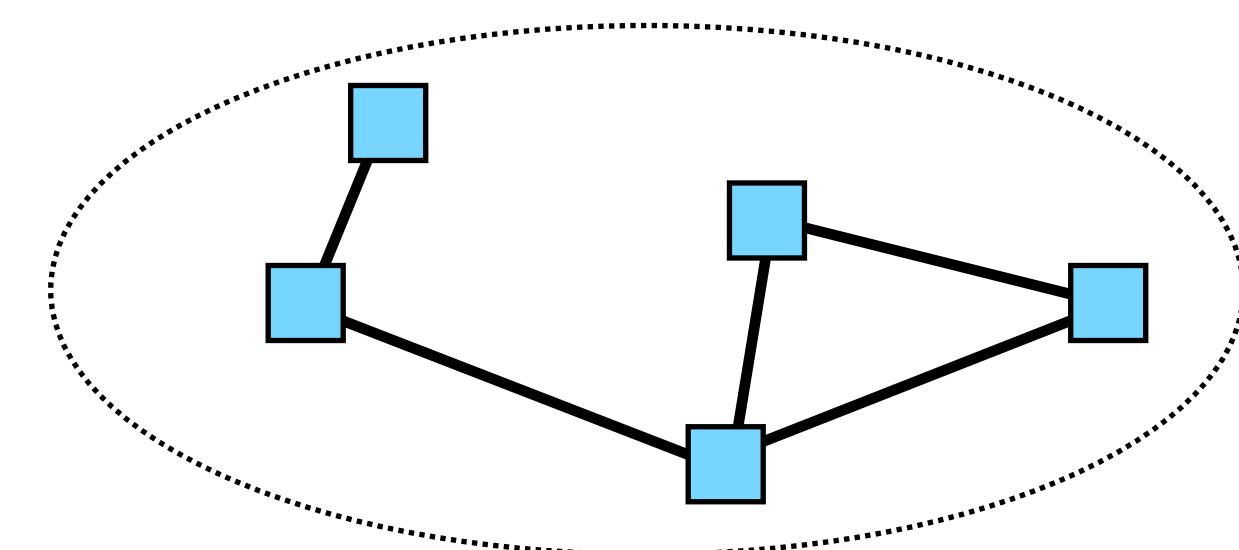
- Internet Routing Security
- RPKI
- MANRS

# Intra-domain Routing

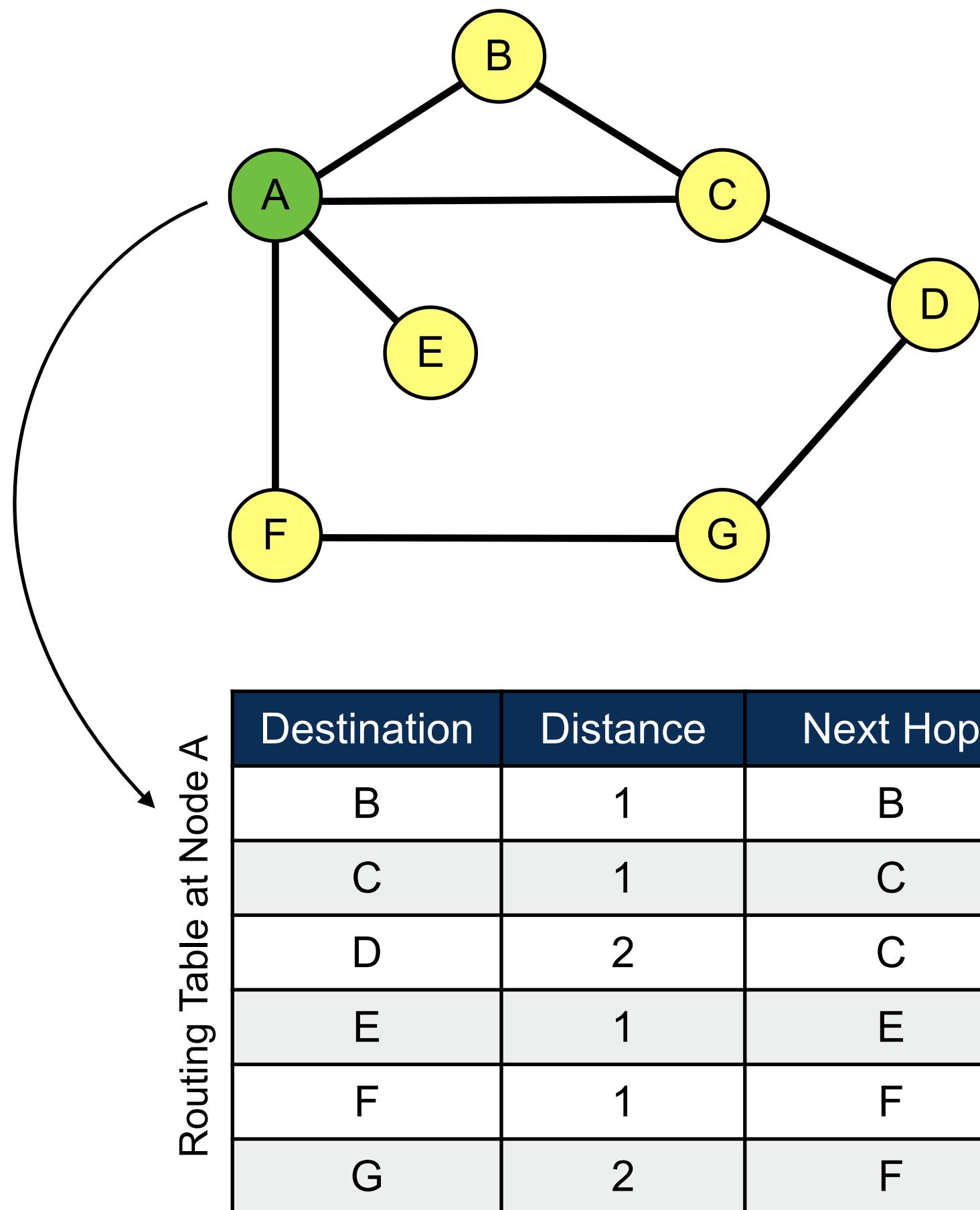
- Distance vector and link state algorithms

