Network Layer

- Network Layer Services
- Routing principles: path selection
- Hierarchal routing
- IP
- Internet Routing Protocols: Reliable Transfer
    a. Intra-Domain
    b. Inter-Domain
- What's inside a router
- IPv6
- Mobility

Network Layer Services

Network Layer Functions

- transport packet from sending to receiving hosts
- network layer protocols in *every* host, router

Three important functions:

- *path determination:* route taken from source to destinations. (*Routing algorithms)*
- *forwarding:* move packets from router's input to appropriate router output.
- *call setup:* some network architectures require router call setup along path before data flows.

Network Service Model

Which *service model* for transporting packets from sender to receiver?

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

The most important abstraction provided by network layer: **Virtual Circuits** or **Datagrams? Big fight!!!**

Virtual Circuit: The Telephone Model

Source-to-Destination path behaves much like a telephone circuit:

- performance-wise
- network actions along source-to-destination path

How it works?

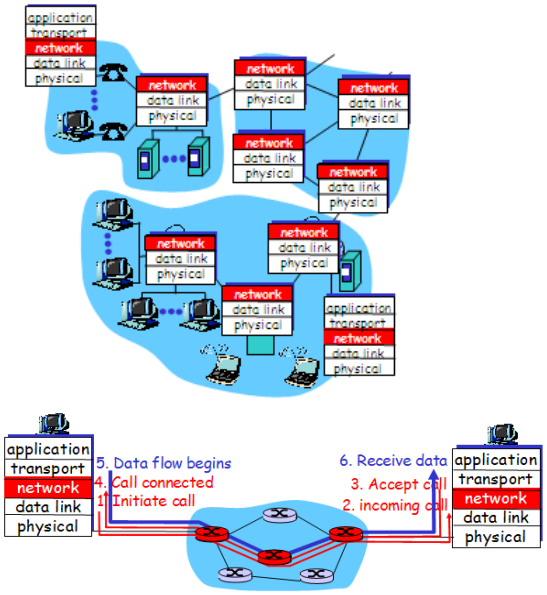
- call setup, teardown for each call *before* data can flow
- each packet carries VC Identifier (not destination host ID)
- *every* router on source-destination path keeps *state* for each passing connection

Note: Transport-layer connections only involved in the two end systems.

- link, router resources (bandwidth, buffers) may be *allocated* to VC to get circuit-like performance.

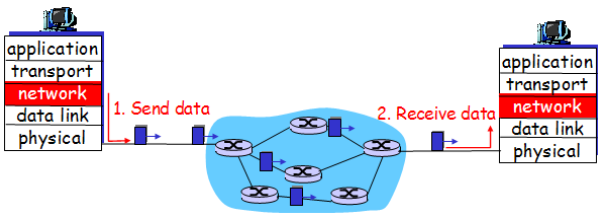
Signaling Protocols

- used to setup, maintain, and teardown Virtual Circuit
- used in ATM, frame-relay, X.25 (*more later*)



Datagram Network: The Internet Model

- no call setup at network layer
- routers: no state about end-to-end connections
    - no network-level concept of "connection"
- packets forwarded using destination host address
    - packets between same source-destination pair may take different paths



Network Layer Service Model Comparison Chart

Network Architecture	Service Model	Guarantees ?				Congestion feedback
		Bandwidth	Loss	Order	Timing	
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	CBR	constant rate	yes	yes	yes	no congestion
ATM	VBR	guaranteed rate	yes	yes	yes	no congestion
ATM	ABR	guaranteed minimum	no	yes	no	yes
ATM	UBR	none	no	yes	no	no

Datagram v/s VC Network: Why?

Internet:

- data exchange among computers
  "elastic" service, no strict timing requirements.
- "smart" end systems (computers)
  can adapt, perform control, error recovery.
  simple inside network, **complexity** at "edge"
- many link types
  different characteristics, thus a uniform service is difficult to achieved.

