Network Layer
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
Network Layer Introduction

- Concerned with getting packets from source to destination.
- The network layer must know the topology of the subnet and choose appropriate paths through it.
- When source and destination hosts are in different networks, the network layer (IP) must deal with these differences.

*Key issue: What service does the network layer provide to the transport layer? connection-oriented or connectionless
Network Layer Design Goals

1. The services provided by the network layer should be independent of the subnet topology.

2. The Transport Layer should be shielded from the number, type and topology of the subnets present.

3. The network addresses available to the Transport Layer should use a uniform numbering plan (even across LANs and WANs).
Network Layer

- Physical layer
- Data link layer
- Network layer
- Transport layer

Messages

Segments

End system α

Network service

End system β

Leon-Garcia & Widjaja: Communication Networks
Network Layer

Machine A
- Application
- Transport
- Internet
- Network Interface

Router/Gateway
- Internet
- Network Interface

Network 1

Machine B
- Application
- Transport
- Internet
- Network Interface

Network 2

Leon-Garcia & Widjaja: Communication Networks
Metropolitan Area Network (MAN)

Gateway

To the Internet or wide area network

Backbone

Organization Servers

Departmental Server

Leon-Garcia & Widjaja: Communication Networks
Wide Area Network (WAN)

Interdomain level

Border routers

Autonomous system or domain

Border routers

LAN level

Intradomain level

Internet Service Provider (ISP)

Leon-Garcia & Widjaja: Communication Networks
Modern Internet Backbone

National service provider A

National service provider B

National service provider C

Network Access Point

National Internet Service Providers

Leon-Garcia & Widjaja: Communication Networks
Network Access Point

Diagram showing a network access point (NAP) connected to route servers $R_A$, $R_B$, and $R_C$. The diagram illustrates the network layer with a LAN connected to the NAP.
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing
### Datagram Routing Table

<table>
<thead>
<tr>
<th>Destination address</th>
<th>Output port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0785</td>
<td>7</td>
</tr>
<tr>
<td>1345</td>
<td>12</td>
</tr>
<tr>
<td>1566</td>
<td>6</td>
</tr>
<tr>
<td>2458</td>
<td>12</td>
</tr>
</tbody>
</table>
Virtual Circuit Packet Switching

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

Packet

 Packet
### Virtual Circuit Routing Table

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Output port</th>
<th>Next identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>44</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>23</td>
</tr>
<tr>
<td>27</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>58</td>
<td>7</td>
<td>34</td>
</tr>
</tbody>
</table>

Entry for packets with identifier 15

Packet leaves with new identifier 23
- transport segment from sending to receiving host.
- on sending side, encapsulates segments into datagram packets.
- 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.
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
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 destination.

**analogy:**

- **forwarding:** process of getting through single interchange

- **routing:** process of planning trip from source to destination

K & R
Interplay between Routing and Forwarding

Routing algorithm

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

value in arriving packet's header
Router Node

node 15

Incoming Link

Router Link Buffer

Server

Outgoing Link

packet

Computer Networks  Network Layer
Routing in an internet

Figure 3.14  A Simple internetwork with Three Routers
- **IP** runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork.

**Figure 3.15** Protocol Layers used for a Simple internet
### Table 3.6 Complete Forwarding Table for Router R2 in Figure 3.14

<table>
<thead>
<tr>
<th>Network Number</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R1</td>
</tr>
<tr>
<td>2</td>
<td>Interface 1</td>
</tr>
<tr>
<td>3</td>
<td>Interface 0</td>
</tr>
<tr>
<td>4</td>
<td>R3</td>
</tr>
</tbody>
</table>

**Note** - As R2 is on Network 2 and Network 3, this table shows packets headed for H1, H2 and H3 are not forwarded by R2 to another router.
Host, router network layer functions:

**Routing protocols**
- path selection
- RIP, OSPF, BGP

**IP protocol**
- addressing conventions
- datagram format
- packet handling conventions

**ICMP protocol**
- error reporting
- router "signaling"

**Transport Layer**: TCP, UDP

**Data Link Layer**

**Physical Layer**

*Computer Networks Network Layer*
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
IP Datagram Format