# Intra-domain Unicast Routing

- BGP gives information on the path to reach other networks
- How to route traffic within an AS?
  - Single trust domain
    - No policy restrictions on who can determine network topology
    - No policy restrictions on which links can be used
  - Desire efficient routing → shortest path
    - Make best use of the network you have available
  - Two approaches
    - Distance vector – the Routing Information Protocol (RIP)
    - Link state – Open Shortest Path First routing (OSPF)



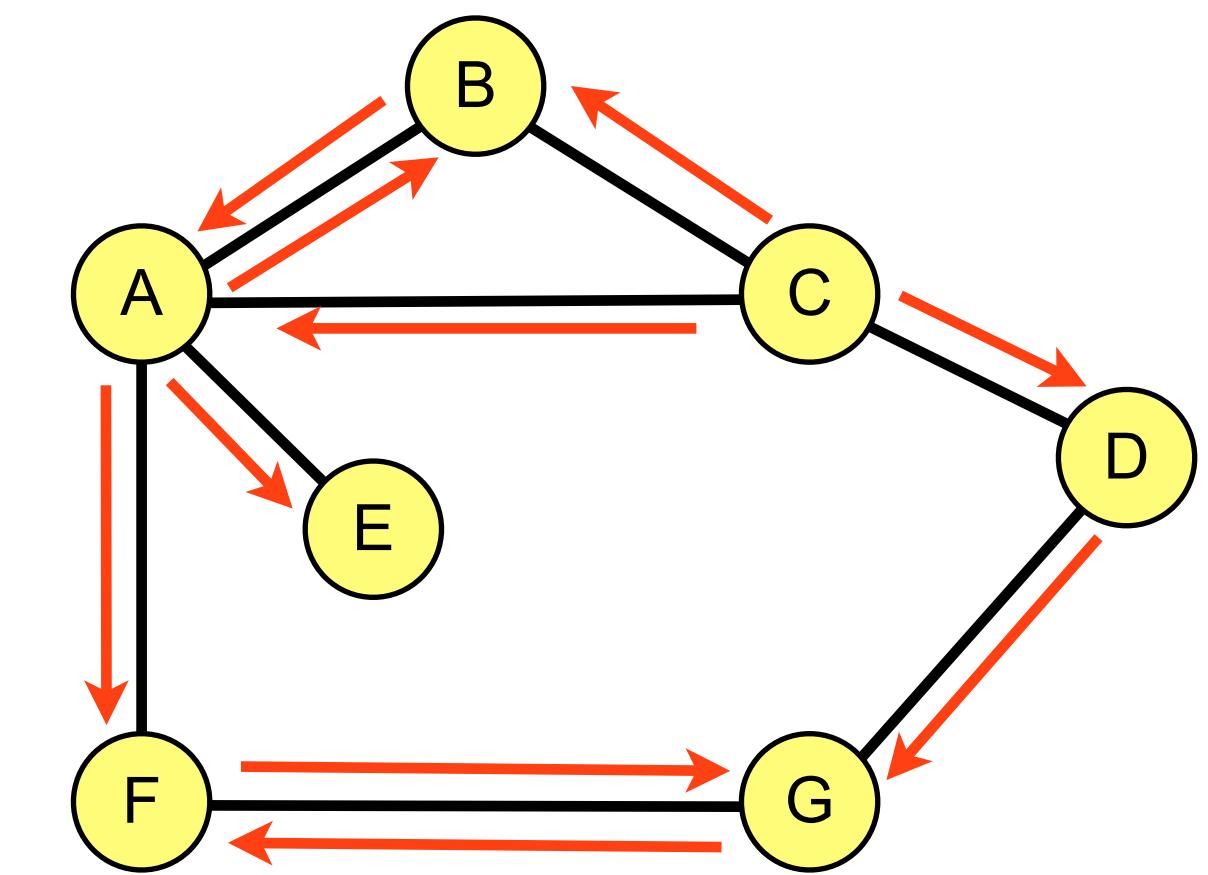
# Distance Vector Routing



- Nodes maintain a vector containing the distance to every other node in network
  - Periodically exchanged with neighbours, eventually every node know distance and next hop to every other node
- Packets forwarded on shortest path to destination
- Not widely used in practice:
  - Slow to converge in normal operation
  - Count-to-infinity problem makes this worse on failure in networks containing loops

