University of Glasgow

School of Computing Science

Interdomain Routing/The Transport Layer

Networked Systems (H)

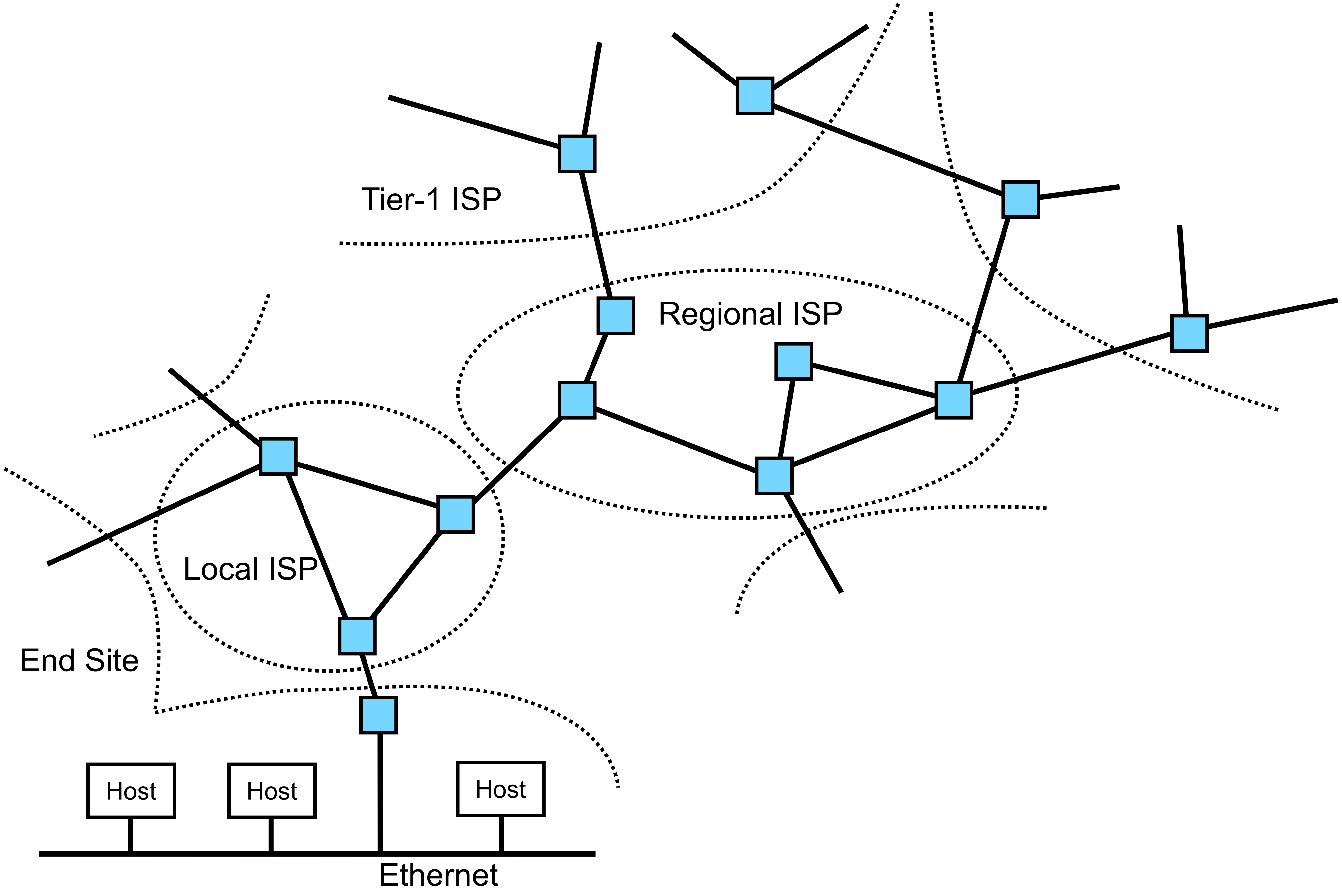
Lecture 6

Lecture Outline

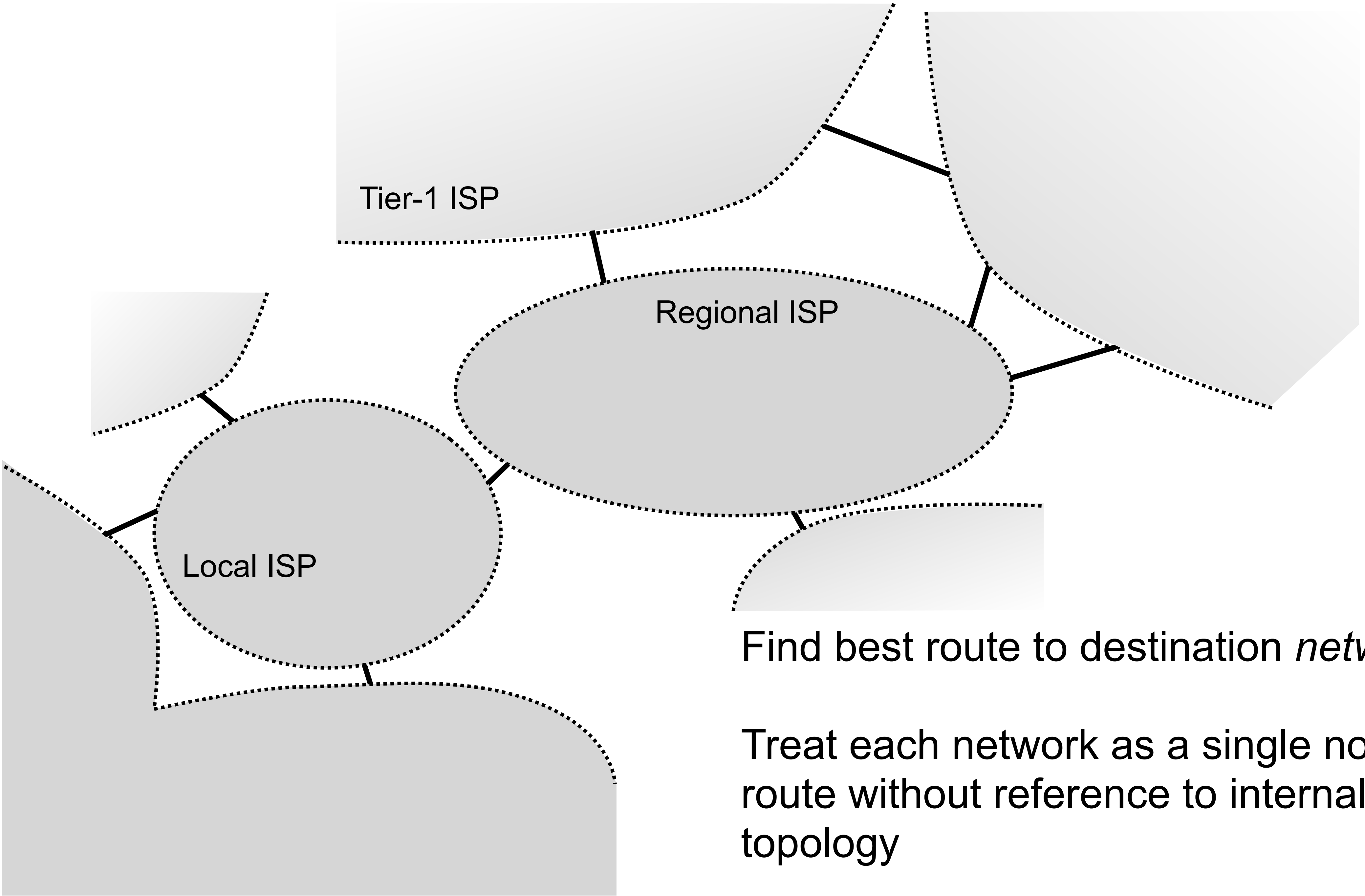
- Interdomain routing
  - Autonomous systems and the Internet AS-level topology
  - BGP and Internet routing
- The Transport Layer
  - Role of the transport layer
  - Transport layer functions
  - Transport protocols in the Internet