- **IP protocol version number** (bytes)
- **header length**
- **TOS:: data type**
- **TTL:: max hops remaining (each router decrements)**
- **upper layer protocol to deliver payload to**

**How much overhead with TCP?**
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead

**32 bit source IP address**

**32 bit destination IP address**

**Options (if any)**
- E.g. timestamp, record route taken, specify list of routers to visit.

**Three fields for fragmentation/reassembly**
- header
- checksum
- fragment offset

**Header Length (bytes)**
- ver
- head
- len
- type of service
- 16-bit identifier
- flgs
- time to live
- upper layer
- header
- checksum
- length
- total datagram length
- fragment length

**Data (variable length, typically a TCP or UDP segment)**

**Computer Networks Network Layer**
IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams.
  - "reassembled" only at final destination.
  - IP header bits used to identify, order related fragments.

fragmentation:
in: one large datagram
out: 3 smaller datagrams

reassemble
Figure 3.18 Header fields used in IP fragmentation: (a) unfragmented packet; (b) fragmented packets.
Figure 3.17 IP datagrams traversing the sequence of physical networks graphed in Figure 3.14.
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing
Global Internet

Figure 4.1 The tree structure of the Internet in 1990
Global Internet

- Each provider network is regional and a single autonomous system (AS)

- Major issues are:
  - Scalability of routing
  - Address utilization
    - Now out of IPv4 addresses

- Hierarchy is used to improve scalability.
  - Namely, utilize subnets with masks.
IPv4 Network Classes

Figure 3.19 IP addresses: (a) class A; (b) class B; (c) class C.
Network Number Problems

- Assigning one network number per physical network uses up IP address too fast!
- Adding more network numbers also increases forwarding table size.

**Subnet solution ::**
- The idea is to take a single IP network number and allocate the IP addresses with that network number to several physical networks which are referred to as **subnets**.
- The subnets need to be physically close to each other for routing purposes.
Subnetting

- The mechanism by which a single network number can be shared among multiple networks involves configuring all the nodes on each subnet with a **subnet mask**.
- The **subnet mask** enables introduction of a single subnet number which provides for another level of hierarchy into the IP address.
- All hosts on a given subnet are configured with the same mask, i.e., there is one subnet mask per subnet.
IP Addressing: Introduction

- **IP address**: 32-bit identifier for host, router *interface*.
- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces.
  - host typically has one interface.
  - IP addresses are associated with each interface.

223.1.1.1 = 11011111 00000001 00000001 00000001

223.1.1.1

223.1.1.2

223.1.1.3

223.1.1.4

223.1.2.9

223.1.2.2

223.1.2.1

223.1.3.2

223.1.3.27
**IP address:**
- subnet part (high order bits)
- host part (low order bits)

**What is a subnet?**
- device interfaces with same subnet part of IP address.
- can physically reach each other without intervening router.

network consisting of **three** subnets
Subnet Masks

**Figure 3.20 Subnet Addressing**

<table>
<thead>
<tr>
<th>Network number</th>
<th>Host number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B address</td>
<td></td>
</tr>
<tr>
<td>11111111111111111111111111111111</td>
<td>00000000</td>
</tr>
<tr>
<td>Subnet mask (255.255.255.0)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network number</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnetted address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Subnet Concepts

- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is called a subnet.
- Sending host does bitwise **AND** between subnet mask and destination address to determine whether packet needs to be routed or not.
Subnets

How many?

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4

223.1.2.1
223.1.2.2
223.1.2.6

223.1.3.1
223.1.3.2
223.1.3.27

223.1.7.0
223.1.7.1
223.1.8.0
223.1.8.1

223.1.9.1
223.1.9.2
CIDR: Classless InterDomain Routing

- Allows a subnet portion of address of arbitrary length.

- address format: \texttt{a.b.c.d/x}, where \texttt{x} is number of bits in subnet portion of address.

\begin{align*}
\text{subnet part} & \quad \text{host part} \\
11001000 & \quad 00010111 & \quad 00010000 & \quad 00000000 \\
\end{align*}

200.23.16.0/23
Classless Routing (CIDR)

- CIDR helps aggregate routes by breaking up rigid boundaries between classes.
- Handing out Class C addresses in contiguous blocks by address, makes it possible for addresses to share a common prefix.
  
  ➔ allocate Class C networks as a power of 2.