# Link State Routing

- Nodes know the links to their neighbours, and cost of using those links – **link state** information
  - Flood this information throughout the network on startup, and whenever a link is added or removed
    - Send address of node sending the update, the list of directly connected neighbours and costs for link usage, and message sequence number
    - Sequence numbers prevent loops
  - Each node then directly calculates shortest path to every other node, uses as routing table
- Flooding link state data from all nodes ensures all nodes know the entire topology
  - Each node uses Dijkstra's shortest-path algorithm to calculate optimal route to every other node
  - Optimal is assumed to be the shortest path, by cost



# Distance Vector vs. Link State

## Distance vector routing:

- Simple to implement
- Routers only store the distance to each other node:  $O(n)$
- Suffers from slow convergence

## Link State routing:

- More complex
- Requires each router to store complete network map:  $O(n^2)$
- Much faster convergence

Slow convergence times make distance vector routing unsuitable for large networks

# Challenges in Intra-domain Routing

- How to rapidly recover from link failures?
  - Construction work surprisingly good at breaking network cables; trawlers do similar damage to undersea cables
  - Good network designs have multiple paths from source to destination
  - Must quickly notice failure and switch to backup path
    - If the link is used for telephony or video streaming, this must happen in  $<1/60^{\text{th}}$  of a second else it interrupts the speech or video playout
  - **Fast failover** – pre-calculate alternative paths to allow rapid switchover
- How to balance load across multiple paths?
  - **Equal cost multipath routing** – if two paths with the same cost, alternate traffic between them
  - Care needed: TCP uses a triple duplicate ACK to signal loss so is insensitive to small amount of packet re-ordering, but treats large amounts of reordering as loss and slows down
  - RTP applications don't care about order, as long as packets arrive before deadline



# Internet Routing

- Content distribution networks
- Inter-domain routing
- Routing security
- Intra-domain routing



University  
ofGlasgow

# Future Directions

Networked Systems (H)  
Lecture 10



Colin Perkins | <https://csperrkins.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, Mountain View, CA 94042, USA.

# Evolving the Network (1/2)

- The course has focussed on how can the Internet change and evolve to address coming challenges:
  - How to establish connections in a fragmented network?
  - How can encryption protect against pervasive monitoring and prevent transport ossification?
  - How can we reduce latency and support real-time and interactive content?
  - How can we adapt to the vagaries of wireless networks?
  - How can we identify and distribute content? How can we manage the tussle for control of the DNS and naming?
  - How can we manage interdomain routing to efficiently delivery content?
  - How can we re-decentralise the network?

# Evolving the Network (2/2)

- To address these challenges, the Internet is in the middle of a significant change
  - IPv6 as a basis for routing and forwarding evolution
  - TLS 1.3 to simplify and improve security
  - QUIC as a basis for future transport evolution, avoiding protocol ossification
  - HTTP/3 as a basis for web evolution
  - DNS over encryption (HTTPS, TLS, QUIC) for secure name resolution
  - CDNs and overlays

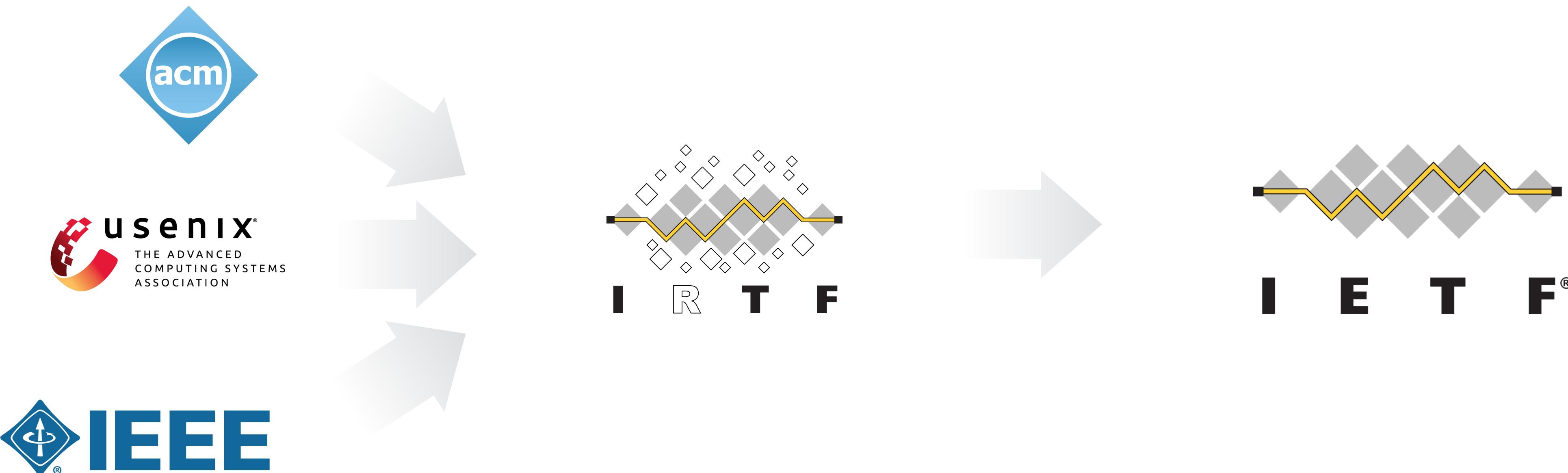


# Future Directions

- What are the longer-term challenges for the development of the Internet?
- *(Nothing in this section is assessed)*