Inter-domain Routing

Interdomain Unicast Routing



Interdomain Unicast Routing



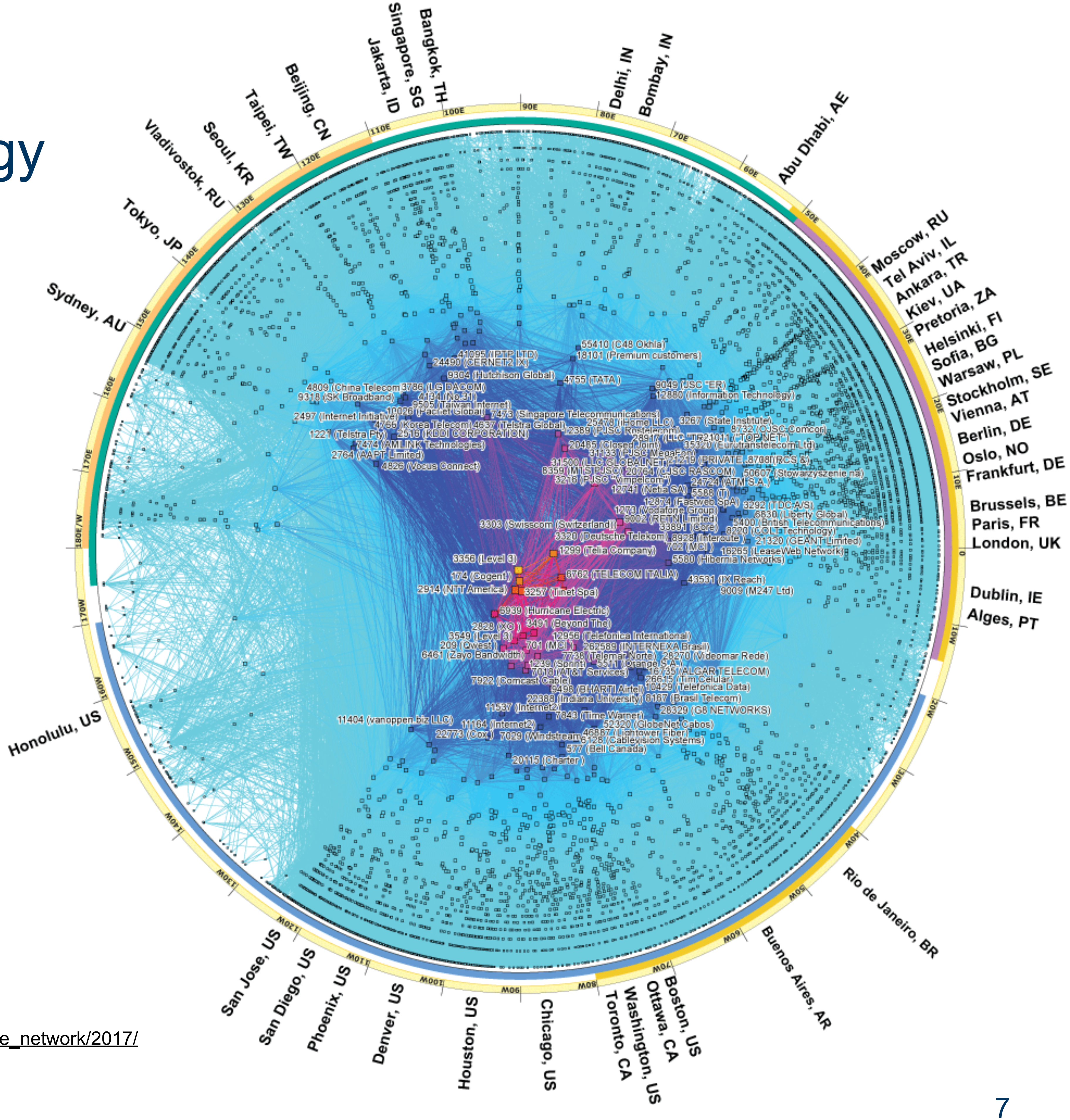
Find best route to destination *network*

Treat each network as a single node, and route without reference to internal network topology

Autonomous Systems

- Network comprised of autonomous systems (ASes)
  - Each AS is an independently administered 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; etc.
    - Each AS is identified by a unique number, allocated by the RIR
      - ~85,000 AS numbers allocated: http://bgp.potaroo.net/cidr/autnums.html (December 2017)
- Routing problem is finding best AS-level path from source AS to destination AS
  - Treat each AS as a node on the routing graph (the "AS topology graph")
  - Treat connections between ASes as edges in the graph

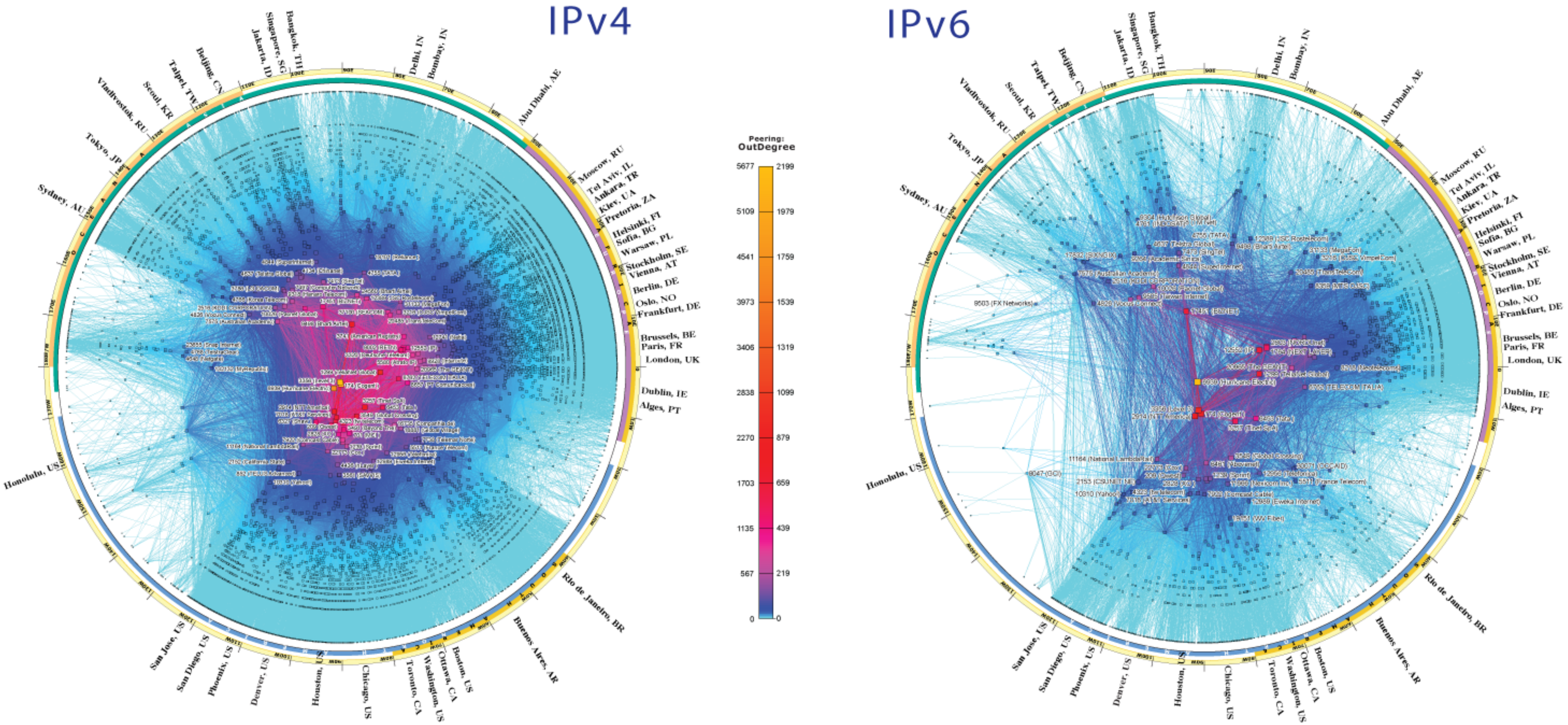
IPv4 AS Level Internet Topology



Source: CAIDA (Feb. 2017)
http://www.caida.org/research/topology/as_core_network/2017/