ATM:

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - **complexity** inside network

Routing Principles: path selection

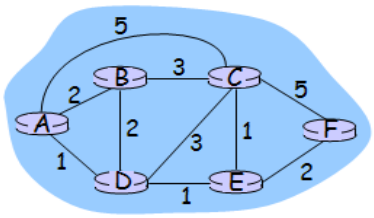
Routing Protocol:

Goal: *Determine "good" path through network from source to destination*

Graph abstraction for routing algorithms:

- graph nodes are routers
- graph edges are physical links
- Link cost: delay, physical cost, or congestion level.

e.g.



What does ***"good"*** path mean?

- typically means **minimum cost** path
- other definitions also possible

Routing Algorithms Classification

Global:

- all routers have complete topology and link cost info.
- **"link state" algorithms**

Decentralized:

- router knows physically-connected neighbours, link costs to neighbours only
- iterative process of computation, exchange of info with neighbours
- **"distance vector" algorithms**

Static:

- routes change slowly over time.

Dynamic:

- routes change more quickly
  - periodic update in changes in link costs

An example of a static, "link-state" 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 routing table for that node.
- iterative: after k iterations, know least cost path to k destinations

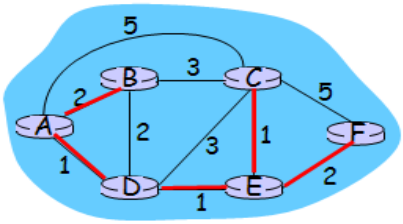
Notation:

- c(i,j): link cost from node i to j. cost infinite if not direct neighbours
- D(v): current value of cost of path from source to destination V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

Pseudo-Code:

```
1  Initialization:
2   N = {A}
3   for all nodes v
4    if v adjacent to A
5      then D(v) = c(A,v)
6      else D(v) = infinity
7
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:
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N
```

e.g.



Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

Dynamic, Distance Vector Routing algorithm

iterative:

- ❖ continues until no nodes exchange info.
- ❖ *self-terminating*: no "signal" to stop

asynchronous:

- ❖ nodes need *not* exchange info/iterate in lock step!

distributed:

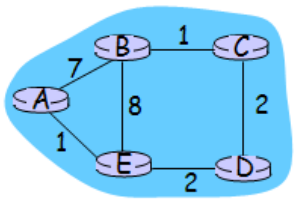
- ❖ each node communicates *only* with directly-attached neighbours

Distance Table data structure

- ❖ each node has its own
- ❖ row for each possible destination
- ❖ column for each directly-attached neighbour to node
- ❖ eg: in node X, for destination Y via neighbour Z:

$$D^X(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop} = c(X,Z) + \min_w\{D^Z(Y,w)\}$$

Example:

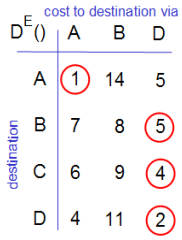


$$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\} = 2+2 = 4$$

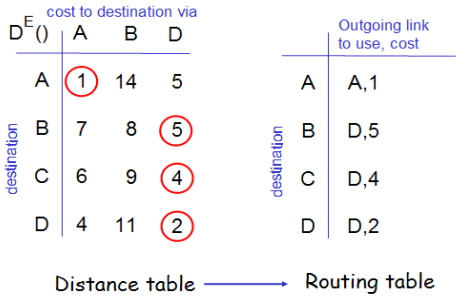
$$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\} = 2+3 = 5 \quad \text{loop!}$$

$$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\} = 8+6 = 14 \quad \text{loop!}$$

Distance Table:



Distance Table gives Routing Table:



Summary

Iterative, asynchronous: each local iteration caused by:

- ❖ local link cost change
- ❖ message from neighbour: its least cost path change from neighbour

Distributed:

- ❖ each node notifies neighbours *only* when its least cost path to any destination changes
- ❖ neighbours then notify their neighbours if necessary

Hierarchal Routing

Our routing study has so far been idealized. But this is not true in practice!!! ***Why?***

1. All routers are not identical
2. Network is not 'flat'.

Assume that there are roughly about **200 millions** nodes (i.e. destinations)
We obviously cannot store all those destinations in routing tables because routing exchange will saturate links.