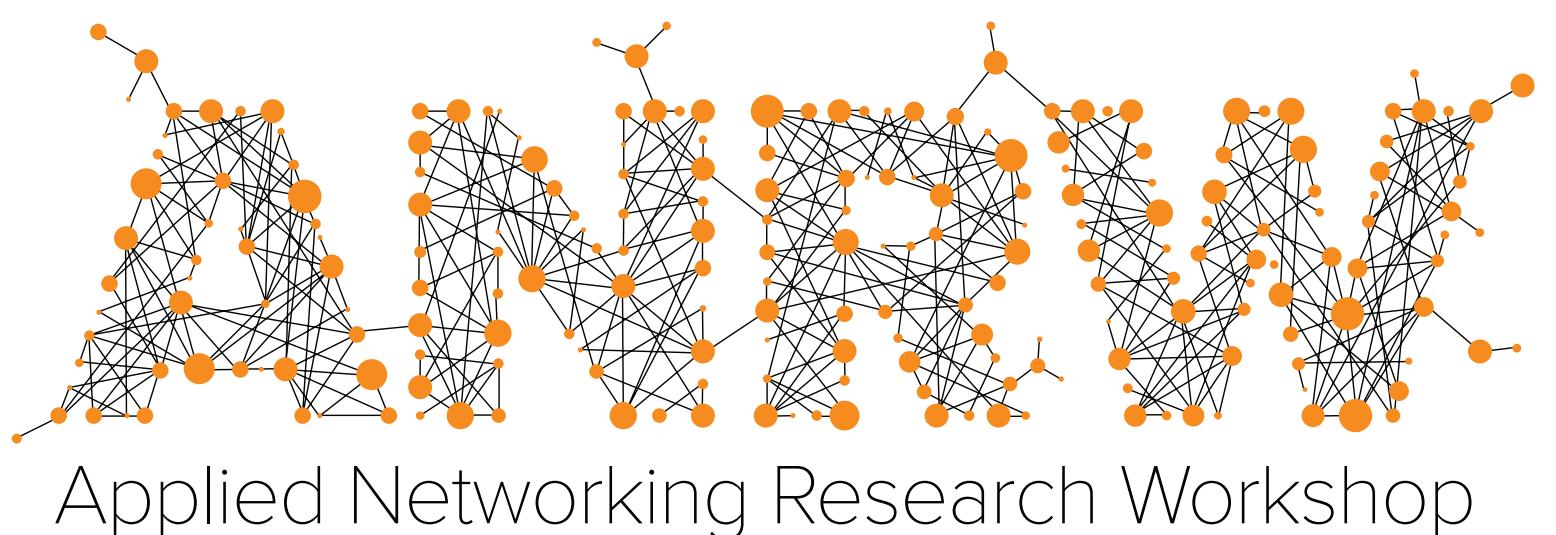
# Long-term Evolution of the Network

- The Internet Research Task Force (IRTF) promotes evolution of the Internet through applied, longer-term, research on Internet protocols, applications, architectures, and technology → <https://irtf.org/>



# IRTF Activities

- Organised around longer-term research groups
- A forum where researchers and engineers can explore the feasibility of research ideas
- A venue where researchers can learn from the engineers who build and operate the Internet – and where the standards, implementation, and operations community can learn from research