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

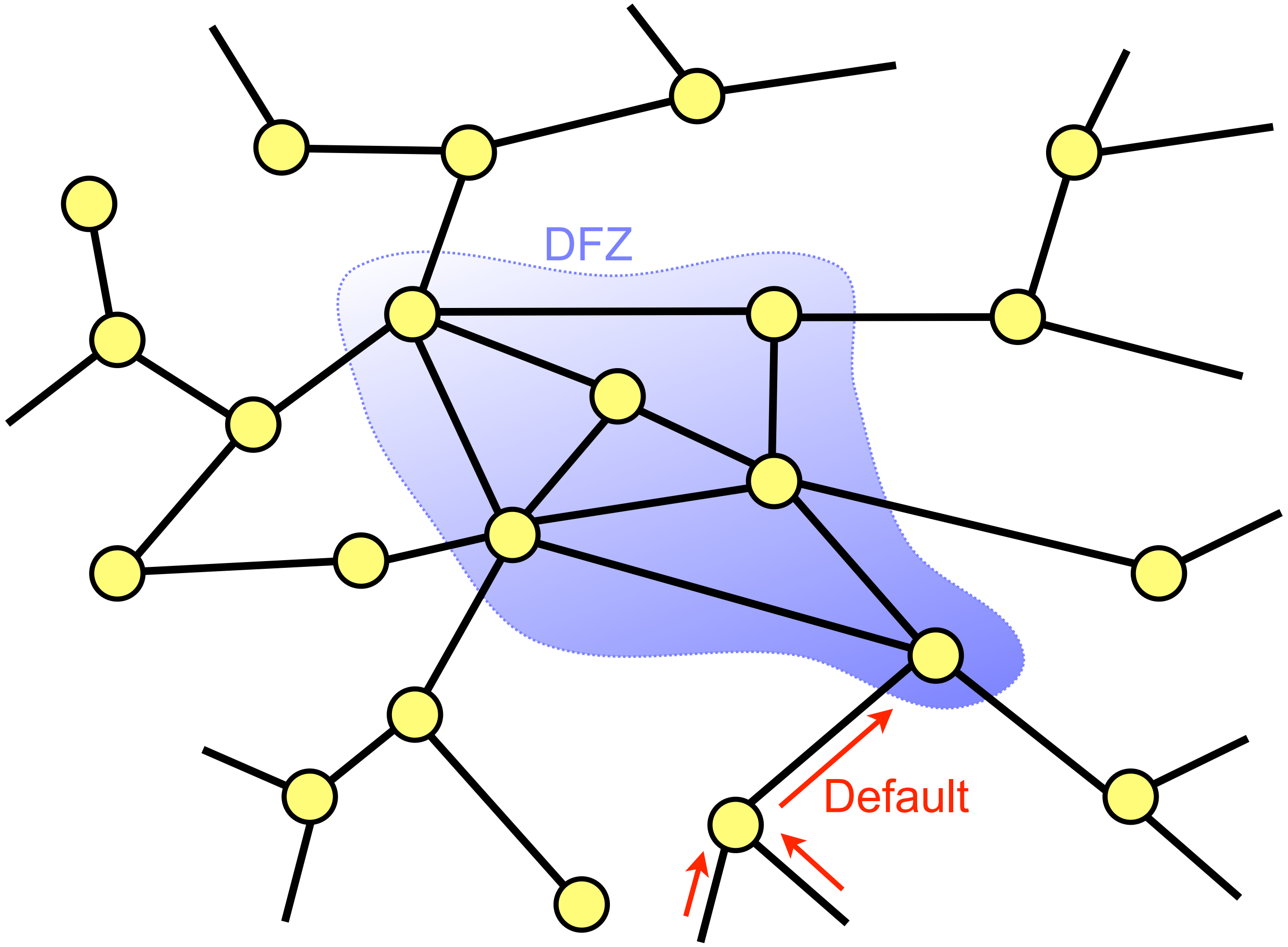
Archipelago January 2015



http://www.caida.org/research/topology/as_core_network/2015/

Default Routes and the DFZ

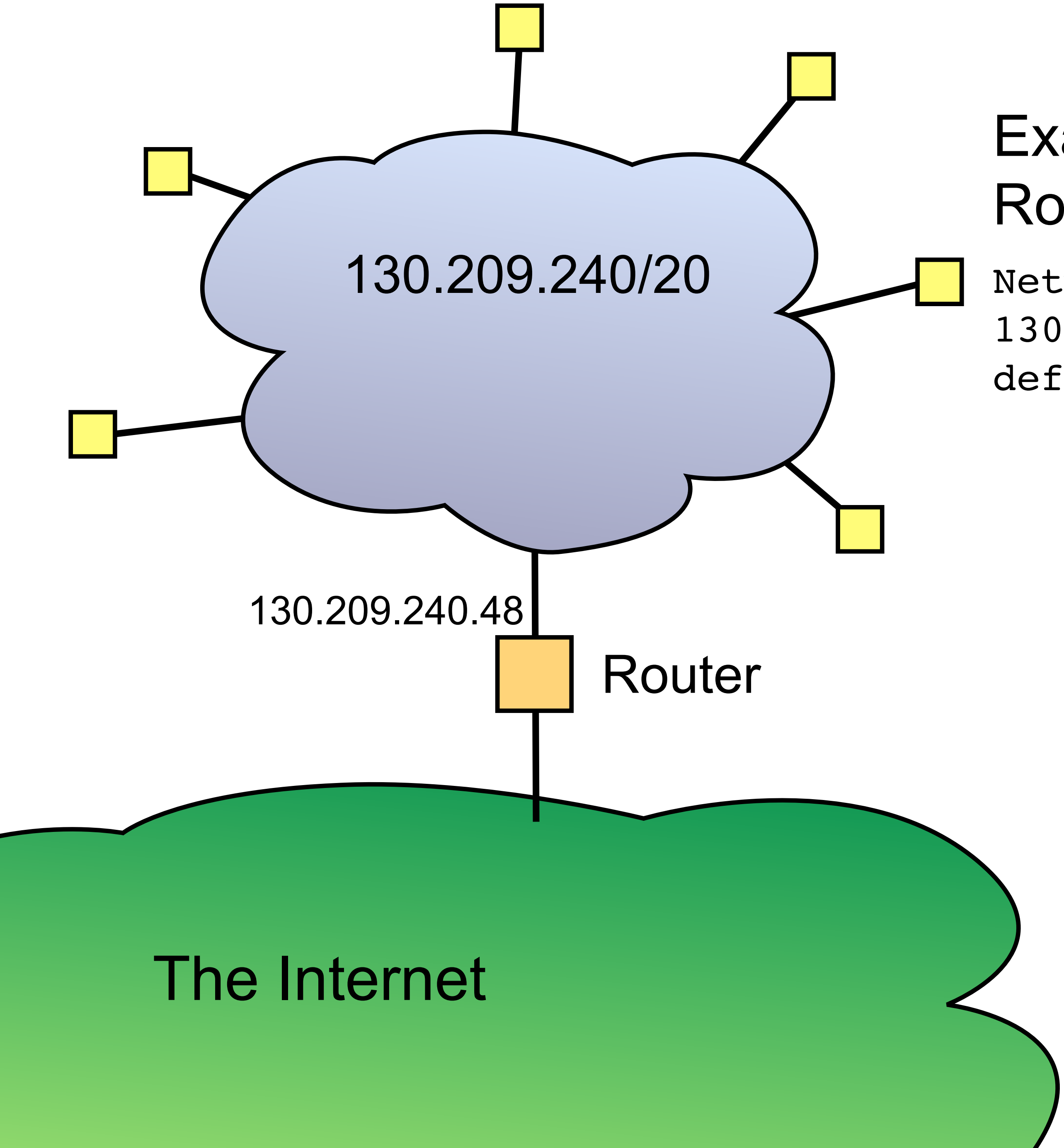
- The AS-level topology:
  - Well connected core networks
  - Sparsely connected edges, getting service from the core networks
- Edge networks can use a default route to the core
- Core networks need full routing table
  - The default free zone (DFZ)



● = AS network

╲ = Inter-AS link

Routing at the Edge



Example:
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 DFZ

- Core networks are well-connected, must know about every other network
  - The default free zone where there is no default route
  - Route based on policy, not necessarily shortest path
    - Use AS x in preference to AS y
    - Use AS x only to reach addresses in this range
    - Use the path that crosses the fewest number of ASes
    - Avoid ASes located in that country
  - Requires complete AS-level topology information