- We need a protocol that understands these rules, e.g., BGP!!

- Network numbers are represented by (length,value) where length is the length of the prefix {similar to a mask}. 
Classless Routing (CIDR)

Border gateway (advertises path to 11000000000001)

Regional network

Corporation X (11000000000001000001)

Corporation Y (11000000000001000000)

Figure 4.27 Route Aggregation with CIDR
CIDR Route Aggregation

Figure 3.22 Route Aggregation with CIDR

Eight ISP customers share a 21-bit common prefix.
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
Routing Algorithm Classification
Routing algorithm: that part of the Network Layer responsible for deciding on which output line to transmit an incoming packet.

- Remember: For virtual circuit subnets the routing decision is made ONLY at set up.

Algorithm properties: correctness, simplicity, robustness, stability, fairness, optimality, and scalability.
## Routing Classification

<table>
<thead>
<tr>
<th>Adaptive Routing</th>
<th>Non-Adaptive Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>based on current measurements of traffic and/or topology.</td>
<td>routing computed in advance and off-line</td>
</tr>
<tr>
<td>1. centralized</td>
<td>1. flooding</td>
</tr>
<tr>
<td>2. isolated</td>
<td>2. static routing using shortest path algorithms</td>
</tr>
<tr>
<td>3. distributed</td>
<td></td>
</tr>
</tbody>
</table>

Adaptive Routing

- based on current measurements of traffic and/or topology.
- 1. centralized
- 2. isolated
- 3. distributed

Non-Adaptive Routing

- routing computed in advance and off-line
- 1. flooding
- 2. static routing using shortest path algorithms
Flooding

- **Pure flooding**: every incoming packet to a node is sent out on every outgoing line.
  - Obvious adjustment - do not send out on arriving link (assuming full-duplex links).
  - The routing algorithm can use a hop counter (e.g., TTL) to **dampen the flooding**.
- **Selective flooding**: only send on those lines going “approximately” in the right direction.
Routing is Graph Theory Problem

Figure 3.28 Network represented as a graph.

edges have costs
Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra’s Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

We need a cost metric (edges in graph):

a. Minimize the number of hops along the path.
b. Minimize the mean packet delay.
c. Maximize the network throughput.
Internetwork Routing

Adaptive Routing

Centralized
[IGP] Intradomain routing
[IGP] Interior Gateway Protocols
Distance Vector routing
[RIP]

Distributed
Interdomain routing
[EGP] Exterior Gateway Protocols
Link State routing
[OSPF, IS-IS, PNNI]

Isolated
[EGP] Exterior Gateway Protocols
[BGP, IDRP]
Adaptive Routing Design

Design Issues:

1. How much overhead is incurred due to gathering the routing information and sending routing packets?

2. What is the time frame (i.e., the frequency) for sending routing packets in support of adaptive routing?

3. What is the complexity of the routing strategy?
Adaptive Routing

Basic functions:

1. Measurement of pertinent network data \(\text{e.g. the cost metric}\).

2. Forwarding of information to where the routing computation will be done.

3. Compute the routing tables.

4. Convert the routing table information into a routing decision and then dispatch the data packet.
Centralized Routing

A

B

RCC

W

Z

Computer Networks  Network Layer
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
Distance Vector Routing

{Tanenbaum & Perlman version}
Historically known as the old ARPANET routing algorithm {or known as Bellman-Ford (BF) algorithm}.

BF Basic idea: each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes.

Distances, based on the chosen metric, are computed using information from the neighbors’ distance vectors.

Distance Metric: usually hops or delay
Distance Vector Routing

Information kept by DV router

1. each router has an ID
2. associated with each link connected to a router, there is a link cost (static or dynamic).

Distance Vector Table Initialization

Distance to itself = 0
Distance to ALL other routers = infinity number
1. A router transmits its **distance vector** to each of its **neighbors** in a routing packet.

2. Each router receives and saves the most recently received **distance vector** from each of its neighbors.

3. A router recalculates its **distance vector** when:
   a. It receives a **distance vector** from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a **topology change**).