**CFRG**  
Crypto Forum Research Group

**COIN**  
Computation in the Network

**DINRG**  
Decentralised Internet Infrastructure

**GAIA**  
Global Access to the Internet for All

**HRPC**  
Human Rights Protocol Considerations

**ICCRG**  
Congestion Control

**ICNRG**  
Information-centric Networking

**MAPRG**  
Measurement and Analysis for Protocols

**NMRG**  
Network Management

**NWCRG**  
Network Coding

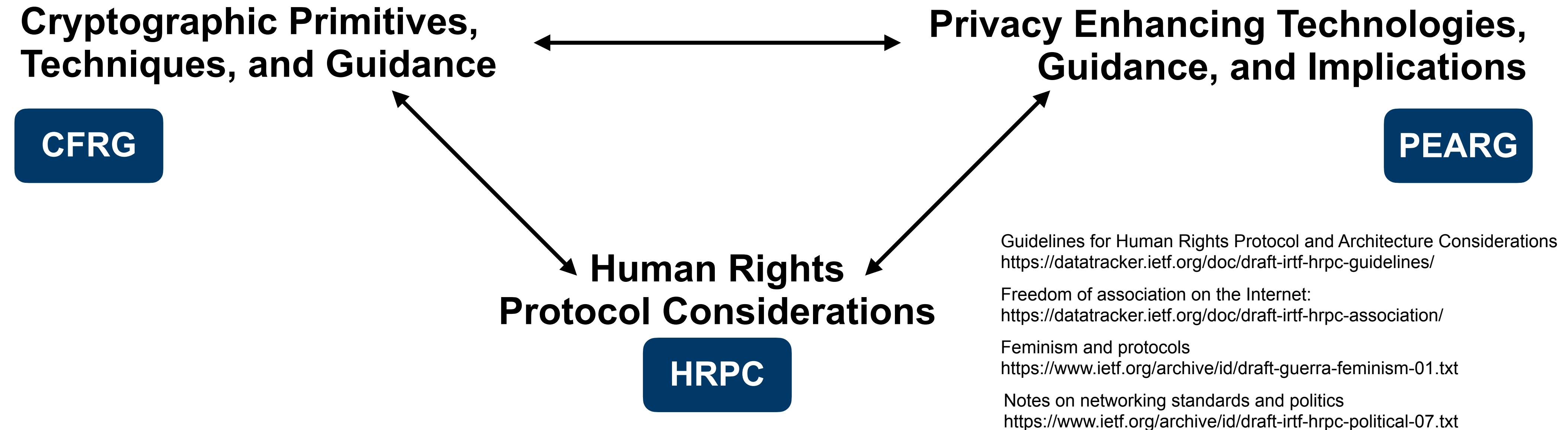
**PANRG**  
Path Aware Networking

**PEARG**  
Privacy Enhancements and Assessments

**QIRG**  
Quantum Internet

**T2TRG**  
Thing-to-Thing

# Security, Privacy, and Human Rights

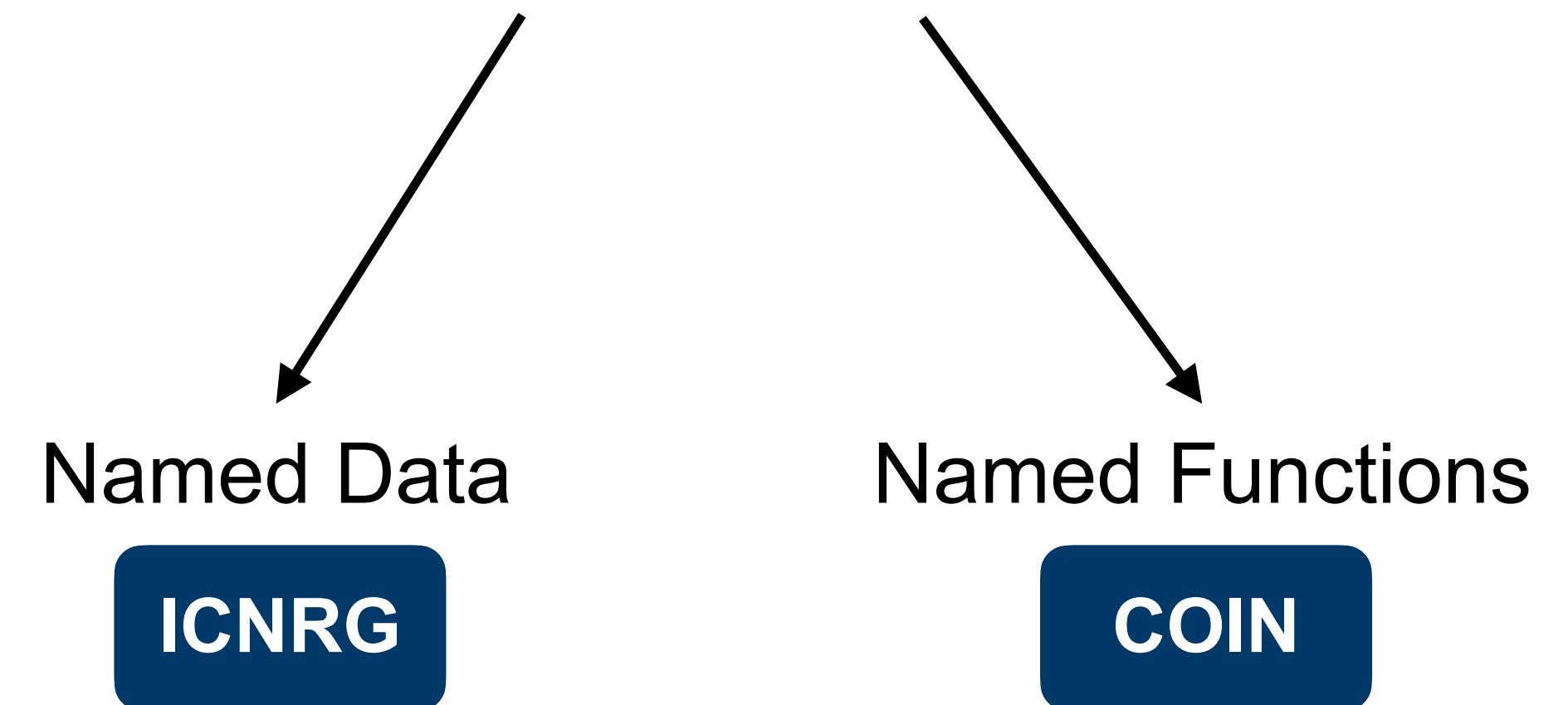


- Begin to understand how Internet protocols and standards impact human rights and privacy – at the Internet infrastructure level
- Discuss interplay between security mechanisms, privacy, and human rights; seek to raise awareness of broader societal and policy issues to the IETF community