Routing Policy

- Interdomain routing is between competitors
  - ASes are network operators and businesses that compete for customers
  - Implication: an AS is unlikely to trust its neighbours

- Routing must consider policy
  - Policy restrictions on who can determine your topology
  - Policy restrictions on which route data can follow
  - Prefer control over routing, even if that means data doesn't necessarily follow the best (shortest) path – the shortest path might pass through a competitor's network, or a country you politically disagree with, or over an expensive link…

Border Gateway Protocol

- Interdomain routing in the Internet uses the Border Gateway Protocol (BGP)
    - External BGP (eBGP) used to exchange routing information between ASes
        - Neighbouring ASes configure an eBGP session to exchange routes
        - Runs over a TCP connection between routers; exchanges knowledge of the AS graph topology
        - Used to derive “best” route to each destination; installed in routers to control forwarding
    - Internal BGP (iBGP) propagates routing information to routers within an AS
        - The intra-domain routing protocol handles routing within the AS
        - iBGP distributes information 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

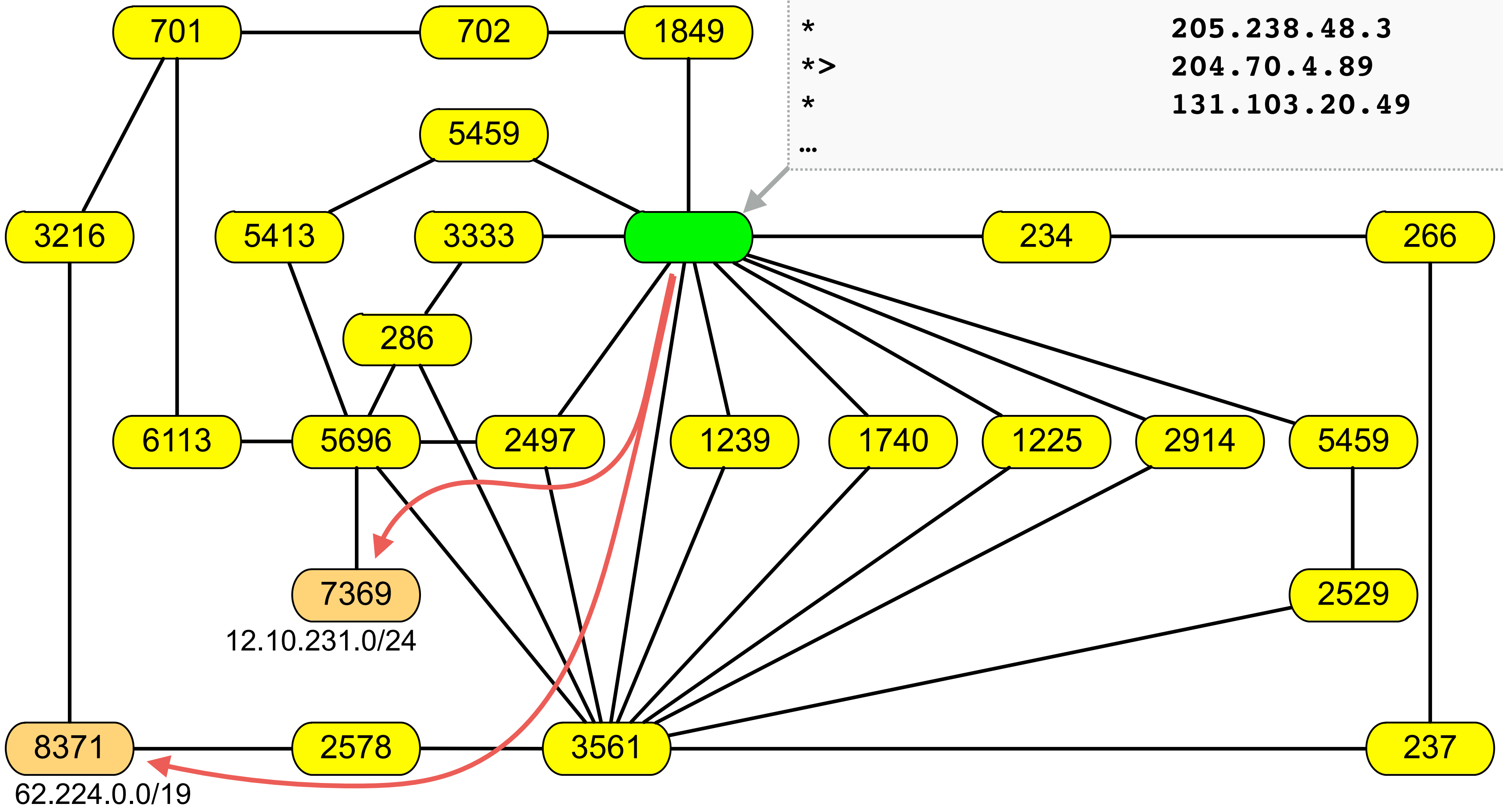
```
   Prefix           Next Hop          AS Path
…
*  12.10.231.0/24   194.68.130.254    5459 5413 5696 7369 i
*                   158.43.133.48     1849 702 701 6113 5696 7369 i
*                   193.0.0.242       3333 286 5696 7369 i
*                   204.212.44.128    234 266 237 3561 5696 7369 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 and 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

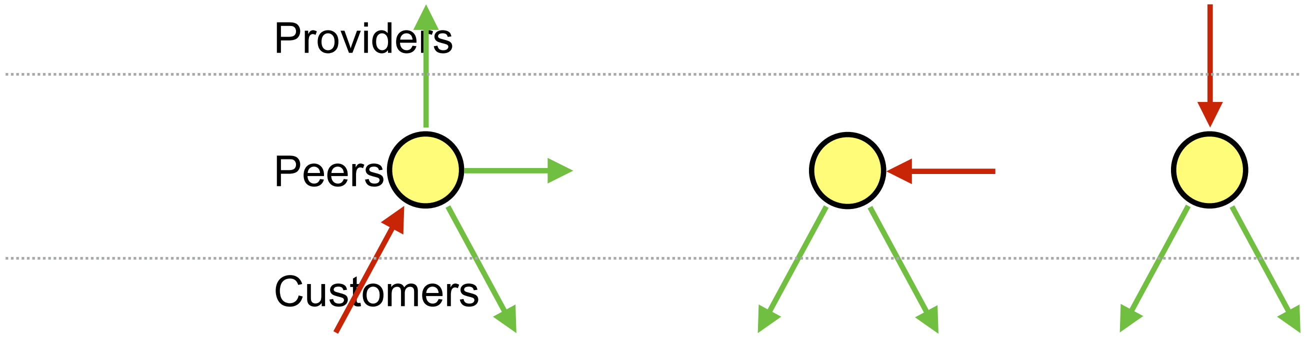
An example fragment of the AS topology graph:

