Module 4: UDP, TCP, IP

our goals:

- understand key principles behind Internet's transport and network layers:
  - reliable data transfer
  - congestion control
  - routing

- learn about Internet's transport and network layer protocols:
  - UDP: connectionless transport
  - TCP: reliability and congestion control
  - IP: the Internet Protocol
  - Routing protocols: determine end-end path through network routers

Network layer

- transport segment from sending to receiving host
- on sending side, encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in every host, router
- router examines header fields in all IP datagrams passing through it
Two key network-layer functions

- **forwarding**: move packets from router’s input to appropriate router output

- **routing**: determine route taken by packets from source to dest.
  - **routing algorithms**

**analogy:**

- **routing**: process of planning trip from source to dest
- **forwarding**: process of getting through single interchange

Interplay between routing and forwarding

- Routing algorithm determines end-end-path through network
- Forwarding table determines local forwarding at this router

Destination IP address in arriving packet’s header:

128.119.*.*
Interplay between routing and forwarding

4 billion IP addresses, so rather than list individual destination address list range of addresses (aggregate table entries)

What’s inside a router?

Connector slots for incoming and outgoing links

... and probably some circuit chips
**Router architecture overview**

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- forwarding datagrams from incoming to outgoing link

```
forwarding tables computed, pushed to input ports
```

```
routing processor
```

```
high-seed switching fabric
```

```
forwarding data plane (hardware)
```

```
routing, management control plane (software)
```

```
router input ports
```

```
router output ports
```

**Input port functions**

```
line termination
```

```
physical layer: bit-level reception
```

```
data link layer: e.g., Ethernet
```

```
queueing: if datagrams arrive faster than forwarding rate into switch fabric
```

```
link layer protocol (receive)
```

```
lookup, forwarding queueing
```

```
switch fabric
```

```
```
Network Layer 4-7
```

```
```
Network Layer 4-8
```
Output ports

- **buffering** required when datagrams arrive from fabric faster than the transmission rate
- **scheduling discipline** chooses among queued datagrams for transmission

Datagram (packets) can be lost due to congestion, lack of buffers

Priority scheduling – who gets best performance, network neutrality

Network Layer 4-9

Network Layer 4-10
Interplay between routing and forwarding

4 billion IP addresses, so rather than list individual destination address list range of addresses (aggregate table entries)

The Internet network layer

host, router network layer functions:

- **transport layer**: TCP, UDP
- **routing protocols**
  - path selection
  - RIP, OSPF, BGP
- **IP protocol**
  - addressing conventions
  - datagram format
  - packet handling conventions
- **ICMP protocol**
  - error reporting
  - router "signaling"
**IPv4 addressing: introduction**

- **IPv4 address**: 32-bit identifier for host, router *incoming link*
  - routers have multiple incoming links
  - host often has one or two incoming links (e.g., wired Ethernet, wireless 802.11)
- IP addresses associated with each interface
- your IP address changes as you change network to which you’re attached.

**IP datagram format**

- IP protocol version number
- header length (bytes)
- “type” of data
- max number remaining hops (decremented at each router)
- upper layer protocol to deliver payload to
- how much overhead?
  - 20 bytes of TCP
  - 20 bytes of IP
  - = 40 bytes + app layer overhead
- header length (bytes)
- length
- type of service
- checksum
- identifier
- time to live
- fragment offset
- header
- options (if any)
- data
  - (variable length, typically a TCP or UDP segment)
- source IP address
- destination IP address
- total datagram length (bytes)
- e.g. timestamp, record route taken, specify list of routers to visit.
IPv4 addressing: more

- **IP address**: each device has its own globally unique address when connected to a network
  - anyone anywhere in world can address/send datagrams
  - IP address will change as host changes attachment to Internet (e.g., home, dorm, café)
- **What’s your IP address?**
  - MAC: Apple->system-preferences->network -> advanced -> TCP/IP
  - Windows: control panel -> Network and Internet -> Network Sharing -> connection -> view status -> details
  - Iphone: settings -> WiFi -> “>” for current network
- What’smyipaddress.com

IP addresses: how to get one?

**Q**: How does a host get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- **DHCP**: Dynamic Host Configuration Protocol: dynamically get address from as server
  - “plug-and-play”
DHCP: Dynamic Host Configuration Protocol

**goal:** allow host to *dynamically* obtain its IP address from network server when it joins network
- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)

**DHCP overview:**
- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP client-server scenario
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

Broadcast: is there a DHCP server out there?