# Computation in the Network

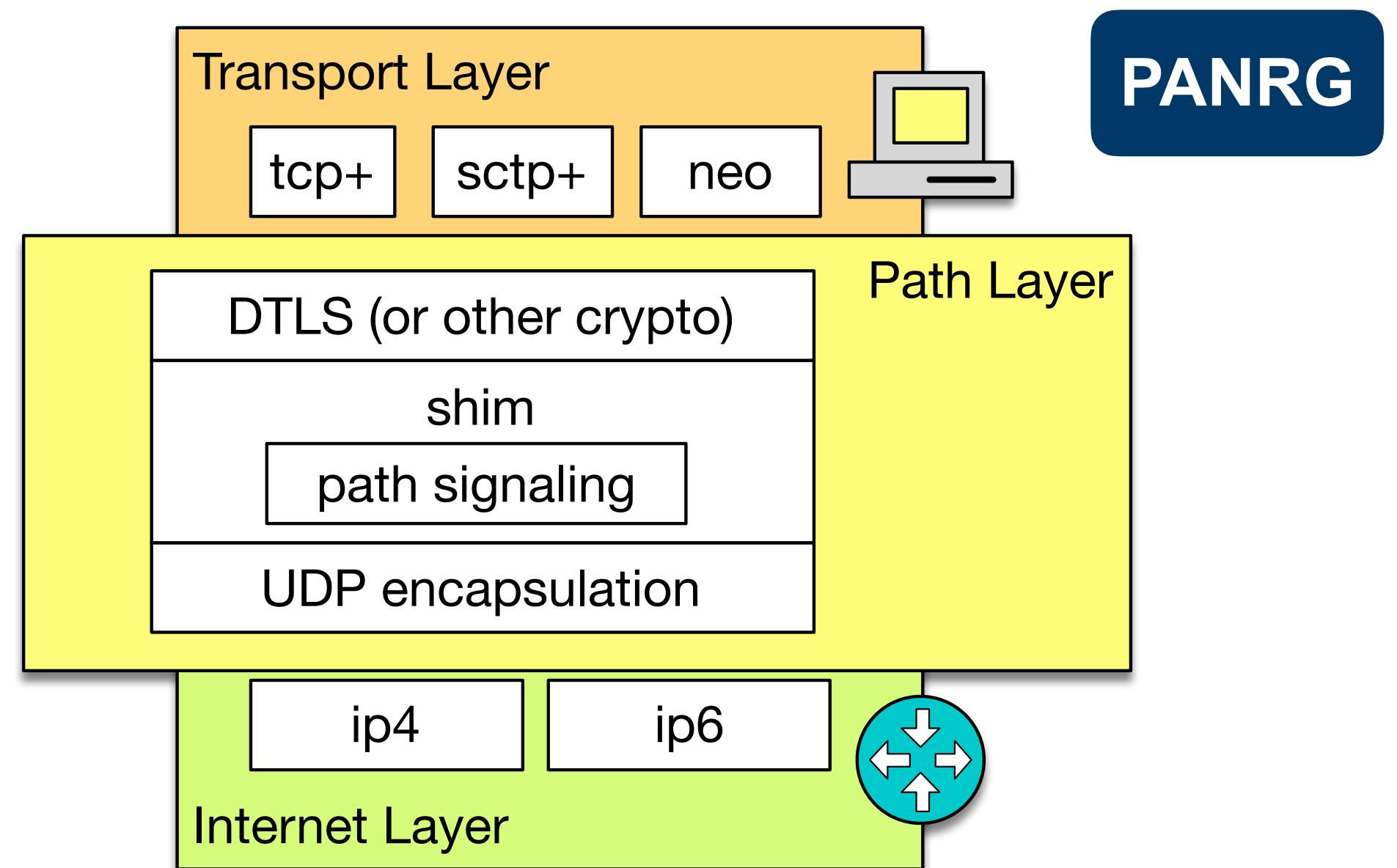
- Speculative new architectures for an internet, emphasising named data or named functions
  - Generalisation of content distribution networks and web caching infrastructure – mature work, with competing experimental implementations
  - Generalisation of lambda functions and “server-less” computation – early stage research
  - Long-term replacements for the Internet?

Current host-centric Internet



- Does it make sense to re-architect the network around content or computation? To replace IP addresses with content names?
- What are the implications of such a change for the content provider/consumer relationship – democratisation or ossification of current roles?