```
   Prefix           Next Hop          AS Path
…
*  12.10.231.0/24   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
…
```



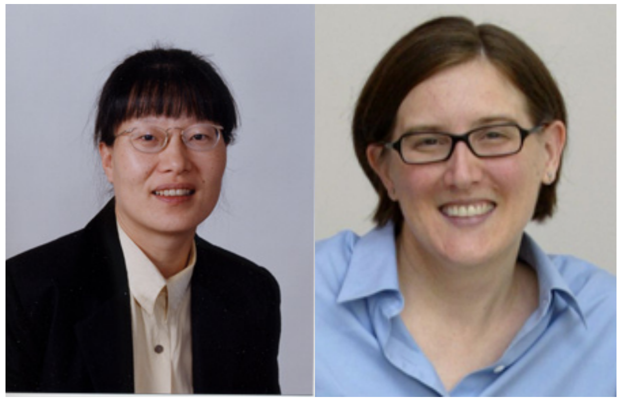
Routing Policy in eBGP

- 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:



Routes from customers advertised to everyone

Routes from peers and providers only advertised to customers



Lixin Gao

Jennifer Rexford

Ensures the AS graph is a valley-free DAG (recommended, but not required, policy)

BGP Routing Decision Process

- BGP routers receive path vectors from neighbouring ASes giving possible routes to prefixes
  - Filtered based on the policy of each AS in the path from the source
- BGP decision process is complex and policy-driven
  - Choose what route to install for destination prefix in forwarding table based on multiple criteria – policy, shortest path, etc.
  - BGP doesn't always find a route, even if one exists, as may be prohibited by policy
  - Routes are often not the shortest AS path
  - Mapping business goals to BGP policies is a poorly documented process, with many operational secrets

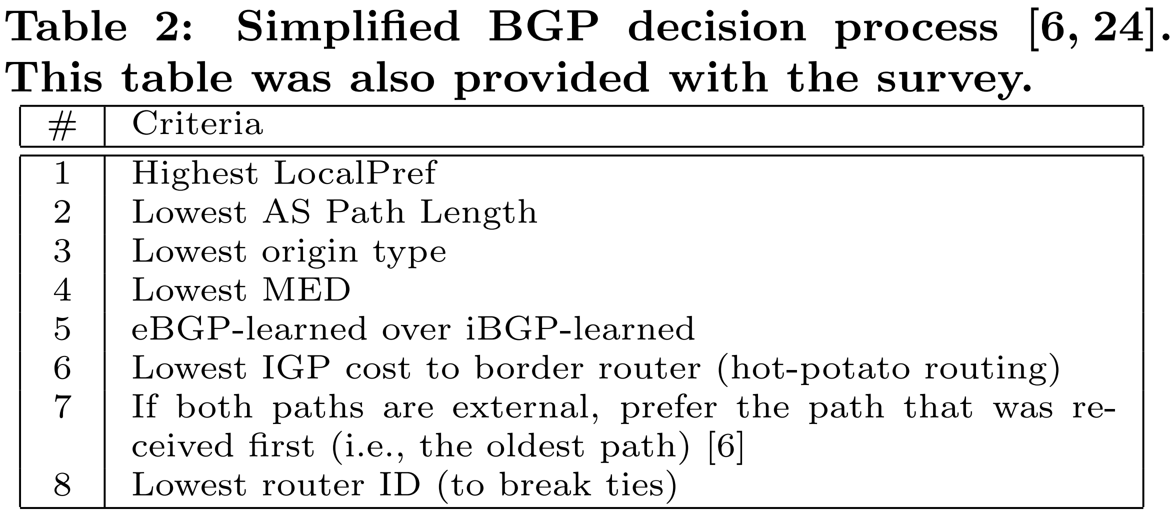


Table 2: Simplified BGP decision process [6, 24]. This table was also provided with the survey.

#	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

Summary

- The interdomain routing problem
  - Autonomous systems
  - Routing on the AS graph
  - Trust and policy constraints
- Interdomain routing in the Internet
  - BGP

The Transport Layer

The Transport Layer

- Role of the transport layer
- Transport layer functions
- Transport protocols in the Internet
    - TCP, UDP, DCCP, and SCTP
    - Deployment considerations

Role of the Transport Layer

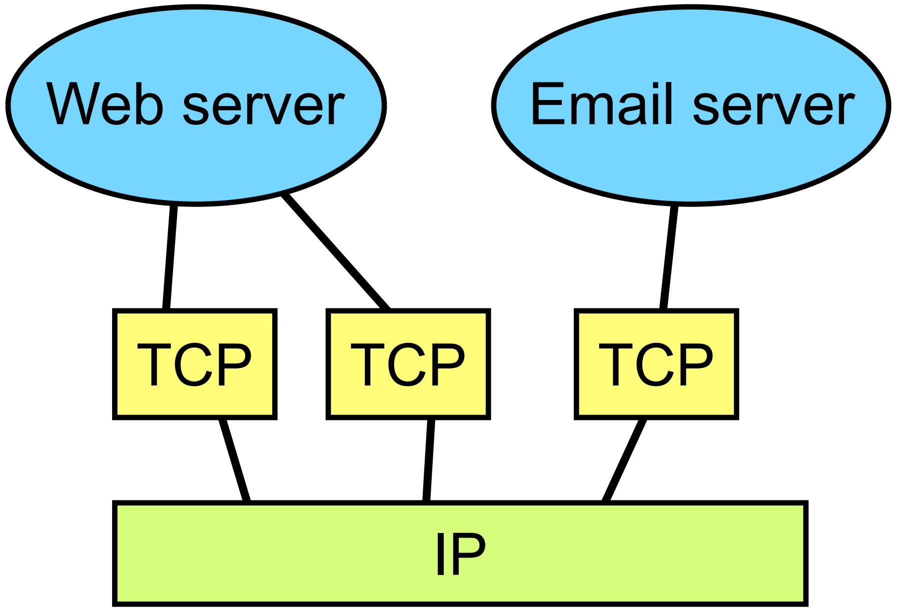
- Isolate upper layers from the network layer
  - Hide network complexity; make unreliable network appear reliable; enhance QoS of network layer
- Provide a useful, convenient, easy to use service
  - An easy to understand service model
  - An easy to use programming API
    - The Berkeley sockets API – very widely used by application programmers
    - Compare to network layer API – usually hidden in operating system internals

Transport Layer Functions

- Transport layer provides the following functions:
  - Addressing and multiplexing
  - Reliability
  - Framing
  - Congestion control
- Operates process-to-process, not host-to-host

Addressing and Multiplexing

- The network layer address identifies a host
- The transport layer address identifies a user process – a *service* – running on a host
- Provides a demultiplexing point
  - Each service has a unique transport layer address

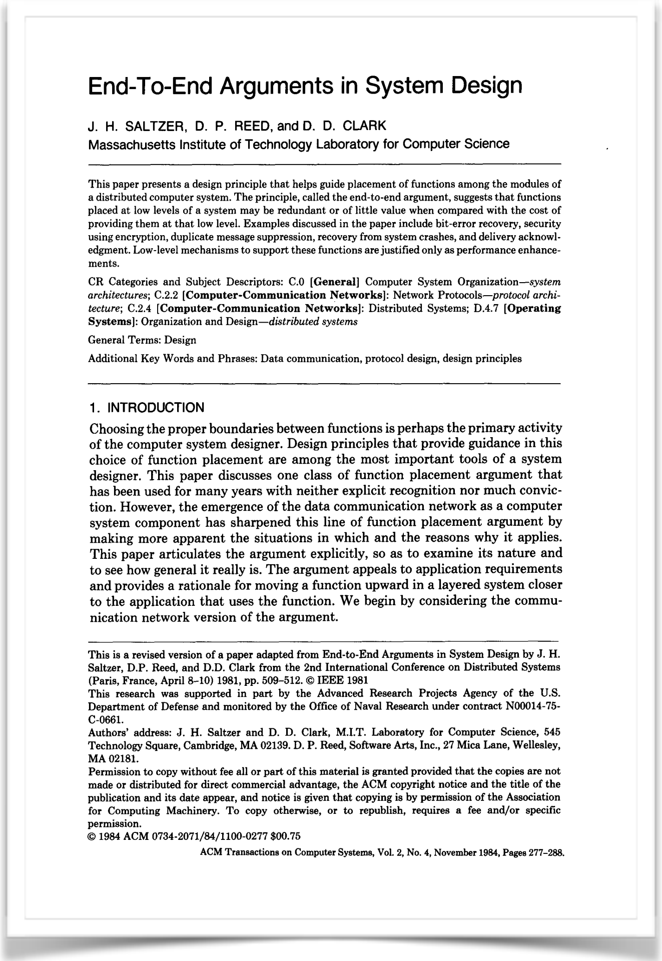


Reliability

- Network layer *is* unreliable
  - Best effort packet switching in the Internet