Solution: **Administrative Autonomy**

- ○ **internet** = network of networks
- ○ each **network administrator** may want to control routing in its own network

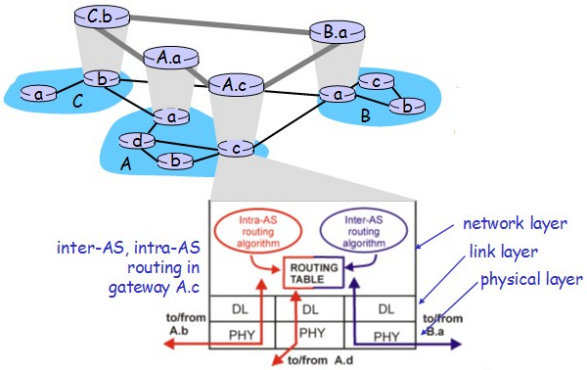
Aggregate routers into regions, "A**utonomous Systems**"
Routers in same **AS** run same routing protocol

- ○ "***intra-AS***" routing protocol
- ○ routers in different AS can run different ***intra-AS*** routing protocol

Gateway/Border Routers

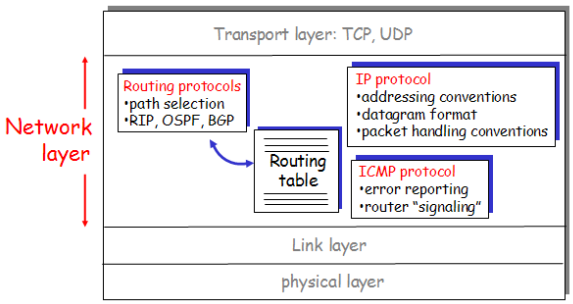
- special routers in AS
- run ***intra-AS*** routing protocol with all other routers in AS
- *also* responsible for routing to destinations ***outside AS***
  - run ***inter-AS*** routing protocol with ***other gateway routers***

Intra-AS and Inter-AS Routing

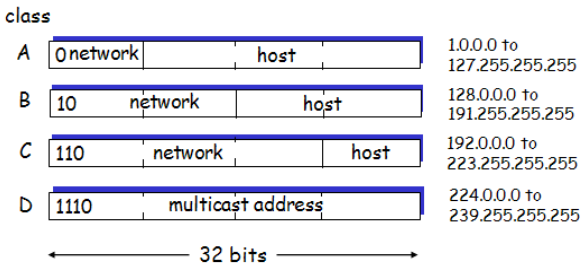


The Internet Protocol (IP) revisited

Network Layer Functions (overview)



IP Addressing "Class-full" (revisited)



Classless Inter Domain Routing (CIDR)

IP has been extremely successful with its exponential growth, but it is running out of address space.
In principle, over 2 billion addresses exist, but in practice millions of them are wasted by classes.
For most organizations, class A with 16 million addresses is too big and class C with 256 addresses is too small.
A class B network with 65,536 is just right.
Studies have shown that more than half of all class has fewer than 50 hosts.
Another problem is table explosion. Routers do not have to know about all the hosts, but they know about other networks.
Having ½ a million class C networks, every router would require a table with ½ a million entries.
The routing table problem can be solved by going to a deeper hierarchy (like telephone), but it requires more than 32-bit for IP addresses.
Most solutions solve 1 problem but create new ones.
One solution currently being implemented is **Classless Inter Domain Routing.**

The basic idea behind CIDR is to allocate the remaining class C networks (almost 2 million) in variable-sized blocks.

- If a site needs 2000 addresses, it is given 2048 addresses (8 contiguous class C networks), and not a full class B address.

In addition to using contiguous blocks, the allocation rules were also changed. The world was partitioned into four zones.