The DV calculation is based on minimizing the cost to each destination.
Figure 5–9. (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.
Distance Vector Routing

{Kurose & Ross version}
Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]

Then
\[ d_x(y) = \min_v \{ c(x, v) + d_v(y) \} \]

where min is taken over all neighbors v of x.
Clearly, \( d_v(z) = 5 \), \( d_x(z) = 3 \), \( d_w(z) = 3 \)

B-F equation says:

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

The node that achieves minimum is next hop in shortest path ➔ forwarding table. Namely, packets from \( u \) destined for \( z \) are forwarded out link between \( u \) and \( x \).
Distance Vector Algorithm (3)

- \( D_x(y) \) = estimate of least cost from \( x \) to \( y \)
- Node \( x \) knows cost to each neighbor \( v \): \( c(x,v) \)
- Node \( x \) maintains distance vector
  \[ D_x = [D_x(y): y \in N] \]
- Node \( x \) also maintains its neighbors’ distance vectors
  - For each neighbor \( v \), \( x \) maintains
  \[ D_v = [D_v(y): y \in N] \]
Distance Vector Algorithm (4)

**DV Basic idea:**
- From time-to-time, each node sends its own distance vector estimate to neighbors.
- Asynchronous
- When a node $x$ receives a new DV estimate from any neighbor $v$, it saves $v$'s distance vector and it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ converges to the actual least cost $d_x(y)$. 
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
- if DV to any destination has changed, notify neighbors
### Node $x$ Table

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>$y$</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>$z$</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

**Cost to:**

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

### Node $y$ Table

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td></td>
<td></td>
<td>∞</td>
</tr>
<tr>
<td>$y$</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$z$</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

**Cost to:**

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

### Node $z$ Table

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>$y$</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>$z$</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Cost to:**
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \]
\[ = \min\{2+0, 7+1\} = 2 \]

\[ D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} \]
\[ = \min\{2+1, 7+0\} = 3 \]
Link cost changes:

- node detects local link cost change.
- updates routing info, recalculates distance vector.
- if DV changes, it notifies neighbors

At time $t_0$, $y$ detects the link-cost change, updates its DV, and informs its neighbors.

“good news travels fast”

At time $t_1$, $z$ receives the update from $y$ and updates its table. It computes a new least cost to $x$ and sends its neighbors its DV.

At time $t_2$, $y$ receives $z$'s update and updates its distance table. $y$'s least costs do not change and hence $y$ does not send any message to $z$. 
Distance Vector: Link Cost Changes

Link cost changes:
- Good news travels fast
- Bad news travels slow - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see P&D page 248!

Possible solutions:
1. Keep ‘infinity’ small {depends on graph diameter}.
2. Split Horizon: node does not send those routes learned from a neighbor back to that neighbor.
3. Split Horizon with Poison Reverse:
   • If Z routes through Y to get to X, Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z).

Does this solve count to infinity problem?
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
1. Each router is responsible for meeting its neighbors and learning their names.

2. Each router constructs a link state packet (LSP) which consists of a list of names and cost to reach each of its neighbors.

3. The LSP is transmitted to ALL other routers. Each router stores the most recently generated LSP from each other router.

4. Each router uses complete information on the network topology to compute the shortest path route to each destination node.
Figure 3.32 Reliable LSP Flooding
Reliable Flooding

- The process of making sure all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes.

- **LSP** contains:
  - Sending router's node ID
  - List of connected neighbors with the associated link cost to each neighbor
  - Sequence number
  - Time-to-live (TTL) \{an aging mechanism\}
Reliable Flooding

- First two items enable route calculation.
- Last two items make process reliable
  - ACKs and checking for duplicates is needed.
- Periodic Hello packets used to determine the demise of a neighbor.
- The sequence numbers are not expected to wrap around.
  ➔ this field needs to be large (64 bits)!!
A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”.
  - all nodes have same info.
- computes least cost paths from one node (‘source”) to all other nodes
  - gives forwarding table for that node.
- iterative: after k iterations, know least cost path to k destinations.