# Path Aware Networking



Source: Brian Trammell, presentation at IETF 96 PLUS BoF

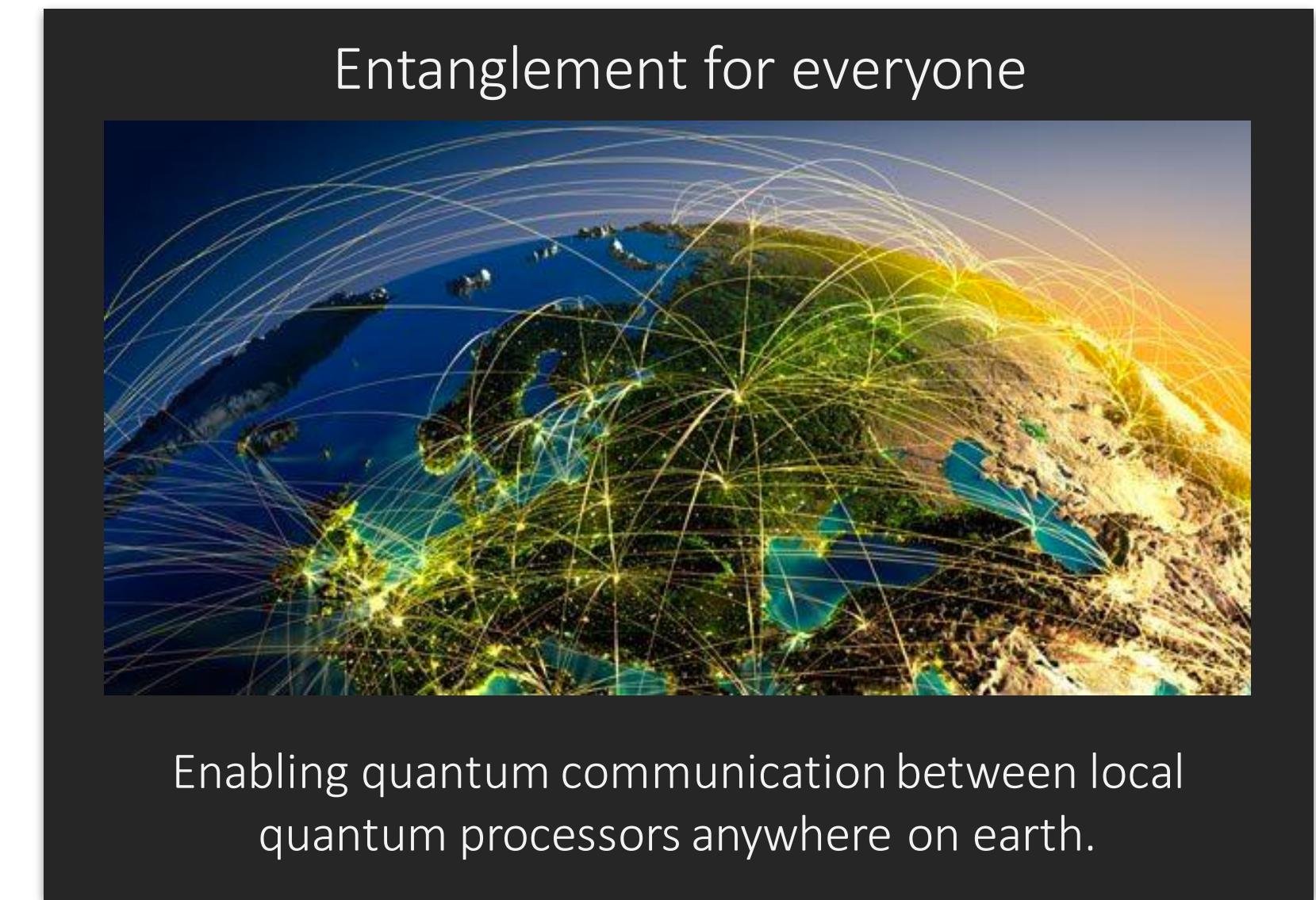
- Can we benefit from making applications and transport protocols aware of the network path taken – or by making the network path aware of the application or transport?
- Introduces a new control point for operators; questions around trust, privacy, and network neutrality are poorly understood
- IETF community seems determined to enter a standardisation phase: SRv6, APN, ...
- IRTF considering broader questions around privacy, security, path definitions, incentives

Current Open Questions in Path Aware Networking  
<https://datatracker.ietf.org/doc/draft-irtf-panrg-questions/>

Path Aware Networking: Obstacles to Deployment (A Bestiary of Roads Not Taken)  
<https://datatracker.ietf.org/doc/draft-irtf-panrg-what-not-to-do/>

# Designing the Quantum Internet

- How to establish and control inter-domain paths that can distribute entangled quantum state?
  - Quantum key distribution for security
  - Distributed quantum computation
  - **Quantum entanglement as a service**
- Architecture and approach generally well defined
  - Classical control plane
  - Managed distribution of entangled quantum state
- Entering a phase of experimentation to validate the architecture, develop prototypes