- Europe: 194.0.0.0 to 195.255.255.255
- North America: 198.0.0.0 to 199.255.255.255
- Central and South America: 200.0.0.0 to 201.255.255.255
- Asia and Pacific: 202.0.0.0 to 203.255.255.255

Each region was given 32 million addresses, with another 320 million class C addresses from 204.255.255.255 to 223.255.255.255 reserved for future use. Within each block allocate sub-block to ISP. ISP then allocates to customers.

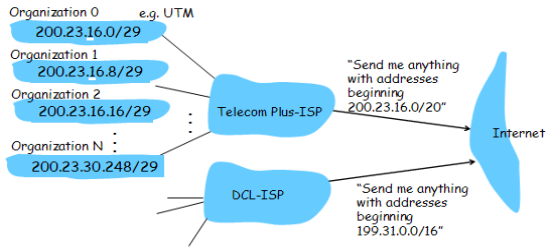
Restrict block sizes to powers of 2

All routers must understand CIDR addressing

Lets see this at work…

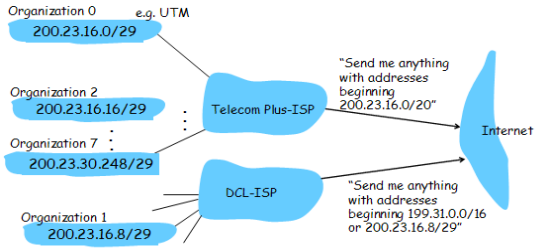
ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/29
Organization 1	11001000 00010111 00010000 00001000	200.23.16.8/29
Organization 2	11001000 00010111 00010000 00010000	200.23.16.16/29
…	….	….
Organization 7	11001000 00010111 00011110 11111000	200.23.30.248/29

Hierarchal Addressing: Route Aggregation



Hierarchical addressing: more specific routes

DCL has a more specific route to Organization 1



How does Host get IP address?

- Hard-coded by system admin in a file
  - Wintel: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config file
- **DHCP**: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play" (more later)

How does an ISP get block of addresses?

- ***ICANN***: **I**nternet **C**orporation for **A**ssigned **N**ames and **N**umbers
  - allocates addresses
  - manages DNS
  - assigns domain names, resolves disputes

Getting a datagram from Source to Destination

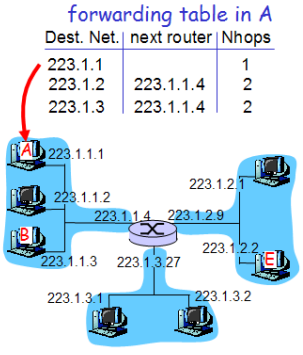
IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Case 1

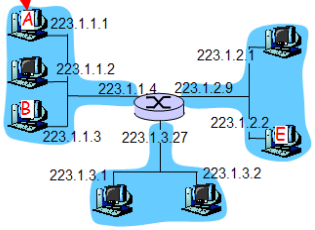
misc fields	223.1.1.1	223.1.1.3	data

Starting at A, send IP datagram addressed to B:

- look up net. address of B in forwarding table
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Case 2

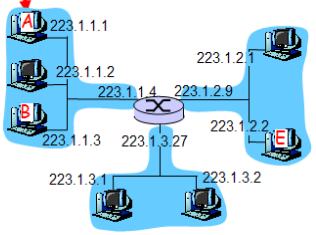
misc fields	223.1.1.1	223.1.2.3	data

Starting at A, dest. E:

- look up network address of E in forwarding table
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued.....

forwarding table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2



Case 3

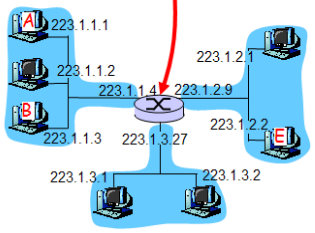
misc fields	223.1.1.1	223.1.2.3	data

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E in router's forwarding table
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)

forwarding table in router

Dest. Net	router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27



IP Datagram Format (revisited)