Notation:
- $c(x,y)$: link cost from node $x$ to $y$; $\infty$ if not direct neighbors.
- $D(v)$: current value of cost of path from source to destination $v$
- $p(v)$: predecessor node along path from source to $v$
- $N'$: set of nodes whose least cost path is definitively known.
Dijsktra's Algorithm \([K&R]\)

1. **Initialization:**
   2. \(N' = \{u\}\)
   3. for all nodes \(v\)
   4. if \(v\) adjacent to \(u\)
   5. then \(D(v) = c(u,v)\)
   6. else \(D(v) = \infty\)

8. **Loop**
   9. find \(w\) not in \(N'\) such that \(D(w)\) is a minimum
   10. add \(w\) to \(N'\)
   11. update \(D(v)\) for all \(v\) adjacent to \(w\) and not in \(N'\): \(D(v) = \min(D(v), D(w) + c(w,v))\)
   12. /* new cost to \(v\) is either old cost to \(v\) or known shortest path cost to \(w\) plus cost from \(w\) to \(v\) */
   13. until all nodes in \(N'\)
## Dijkstra's Algorithm: Example

### Table: Dijkstra's Algorithm Steps

<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>
</tbody>
</table>

### Diagram: Network Connectivity

The diagram illustrates the network connectivity with nodes u, v, w, x, y, and z. The edges and their weights are shown as follows:

- u to v: 2
- u to x: 1
- x to y: 1
- y to z: 2
- w to x: 3
- w to v: 3
- u to w: 5
- y to z: 5

The algorithm progresses as follows:

- **Step 0:** Initialize distances to infinity except for u which is 0.
- **Step 1:** Update distance and path for x.
- **Step 2:** Update distance and path for y.
- **Step 3:** Update distance and path for z.
- **Step 4:** Final distances and paths are determined.

### References

- K&R: *Computer Networks and Internets*. Prentice Hall.
Dijkstra's Algorithm: Example (2)

Resulting shortest-path tree from u:

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>
Algorithm complexity: n nodes
- each iteration: need to check all nodes, w, not in N
- \( n(n+1)/2 \) comparisons: \( O(n^2) \)
- more efficient implementations possible: \( O(n \log n) \)

Oscillations possible:
- e.g., link cost = amount of carried traffic

Dijkstra's Algorithm, Discussion

![Diagram showing the process of Dijkstra's Algorithm with oscillations and recompute routing steps.](image-url)
Network Layer Topics

- 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
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
Hierarchical Routing

- Our routing study thus far - idealization
- all routers identical
- network “flat”
- ... not true in practice

scale: with 200 million destinations:
- can’t store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router
- Direct link to router in another AS
Interconnected AS's

- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal destinations.
  - inter-AS & intra-AS sets entries for external destinations.
Inter-AS Tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

**AS1 must:**
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!

Diagram:
- AS1
- AS2
- AS3
- Router nodes labeled a, b, c, d
- Connections between ASes
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
Intra-AS Routing

- also known as **Interior Gateway Protocols (IGP)**
- most common Intra-AS routing protocols:
  - **RIP**: Routing Information Protocol
  - **OSPF**: Open Shortest Path First
  - **IGRP**: Interior Gateway Routing Protocol (Cisco proprietary)
Network Layer Topics

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What is inside a router?

4.4 IP: Internet Protocol
   - Datagram format
   - IPv4 addressing
   - Subnets
   - CIDR
   - ARP & DHCP
   - NAT

4.5 Routing algorithms
   - Algorithm Classification
   - Link State
   - Distance Vector
   - Hierarchical Routing

4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP

4.7 ICMP
- RIP had widespread use because it was distributed with BSD Unix in “routed”, a router management daemon in 1982.
- **RIP** - most used Distance Vector protocol.
- RFC1058 in June 1988
- Runs over UDP.
- Metric = hop count
- BIG problem is max. hop count =16
  - RIP limited to running on small networks (or AS’s that have a small diameter)!!
Routing Information Protocol (RIP)

- Sends DV packets every 30 seconds (or faster) as Response Messages (also called **advertisements**).
- Each advertisement: list of up to 25 destination subnets within AS.
- Upgraded to RIPv2

From router A to subnets:

<table>
<thead>
<tr>
<th>destination</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>
### Figure 4.17 RIP Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td>Must be zero</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>0, 8, 16, 31</td>
</tr>
<tr>
<td><strong>Family of net 1</strong></td>
<td>Address of net 1</td>
</tr>
<tr>
<td><strong>Address of net 1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Distance to net 1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Family of net 2</strong></td>
<td>Address of net 2</td>
</tr>
<tr>
<td><strong>Address of net 2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Distance to net 2</strong></td>
<td></td>
</tr>
</tbody>
</table>