  - But even nominally reliable circuits may fail
- Transport layer enhances the quality of service provided by the network, to match application needs
  - Appropriate *end-to-end* reliability

The End-to-End Argument

- Is it better to place functionality within the network or at the end points?
  - Only put functions that are absolutely necessary within the network, leave everything else to end systems
    - Example: put reliability in the transport layer, rather than the network
    - If the network is not guaranteed 100% reliable, the application will have to check the data anyway → don't check in the network, leave to the end-to-end transport protocol, where the check is visible to the application
  - 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.

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

J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design.
ACM Transactions on Computer Systems, 2(4):277–288, November 1984.
http://dx.doi.org/10.1145/357401.357402

Transport Layer Reliability

- Different applications need different reliability
  - Email and file transfer → all data must arrive, in the order sent, but no strict timeliness requirement
  - Voice or streaming video → can tolerate a small amount of data loss, but requires timely delivery
- Implication for network architecture:
  - Network layer provides timely but unreliable service
  - Transport layer protocols add reliability, if needed

Framing

- Applications may wish to send structured data
- Transport layer responsible for maintaining the boundaries
    - Transport must *frame* the original data, if this is part of the service model

Congestion and Flow Control

- Transport layer controls the application sending rate
  - To match rate at which network layer can deliver data – *congestion control*
  - To match rate at which receiver can process the data – *flow control*
- Must be performed end-to-end, since only end points know characteristics of entire path

Congestion and Flow Control

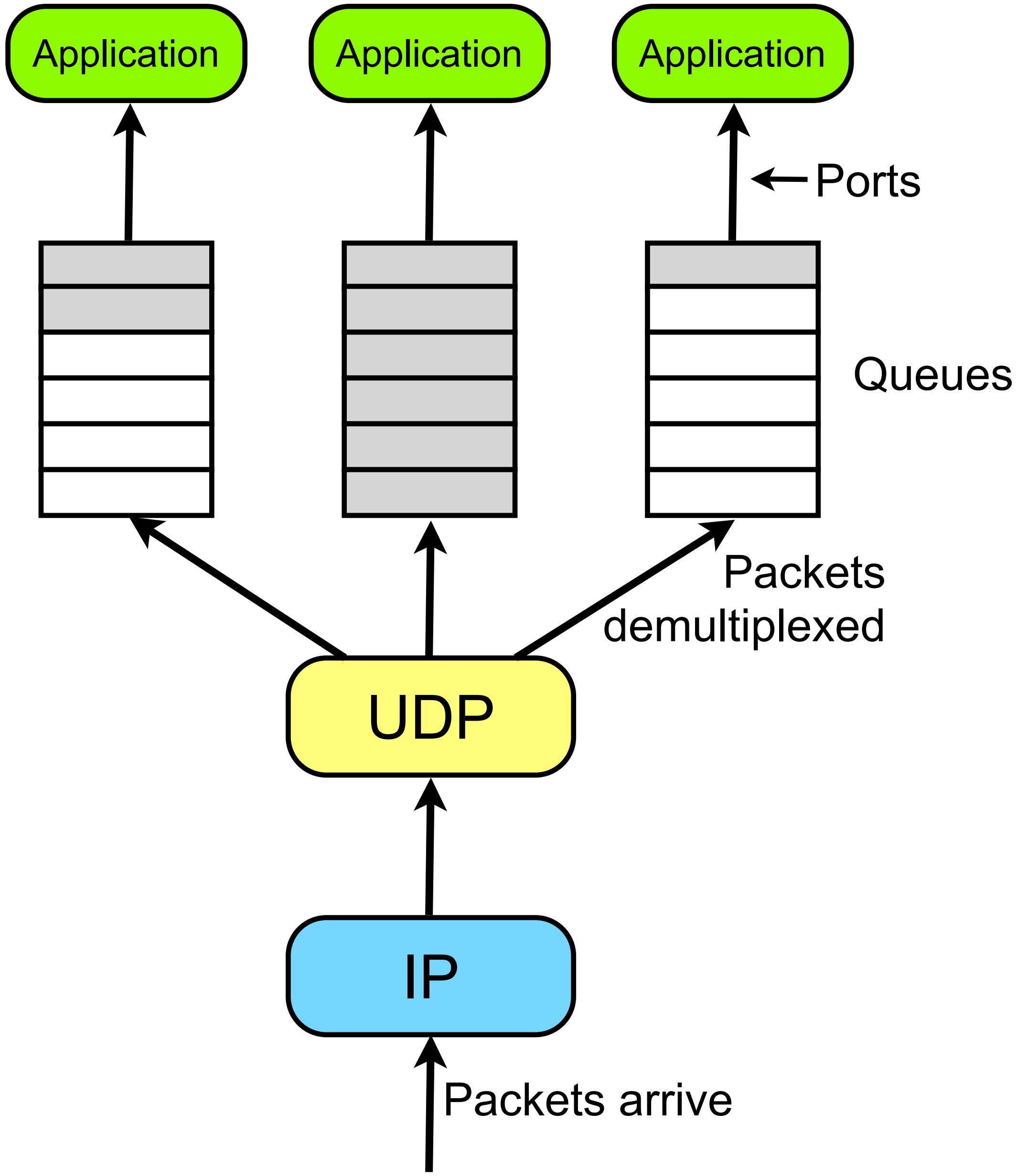
- Different applications have different needs for congestion control
  - Email and file transfer → elastic applications; faster is better, but don't care about actual sending rate
  - Voice or streaming video → inelastic applications; have minimum and maximum sending rates, and care about the actual sending rate
- Want range of congestion control algorithms at transport layer; within the network constraints

Internet Transport Protocols

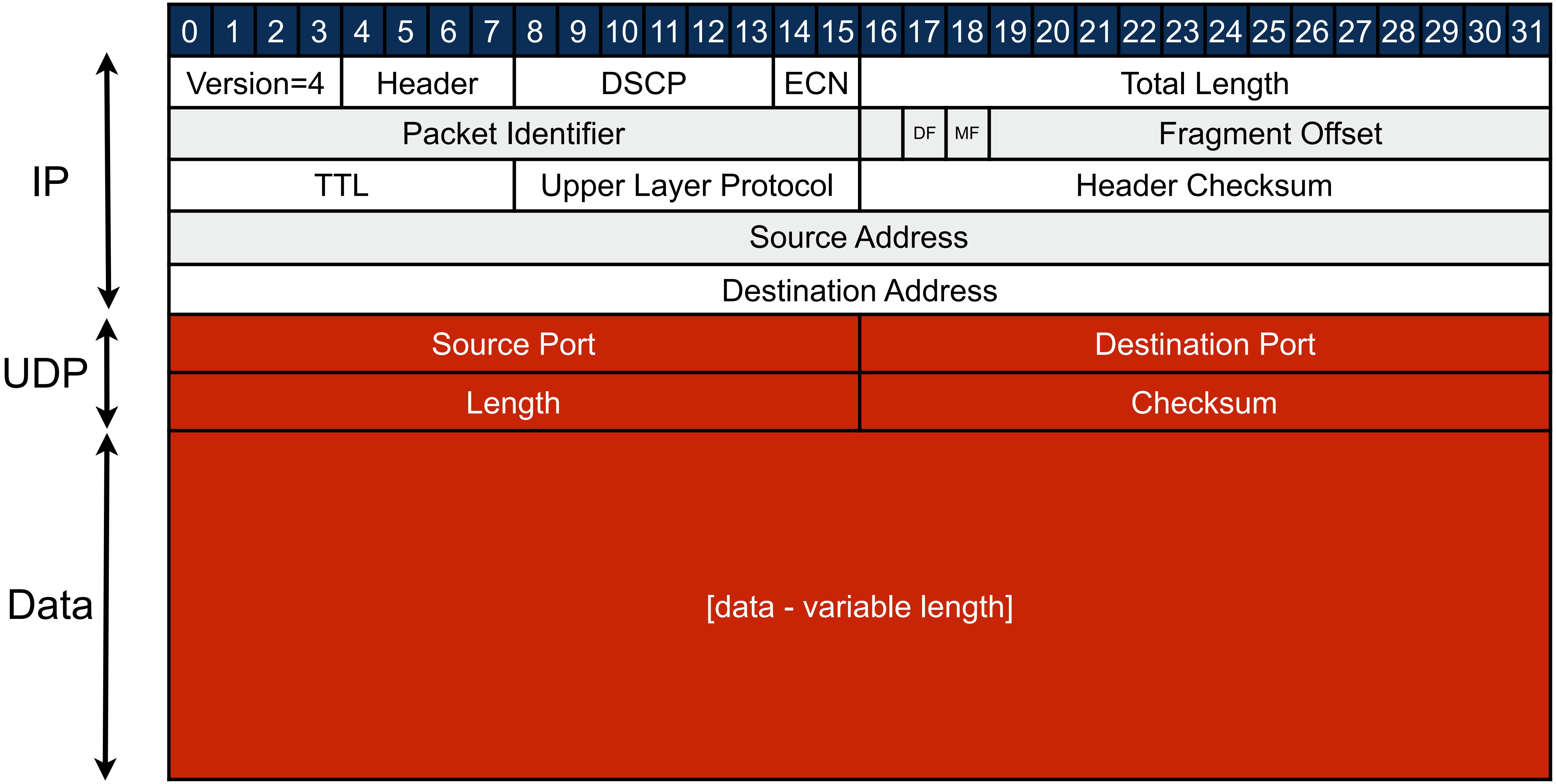
- The Internet Protocol provides a common base for various transports
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
  - Datagram Congestion Control Protocol (DCCP)
  - Stream Control Transmission Protocol (SCTP)