DHCP offer

Broadcast: I'm a DHCP server! Here's an IP address you can use

DHCP request

Broadcast: OK. I'll take that IP address!

DHCP ACK

Broadcast: OK. You've got that IP address!

DHCP: more than IP addresses

DHCP returns more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)
DHCP: example

- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

DHCP: example

- DCP server formulates DHCP ACK containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router
IPv6: motivation

- **initial motivation (early 1990’s):** 32-bit address space soon to be completely allocated
  - prediction: no more IPv4 addresses available by ~2012
- **additional motivation:**
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

---

priority: identify priority among datagrams in flow
flow Label: identify datagrams in same “flow.”
address: 128 bits!
IPv6: adoption

- US National Institutes of Standards estimate [2013]:
  - ~3% of industry IP routers
  - ~11% of US gov’t routers

- Long (long!) time for deployment, use
  - 20 years and counting!
  - think of application-level changes in last 20 years:
    WWW, Facebook, …
  - Why?

Module 4: UDP, TCP, IP

our goals:

- understand key principles behind Internets transport and network layers:
  - reliable data transfer
  - congestion control
  - routing

- learn about Internet’s transport and network layer protocols:
  - UDP: connectionless transport
  - TCP: reliability and congestion control
  - IP: the Internet Protocol
  - Routing protocols: determine end-end path through network routers
Interplay between routing, forwarding

Graph abstraction

graph: $G = (N,E)$

set of routers = \{ u, v, w, x, y, z \}

set of links =\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}
Graph abstraction: costs

c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or related to bandwidth or congestion level

cost of path from source to dest equals sums of links costs along path:
cost (u,v,w) = cost(u,v) + cost(v,w) = 2+3 = 5
cost (u,x,w) = cost(u,x) + cost(x,w) = 1+3 = 4
cost (u,x,y,w) = cost(u,x) + cost(x,y) + cost(y,w) = 1+1+1 = 3
cost(u,x,y,z,w) = cost(u,x) + cost(x,y) + cost(y,z) + cost(z,w) = 1+1+2+5 = 9
cost (u,x,v,w) = cost(u,x) + cost(x,v) + cost(v,w) = 1+2+3 = 6

cost (u,w) = cost(u,w) = 6

cost of path (x_1, x_2, x_3, ..., x_j) = c(x_1,x_2) + c(x_2,x_3) + ... + c(x_{j-1},x_j)

six possible paths from u to w, with different costs!
which one to choose?
choose path this least cost
• toll road analogy?
**Graph abstraction: costs**

- six possible paths from $u$ to $w$, with different costs!
- which one to choose?
- choose path with least cost
- toll road analogy

**cost of path from source to dest equals sums of links costs along path:**

- $\text{cost}(u,v) = \text{cost}(u) + \text{cost}(v) = 2 + 3 = 5$
- $\text{cost}(u,x) = \text{cost}(u) + \text{cost}(x) = 1 + 3 = 4$
- $\text{cost}(u,x,w) = \text{cost}(u,x) + \text{cost}(x,w) = 1 + 1 + 1 = 3$
- $\text{cost}(u,x,y,w) = \text{cost}(u,x) + \text{cost}(x,y) + \text{cost}(y,w) = 1 + 1 + 2 + 5 = 9$
- $\text{cost}(u,v) = \text{cost}(u) + \text{cost}(v) = 1 + 2 + 3 = 6$
- $\text{cost}(u,w) = \text{cost}(u) + \text{cost}(w) = 6$

**cost of path** $(x_1, x_2, x_3, ..., x_p) = c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1}, x_p)$

---

**Graph abstraction: costs**

- Example: what is least cost path from $u$ to $z$?

**Algorithm:** sequence of steps to be taken to solve a problem
Graph abstraction: costs

\[ c(x,x') = \text{cost of link } (x,x') \]

\[ \text{e.g., } c(w,z) = 5 \]

Cost of path \((x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)\]

**Question:** what is the least-cost path between \(u\) and \(z\)?

**Routing algorithm:** algorithm that finds that least cost path

Routing algorithm classification

**Q: global or decentralized information?**

**Global:**
- all routers have complete topology, link cost info
- “link state” algorithms

**Decentralized:**
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

**Q: static or dynamic?**

**Static:**
- routes change slowly over time

**Dynamic:**
- routes change more quickly
  - periodic update
  - in response to link cost changes
Routing algorithm classification

Q: global or decentralized information?