*(network_address, distance)* pairs
RIPv2 Packets

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>Must be zero</td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>Family of net 1</td>
<td>Route Tags</td>
</tr>
<tr>
<td>Address prefix of net 1</td>
<td></td>
</tr>
<tr>
<td>Mask of net 1</td>
<td></td>
</tr>
<tr>
<td>Distance to net 1</td>
<td></td>
</tr>
<tr>
<td>Family of net 2</td>
<td>Route Tags</td>
</tr>
<tr>
<td>Address prefix of net 2</td>
<td></td>
</tr>
<tr>
<td>Mask of net 2</td>
<td></td>
</tr>
<tr>
<td>Distance to net 2</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.31 RIPv2 Packet Format**

subnet masks
Network Layer Topics

- 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
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
OSPF (Open Shortest Path First)

- “open” :: publicly available (due to IETF)
- uses Link State algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation uses Dijkstra’s algorithm.
- OSPF advertisement carries one entry per neighbor router.
- advertisements disseminated to entire AS (via flooding*).
  - carried in OSPF messages directly over IP (rather than TCP or UDP).

* However hierarchy (partitioning domains into areas) reduces flooding impact.
Figure 4.2  A domain divided into areas
Hierarchical OSPF

- **Two-level Hierarchy**: local area, backbone.
  - Some Link-State Advertisement (LSA) types are only sent into one area.
  - Each node has detailed area topology; only knows direction (shortest path) to nets in other areas.
- **area border routers**: “summarize” distances to nets in own area, advertise to other Area Border routers.
- **backbone routers**: run OSPF routing limited to backbone.
- **boundary routers**: connect to other AS’s.
OSPF LSA Types

1. Router link advertisement [Hello message]
2. Network link advertisement
   - identifies connected networks.
3. Network summary link advertisement
4. AS border router’s summary link advertisement
5. AS external link advertisement
Figure 3.34 OSPF Header Format
Type 1 OSPF LSA

<table>
<thead>
<tr>
<th>LS Age</th>
<th>Options</th>
<th>Type = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Link-state ID
- Advertising router
- LS sequence number
- LS checksum
- Length
- 0 Flags 0 Number of links
- Link ID
- Link data
- Link type Num_TOS Metric
- Optional TOS information
- More links

**Figure 3.3 OSPF Link-State Advertisement (LSA)**
OSPF “Advanced” Features (not in RIP)

- **security**: all OSPF messages authenticated (to prevent malicious intrusion).
- **multiple same-cost paths** means distributed load balancing of traffic over routes (only one path in RIP).
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort; high for real time).
- **integrated uni- and multicast support**:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF.
- **hierarchical** OSPF is used in large domains.
Figure 5-65. The relation between AS's, backbones, and areas in OSPF
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
BGP (Border Gateway Protocol): de facto standard interdomain routing protocol.

- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine “good” loop-free routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: “I am here!”
BGP Assumes an Arbitrary Interconnection of AS’s

Figure 4.4  A simple multi-provider Internet
• **Scalability**: An Internet backbone router must be able to forward any packet destined anywhere in the Internet.
  - Having a routing table that will provide a match for any valid IP address.

• **Autonomous nature of the domains**
  - It is impossible to calculate meaningful path costs for a path that crosses multiple Ass.
  - A cost of 1000 across one provider might imply a great path but it might mean an unacceptable bad one from another provider.

• **Issues of trust**
  - Provider A might be unwilling to believe certain advertisements from provider B.
Define **local traffic** as traffic that originates at or terminates on nodes within an AS, and **transit traffic** as traffic that passes through an AS.

Classify AS's into three types:

- **Stub AS**: an AS that has only a single connection to one other AS; such an AS will only carry local traffic (**small corporation in Figure 4.4**).

- **Multihomed AS**: an AS that has connections to more than one other AS, but refuses to carry transit traffic (**large corporation at the top in Figure 4.4**).

- **Transit AS**: an AS that has connections to more than one other AS, and is designed to carry both transit and local traffic (**backbone providers in Figure 4.4**).
BGP does not belong to either of the two main classes of routing protocols (distance vectors and link-state protocols).