- Each makes different design choices

UDP: User Datagram Protocol

- Simplest transport protocol
- Exposes raw IP service to applications
  - Connectionless, best effort packet delivery: framed, but unreliable
  - No congestion control
- Adds a 16 bit *port* number to identify services



UDP Packet Format



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

IP

Version=4 | Header | DSCP | ECN | Total Length

Packet Identifier | DF | MF | Fragment Offset

TTL | Upper Layer Protocol | Header Checksum

Source Address

Destination Address

UDP

Source Port | Destination Port

Length | Checksum

Data

[data - variable length]

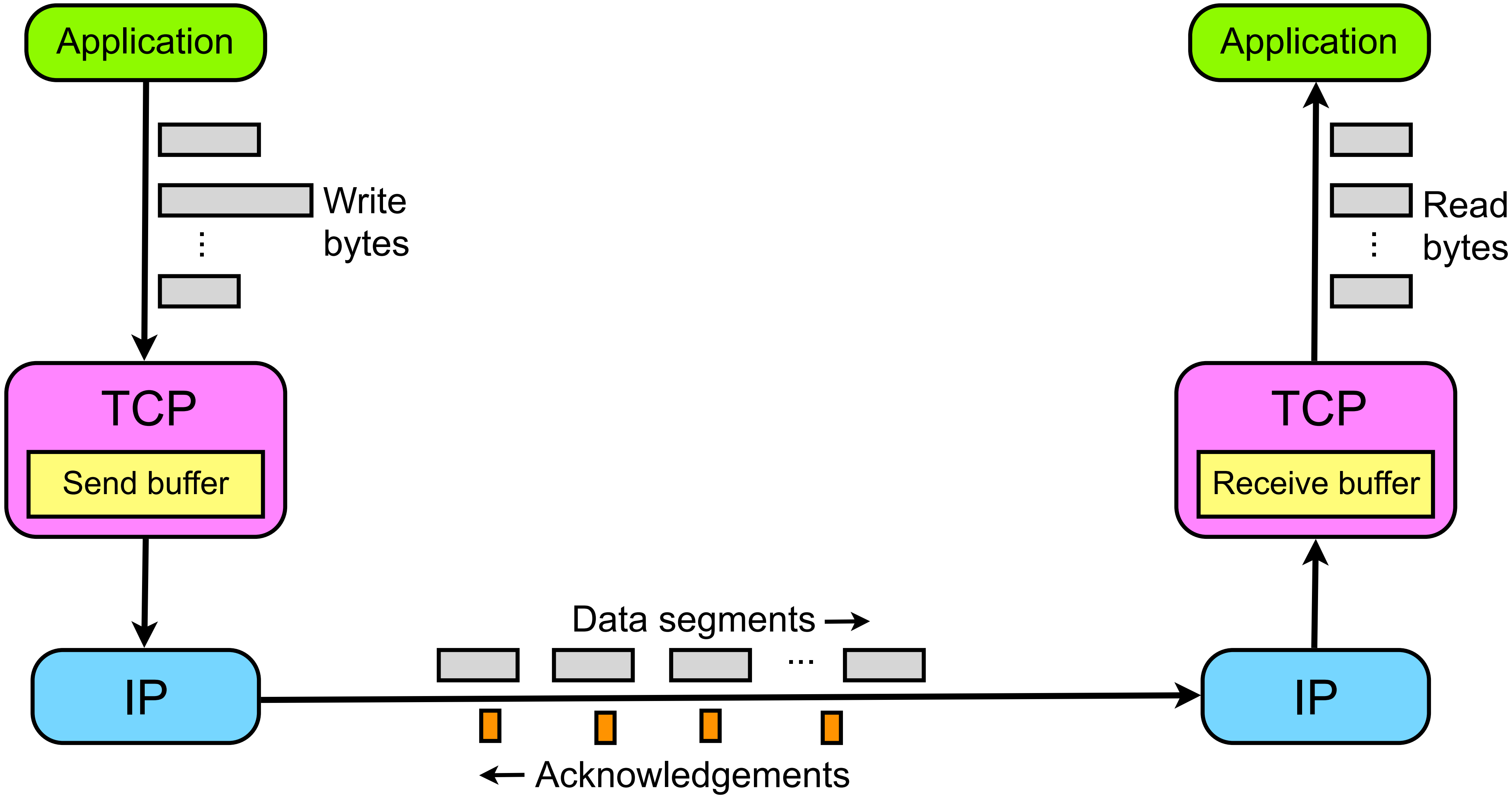
UDP Applications

- Useful for applications that prefer timeliness to reliability
  - Voice-over-IP
  - Streaming video
- Must be able to tolerate some loss of data
- Must be able to adapt to congestion in the application layer

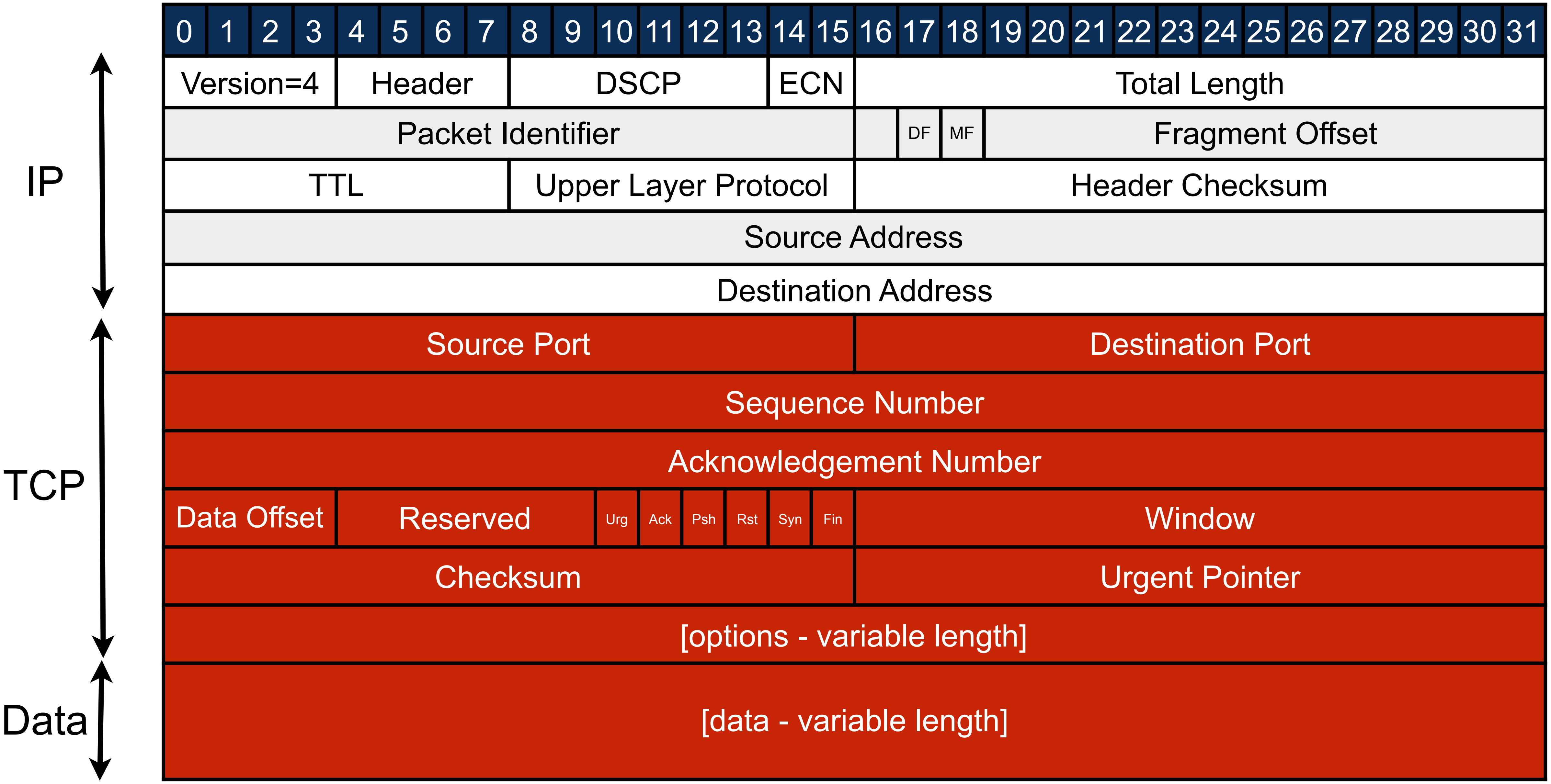
TCP: Transmission Control Protocol

- Reliable byte stream protocol running over IP
  - Adds reliability
    - Packets contain sequence number to detect loss; any lost packets are retransmitted; data is delivered to higher-layers in order, without gaps
  - Adds congestion control – details in lecture 7
  - Adds 16 bit port number as a service identifier
  - Doesn't provide framing
    - Delivers an ordered byte stream, the application must impose structure

TCP Service Model



TCP Packet Format



<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Version=4</td><td colspan="4">Header</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">Packet Identifier</td><td></td><td>DF</td><td>MF</td><td colspan="13">Fragment Offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Upper Layer Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
<tr><td colspan="16">Source Port</td><td colspan="16">Destination Port</td></tr>
<tr><td colspan="32">Sequence Number</td></tr>
<tr><td colspan="32">Acknowledgement Number</td></tr>
<tr><td colspan="4">Data Offset</td><td colspan="6">Reserved</td><td>Urg</td><td>Ack</td><td>Psh</td><td>Rst</td><td>Syn</td><td>Fin</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent Pointer</td></tr>
<tr><td colspan="32">[options - variable length]</td></tr>
<tr><td colspan="32">[data - variable length]</td></tr>
</table>

TCP Applications

- Useful for applications that require reliable data delivery, and can tolerate some timing variation
  - File transfer and web downloads
  - Email
  - Instant messaging
- Default choice for most applications

Other Transport Protocols

- The IP network layer also supports two new transport protocols:
  - DCCP
  - SCTP
- Not widely used at this time, but potentially useful in future

DCCP

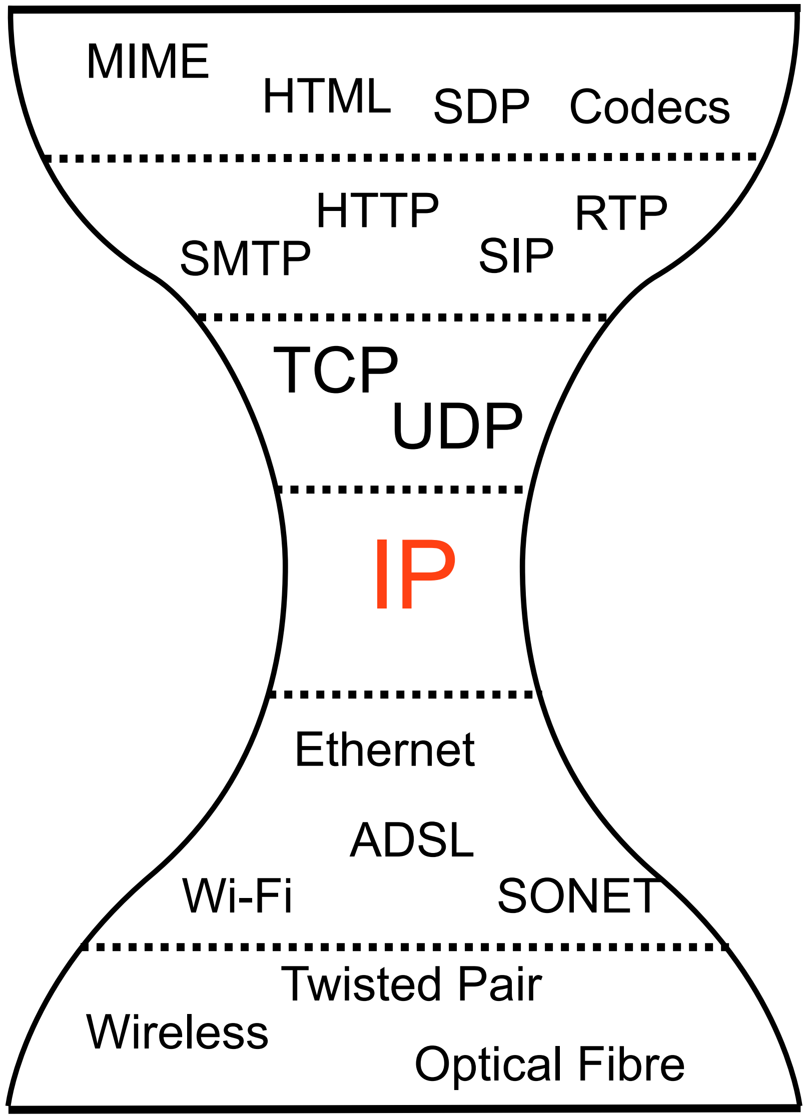