**global:**
- all routers have complete topology, link cost info
- “link state” algorithms

**decentralized:**
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Q: static or dynamic?

**static:**
- routes change slowly over time

**dynamic:**
- routes change more quickly
  - periodic update
  - in response to link cost changes

The routing problem

<table>
<thead>
<tr>
<th>u</th>
<th>y</th>
<th>x</th>
<th>w</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

\[ c(x, x') = \text{cost of link } (x, x') \]

E.g., \( c(w, z) = 5 \)

cost could always be 1, or related to bandwidth or congestion level

Cost of path \( (x_1, x_2, x_3, ..., x_p) = c(x_1, x_2) + c(x_2, x_3) + ... + c(x_{p-1}, x_p) \)

**Algorithm:** sequence of steps to be taken to solve a problem

**key question:** what is the least-cost path between u and z ?

**routing algorithm:** algorithm that finds that least cost path
A Link-State Routing Algorithm

**Dijkstra’s algorithm**

- centralized: network topology, link costs known to all nodes
  - “link state broadcast” : all nodes flood their links state info to all other nodes
  - algorithm has **global** view of network
- computes least cost paths from one node (‘source’) to all other nodes
  - gives **forwarding table** for that node

---

**Diskstra’s algorithm**

**Goal:** calculate shortest path routes from some node (u) to all other nodes

**Iterative algorithm:** repeats a group of steps at each iteration

**How it works:** at ith iteration, calculate shortest path to ith closest node, making use of calculations done in earlier iterations
Dijkstra’s algorithm

*How it works:* at *ith* iteration, calculate shortest path to *ith* “closest” (by path cost) node, making use of calculations done in earlier iterations

Let’s animate this with balls and string

<table>
<thead>
<tr>
<th>Iteration</th>
<th>&quot;Closest node, cost&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>w, 3</td>
</tr>
<tr>
<td>2</td>
<td>x, 5</td>
</tr>
<tr>
<td>3</td>
<td>v, 6</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>y, 10</td>
</tr>
<tr>
<td>6</td>
<td>z, 12</td>
</tr>
</tbody>
</table>

A Link-State Routing Algorithm

*Notation:*

- **c(a,b):** link cost from node *a* to *b*; = ∞ if not direct neighbors
- **D(c):** current value of cost of path from source to dest. *v*
- **p(c):** predecessor node along path from source to *v*
- **N':** set of nodes whose least cost path definitively known
Dijkstra’s Algorithm

1. **Initialization**: suppose the source node is \( a \)
2. \( N' = \{a\} \)
3. for all nodes \( d \)
4. if \( d \) adjacent to \( a \)
5. then \( D(d) = c(a,d) \)
6. else \( D(d) = \infty \)

8. **Loop**
9. find a node \( e \), not in \( N' \), such that \( D(e) \) is a minimum
10. add \( e \) to \( N' \)
11. update \( D(f) \) for all nodes \( f \), neighboring \( e \), and not in \( N' \):
12. \( D(f) = \min(D(f), D(e) + c(e,f)) \)
13. /* new cost to each node \( f \) is either old cost to \( f \) or known
14. shortest path cost to \( e \) plus cost from \( e \) to \( f */
15. **until all nodes are in \( N' \)**

Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>( N' )</th>
<th>( D(v) )</th>
<th>( D(w) )</th>
<th>( D(x) )</th>
<th>( D(y) )</th>
<th>( D(z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( u )</td>
<td>7,( u )</td>
<td>( 3,u )</td>
<td>5,( u )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>1</td>
<td>( uw )</td>
<td>6,( w )</td>
<td>( 5,u )</td>
<td>11,( w )</td>
<td>( \infty )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( uwx )</td>
<td>6,( w )</td>
<td>11,( w )</td>
<td>14,( x )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( uwxv )</td>
<td>( 10,v )</td>
<td>14,( x )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( uwxvy )</td>
<td>( 12,v )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( uwxvzy )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)
## Dijkstra’s algorithm: another example

### Step 0

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v),p(v)</th>
<th>D(w),p(w)</th>
<th>D(x),p(x)</th>
<th>D(y),p(y)</th>
<th>D(z),p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>2,u</td>
<td>5,u</td>
<td>1,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>ux</td>
<td>2,u</td>
<td>4,x</td>
<td>2,x</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Network Layer** 4-43
Dijkstra’s algorithm: example (2)

resulting shortest-path tree from u:

```
  u ---> x ---> y
  |    |    |
  |    |    |
  v    w  z
```

resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>

Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
Distance vector algorithm

A distributed algorithm: minimum cost of path from me (node $a$) to a destination node $c$ is minimum of paths going through each of my neighbors, $b$