BGP advertises complete paths as an enumerated lists of ASs to reach a particular network. Hence, often referred to as a path-vector protocol.

- BGP paths need to be loop-free!!

→ Carried AS numbers need to be unique.
Figure 4.5 Example of a network running BGP
Figure 4.6 Example of loop among autonomous systems
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
A mechanism is needed to translate IP addresses into DL layer addresses that make sense on that network (e.g., 48-bit Ethernet addresses on an Ethernet LAN).

Address pair == (hostID, Network Point of Attachment address)

where

- hostID == IP address
- NPA address == Ethernet address

This translation is needed by a local interior router (e.g., R1 in Figure 3.14) and by hosts on the LAN (e.g., H1–H3 in Figure 3.14)
Routing in an internet

Figure 3.14  A Simple internetwork with Three Routers
One solution :: have the router maintain a table of address pairs for all hosts attached to the LAN. Table is then copied to each host on the LAN.

- This will generate lots of extra traffic on the LAN.

* a better solution:: each host **dynamically** learns the contents of this address pairs table using the LAN.

- This is the function performed by ARP (Address Resolution Protocol).
Address Resolution Protocol (ARP)

- Each node maintains an address pairs table in its cache.
- If IP address of destination host is **NOT** found in the cache, it invokes ARP over the LAN.
- An ARP query for desired IP address is **broadcast** on the LAN. The query includes address pair of sending host.
- Each IP host receives the query and checks to see if it matches its own IP address.
Address Translation Protocol (ARP)

- The target host sends an ARP reply with its DL layer address and adds sending address pair to its local table.
- Query originator updates information in its cached ARP table.
- Note - target hosts will 'refresh' their cache entry if originator is in local cache, but will not make a new table entry otherwise.
- ARP cache entries time out periodically (e.g., every 15 minutes) and non-refreshed entries are discarded.
ARP Packet Format

- **HardwareType**: type of physical network (e.g., Ethernet)
- **ProtocolType**: type of higher layer protocol (e.g., IP)
- **Hlen & Plen**: length of link layer and protocol layer addresses
- **Operation**: request or response

**Figure 3.23 ARP packet format for mapping IP addresses into Ethernet addresses**
**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:** A host dynamically gets address from a server.
  - A “plug-and-play” protocol
Host Configurations

- Ethernet addresses are configured into network by manufacturer and they are unique.
- IP addresses must be unique on a given internetwork but also must reflect the structure of the internetwork.
- Most host Operating Systems provide a mechanism to manually configure the host's IP information.
- Manual configuration disadvantages:
  - A lot of work to configure all the hosts in a large network.
  - Configuration process is error-prune.
- Automated Configuration Process is required.
DHCP

- Relies on the existence of a DHCP Server that provides configuration information.
- Uses UDP and ports 67 and 68.
- Has widespread use in wireless LANs.
- DHCP server maintains a pool of available addresses that are allocated on demand.
DHCP: Dynamic Host Configuration Protocol

**Goal:** Allow a host to *dynamically* obtain its IP address from network server when it joins the network.

- Can renew its lease on address in use.
- Allows reuse of addresses (only hold address while connected an “on”).
- Support for mobile users who want to join network (more shortly).

**DHCP overview:**

1. host broadcasts “DHCP discover” msg [optional]
2. DHCP server responds with “DHCP offer” msg [optional]
3. host requests IP address: “DHCP request” msg
4. DHCP server sends address: “DHCP ack” msg
DHCP Client-Server Scenario

DHCP server arriving DHCP client needs address in this network
DHCP Client-Server Scenario

DHCP server: 223.1.2.5

1. DHCP discover
   - src: 0.0.0.0, 68
   - dest: 255.255.255.255, 67
   - yiaddr: 0.0.0.0
   - transaction ID: 654

2. DHCP offer
   - src: 223.1.2.5, 67
   - dest: 255.255.255.255, 68
   - yiaddr: 223.1.2.4
   - transaction ID: 654
   - Lifetime: 3600 secs

3. DHCP request
   - src: 0.0.0.0, 68
   - dest: 255.255.255.255, 67
   - yiaddr: 223.1.2.4
   - transaction ID: 655
   - Lifetime: 3600 secs

4. DHCP ACK
   - src: 223.1.2.5, 67
   - dest: 255.255.255.255, 68
   - yiaddr: 223.1.2.4
   - transaction ID: 655
   - Lifetime: 3600 secs

K & R
DHCP: More than IP address

DHCP can return more than just allocated IP address on subnet:
- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address).
DHCP Relay Agent

To avoid a DHCP server on every network, DHCP relay agent unicasts the message to DHCP server and waits for the response.