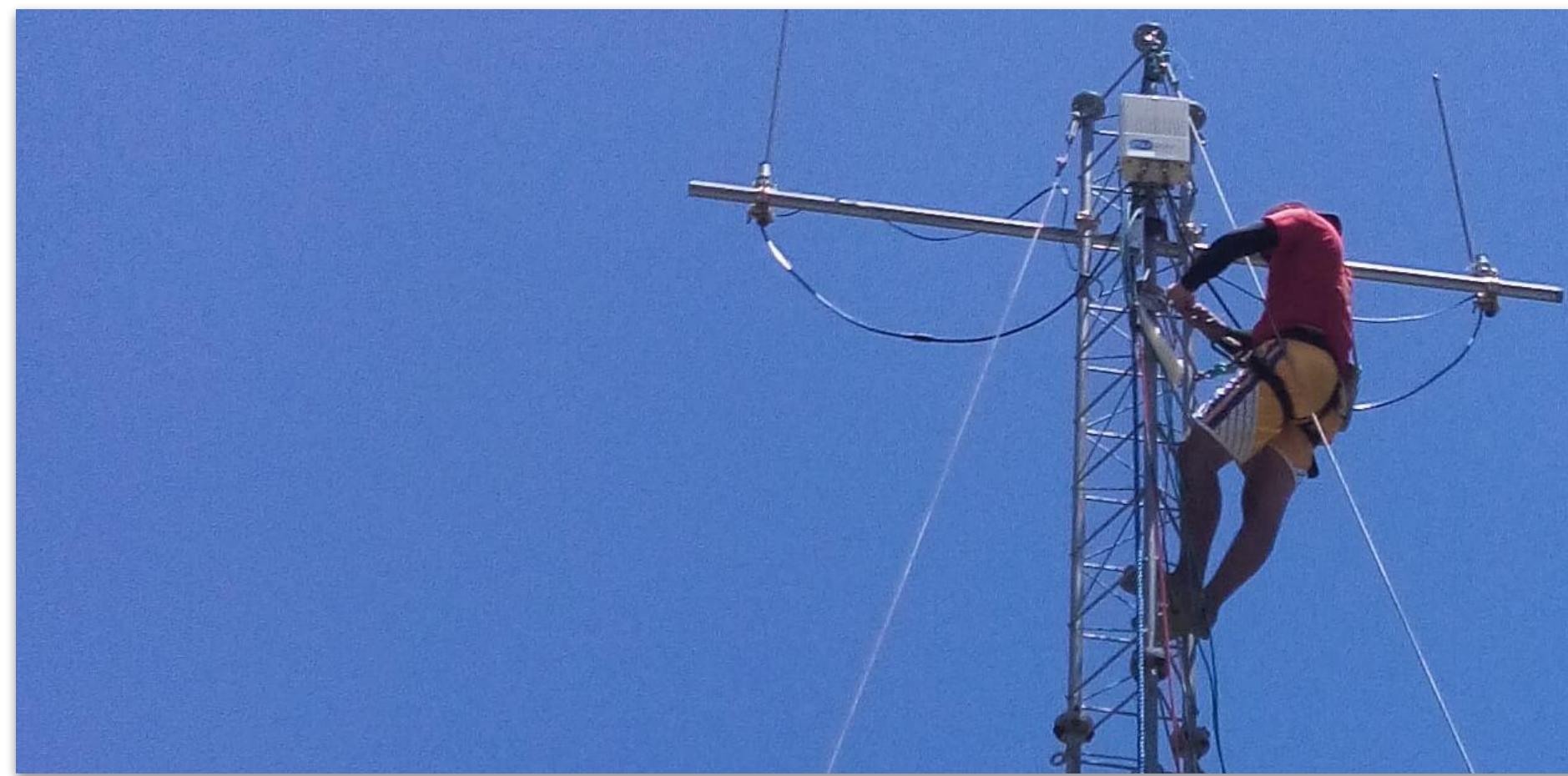
Source: Axel Dahlberg, presentation at IETF 103 QIRG meeting

**QIRG**

Architectural Principles for a Quantum Internet  
<https://datatracker.ietf.org/doc/draft-irtf-qirg-principles/>

Applications and Use Cases for the Quantum Internet  
<https://datatracker.ietf.org/doc/draft-irtf-qirg-quantum-internet-use-cases/>

# Global Access and Sustainability



Source: Maria Theresa Perez, "Community Cellular Networks in the Philippines: Experiences from the VBTS project", Presentation to IRTF GAIA RG, November 2019, <https://www.ietf.org/proceedings/106/slides/slides-106-gaia-up-vbts-philippines-00>

GAIA

- How to address the global digital divide?
  - To share experiences and best practices, foster collaboration, in building, deploying and making effective use of the Internet in rural, remote, or under-developed regions
  - To create increased visibility and interest among the wider community on the challenges and opportunities in enabling global Internet access, in terms of technology as well as the social and economic drivers for its adoption
  - To create a shared vision among practitioners, researchers, corporations, non governmental and governmental organisations on the challenges and opportunities
- Sharing expertise, raising awareness of global access challenges

# Advanced Protocol Development

MAPRG

ICCRG

NMRG

NWCRG

DINRG

T2TRG

- Measuring and understanding network behaviour
- Interfacing between research and standards community to:
  - Develop and validate new congestion control and network coding algorithms in the real world
  - Develop intent-and AI-based approaches to network management
  - Understand issues of trust- and identity- management, name resolution, resource/asset ownership, and resource discovery in decentralised infrastructure
  - Understand research challenges in IoT based on initial real-world deployment experience

# Future Directions

- The Internet is **not** finished – the protocols and fundamental design are still evolving
- With the introduction of IPv6 and QUIC, we see a significant change in network design
- Coming changes are potentially even more significant



# Wrap-Up

# Assessment

- Marks for assessed coursework will be available shortly
- Final exam, worth 80% of the marks for the course, will be held in April/May
  - Three questions, focussed on testing your understanding of networked systems
  - **Tell me what you think** – online, open book, exams focus more on deeper understanding than on bookwork
  - Past papers are on Moodle – the style of questions in the 2020 exam is most representative
- No specific revision lecture – use the Teams chat, or email, for revision questions

# Level 4 and MSci Projects

- If you are interested in Level 4 or MSci projects relating to networked systems, please contact me – [colin.perkins@glasgow.ac.uk](mailto:colin.perkins@glasgow.ac.uk)
  - Keen to work with motivated students to develop projects on topics including:
    - Improving Internet protocol standards and specifications
    - Transport protocols, real-time applications, and QUIC
    - New networking APIs in high-level languages (e.g., Rust)
    - Network measurement
  - Strong focus on interactions with IETF standards and research communities:
    - Some potential projects are strongly technical
    - Others focus heavily on the standardisation process



University  
of Glasgow