- Datagram Congestion Control Protocol
  - Unreliable, connection oriented, congestion controlled datagram service
    - “TCP without reliability” or “UDP with connections and congestion control”
    - Potentially easier for NAT boxes and firewalls than UDP
    - Congestion control algorithm (“CCID”) negotiated at connection setup – range of algorithms supported
  - Adds 32 bit service code in addition to port number
- Use case: streaming multimedia and IPTV

SCTP

- Stream Control Transmission Protocol
  - Reliable datagram service, ordered per stream
  - Multiple streams within a single association
  - Multiple connection management
    - Fail-over from one IP address to another, for reliable multi-homing
  - TCP-like congestion control
- Use case: telephony signalling; “a better TCP”

Deployment Considerations

- IP is agnostic of the transport layer protocol
- But, firewalls perform "deep packet inspection" and look beyond the IP header to make policy decisions
  - The only secure policy is to disallow anything not understood
  - Implication: very difficult to deploy new transport protocols (e.g., DCCP and SCTP) in the Internet
  - Implication: limits future evolution of the network



Deployment Considerations: Tunnelling New Transports

- If protocols cannot be deployed natively, they can be tunnelled
  - WebRTC data channel → SCTP over DTLS over UDP
    - Peer-to-peer data for web applications

      https://webrtc.org/
  - QUIC → multiplexed stream transport protocol running over UDP
    - Google's proposal for a new transport for HTTP/2 running over UDP, currently being standardised by IETF

      A. Langley, *et al.*, "The QUIC transport protocol: Design and Internet-scale deployment", In Proceedings of the SIGCOMM Conference, Los Angeles, CA, USA, August 2017. ACM. https://dl.acm.org/authorize.cfm?key=N33907

      https://datatracker.ietf.org/wg/quic/about/
- UDP passes through NATs and firewalls, that native transport protocols do not – so tunnel new transport inside UDP packets

Summary

Protocol	Addressing	Reliable?	Framed?	Congestion Controlled?
UDP	16 bit port number	Unreliable packet delivery	Yes – uses explicit datagrams	No – handled by application layer
TCP	16 bit port number	Reliable ordered byte stream	No – handled by application layer	Yes – suitable for elastic applications
DCCP	16 bit port number plus service code	Unreliable packet stream	Yes – uses explicit datagrams	Yes – wide range of algorithms possible
SCTP	16 bit port number	Reliable ordered byte stream	Yes – explicit *chunk* boundaries	Yes – suitable for elastic applications

- Wide range of transport protocols in the Internet, each giving a different end-to-end service model
- TCP and UDP globally deployed; others used in limited environments