Figure 3.24 DHCP relay agent receives DISCOVER message
- 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 demux’ed to IP demux’ed, UDP demux’ed 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, demux'ing 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.
DHCP: Wireshark Output (home LAN)

**request**

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID:** 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
**Client MAC address:** Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
   Length: 7; Value: 010016D323688A;
   Hardware type: Ethernet
   Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) **Requested IP Address = 192.168.1.101**
Option: (t=12,l=5) **Host Name = "nomad"**
Option: (55) **Parameter Request List**
   Length: 11; Value: 010F03062C2E2F1F21F92B
   **1 = Subnet Mask; 15 = Domain Name**
   **3 = Router; 6 = Domain Name Server**
   **44 = NetBIOS over TCP/IP Name Server**

**reply**

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID:** 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
**Client IP address:** 192.168.1.1 (192.168.1.1)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP ACK**
Option: (t=54,l=4) **Server Identifier = 192.168.1.1**
Option: (t=1,l=4) **Subnet Mask = 255.255.255.0**
Option: (t=3,l=4) **Router = 192.168.1.1**
Option: (6) **Domain Name Server**
   **Length:** 12; **Value:** 445747E2445749F244574092;
   **IP Address:** 68.87.71.226;
   **IP Address:** 68.87.73.242;
   **IP Address:** 68.87.64.146
Option: (t=15,l=20) **Domain Name = "hsd1.ma.comcast.net."**
Network Layer Topics

- 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
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 ICMP
All datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers.

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual).
**Motivation:** local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices.
- can change addresses of devices in local network without notifying outside world.
- can change ISP without changing addresses of devices in local network.
- devices inside local net not explicitly addressable, visible by outside world (a security plus).
Implementation: NAT router must:

- **outgoing datagrams:** replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  
  ... remote clients/servers will respond using (NAT IP address, new port #) as destination address.

- **remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair.

- **incoming datagrams:** replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table.
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345
NAT Traversal Problem

- Client wants to connect to server with address 10.0.0.1
  - Server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
  - Only one externally visible NATted address: 138.76.29.7

- Solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - E.g., (138.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
NAT Traversal Problem

- Solution 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATted host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration
NAT Traversal Problem

- Solution 3: relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - relay bridges packets between to connections
Network Layer Topics

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What is inside a router?
- 4.4 IP: Internet Protocol
  - Datagram Format
  - IPv4 addressing
  - Subnets
  - CIDR
  - ARP & DHCP
  - NAT
- 4.5 Routing algorithms
  - Algorithm Classification
  - Link State
  - Distance Vector
  - Hierarchical Routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- **ICMP**
Internet Control Message Protocol (ICMP)

- IP is always configured with a companion protocol (ICMP) that defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully, e.g.,
  - Destination host unreachable due to link/node failure.
  - Reassembly process failed.
  - TTL had reached 0 (so datagrams don't cycle forever)
  - IP header checksum failed

- ICMP includes several control messages that a router can use to send back to a source host.
  - ICMP-Redirect
    - Sent from router R1 to a source host to inform host there exists a better path through R2 to a particular destination.
  - With a better route information
Other ICMP functions

- **ping** uses ICMP echo messages to determine if a node is reachable and alive. Can be used to estimate RTT.
- **traceroute** is used to determine the set of routes along the path to a destination.

*Note* - some routers send ICMP packets through a separate 'special' queue.
4.1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What is inside a router?
4.4 IP: Internet Protocol
   - Datagram Format
   - IPv4 addressing
   - Subnets
   - CIDR
   - ARP & DHCP
   - NAT
4.5 Routing algorithms
   - Algorithm Classification
   - Link State
   - Distance Vector
   - Hierarchical Routing
4.6 Routing in the Internet
   - RIP
   - OSPF
   - BGP
ICMP