Let

$$d_a(c) := \text{cost of least-cost path from } a \text{ to } c$$

then

$$d_a(c) = \min \{ \text{cost of path via } b \text{ to } c \}$$

over all neighbors, $b$

- cost from neighbor $b$ to destination $c$
- cost to neighbor $b$

$min$ taken over all neighbors $b$ of $a$
Bellman-Ford example

Suppose we know:
\[ d_v(z) = 5, \quad d_x(z) = 3, \quad d_w(z) = 3 \]

\[ d_u(z) = \min \{ \sum_{\text{neighbors } x, v, w} c(u, x) + d_x(z), \]
\[ c(u, v) + d_v(z), \]
\[ c(u, w) + d_w(z) \} \]
\[ = \min \{ 2 + 5, \]
\[ 1 + 3, \]
\[ 5 + 3 \} = 4 \]

neighboring node achieving minimum is next hop in shortest path, used in forwarding table

Distance vector algorithm

\[ D_a(c) = a's \text{ current estimate of it least cost to } c \]

\[ d_a(c) = \min \{ \sum_{\text{neighbors } b} c(a, b) + d_b(c) \} \]

\[ \text{Infinity if } v's \text{ estimate of its cost to } y \text{ is known} \]

Distance vector:
\[ D_a^{(k)} = a's \text{ current estimate of it least cost to all known destinations} \]
**Distance vector algorithm**

*key idea:*
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when node \( a \) receives new DV estimate from neighbor, it updates its own DV using:
  \[
  D_a(c) \leftarrow \min\{c(a,b) + D_b(c)\} \quad \text{for all destinations, } c
  \]
- neighbors, \( b \)
- Keep exchanging distance vectors with neighbors until distance vectors values do not change

---

**Distance vector algorithm**

*iterative, asynchronous:*
- each local iteration caused by:
  - local link cost change
  - DV update message from neighbor

*distributed:*
- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

*each node:*
- wait for (change in local link cost or msg from neighbor)
- recompute estimates of distance to all destinations, and neighbor on each shortest path
- if DV to any dest has changed, notify neighbors by sending new DV
Network Layer

Your DV: estimate of distance to known neighbors

In-class DV execution

*Initialize* your DV table with distance to immediate neighbors, send to all neighbors

*wait* for (change in local link cost or msg from neighbor)

*recompute DV*: recompute estimates of distance to all destinations, and neighbor on each shortest path

if DV to any dest has changed, *notify* neighbors by sending new DV
\( D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \)

\( D_x(z) = \min\{c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \)

\( D_y(z) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \)

\( D_y(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \)

\( D_z(z) = \min\{c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \)
RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982, RFC 2453
- distance vector routing protocol
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - if no advertisement heard after 180 sec --> neighbor/link declared dead. Link cost set to infinity, re-do distance vector

```

<table>
<thead>
<tr>
<th>subnet</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
</tr>
</tbody>
</table>
```

from router A to destination

OSPF (Open Shortest Path First)

- “open”: publicly available routing protocol, RFC 2328
- uses link state algorithm
  - link state packet flooding in ISPs network
  - full topology map known at each node
  - route computation using Dijkstra’s algorithm
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol, RFC 4271
  - “glue that holds the Internet together”
- BGP provides each ISP (aka “autonomous system”) a means to:
  - determine “good” routes to other networks based on reachability information and policy.
  - advertise its existence to rest of Internet: “I am here”

BGP basics

- BGP session: two BGP routers (“peers”) exchange BGP messages:
  - advertising paths to different destination networks
  - gateway routers at “edge” of network, talks to routers in other networks
- AS1 can decide which paths it wants to use, and whether or not it wants to advertise a destination to its neighbor
  - policy-based routing
- when AS3 advertises a destination to AS1:
  - AS3 promises it will forward datagrams towards that destination
BGP: routing policy

- A, B, C are competing ISPs
- A advertises its customer’s path Aw to B
- B advertises path BAw to its customer, x since x might want to route to w
- Should B advertise path BAw to C?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to w via A
  - B wants to route only to/from its customers!