Final Overview

EECS 122

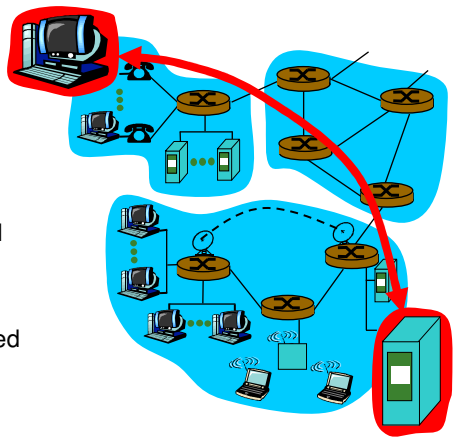
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley

Review: Check List

- Big Picture
  - Layers
  - Where protocols are implemented
  - Switching Techniques
- Applications
  - DNS
  - HTTP
  - SMTP
- Network Layer: Routing
  - Class-Based; Classless Addressing
  - Dijkstra; Bellman-Ford
  - BGP
- Inside Router
  - Architecture: Input, Output
  - Scheduling: Fairness, GPS, WFQ
- Distributed Algorithms
- Overlay Networks

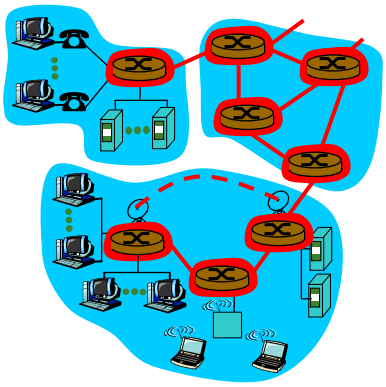
The Network Edge:

- end systems (hosts):
  - run application programs
  - e.g. Web, email
  - at "edge of network"
- client/server model
  - client host requests, receives service from always-on server
  - e.g. Web browser/server; email client/server
- peer-peer model:
  - minimal (or no) use of dedicated servers
  - e.g. Gnutella, KaZaA, Skype

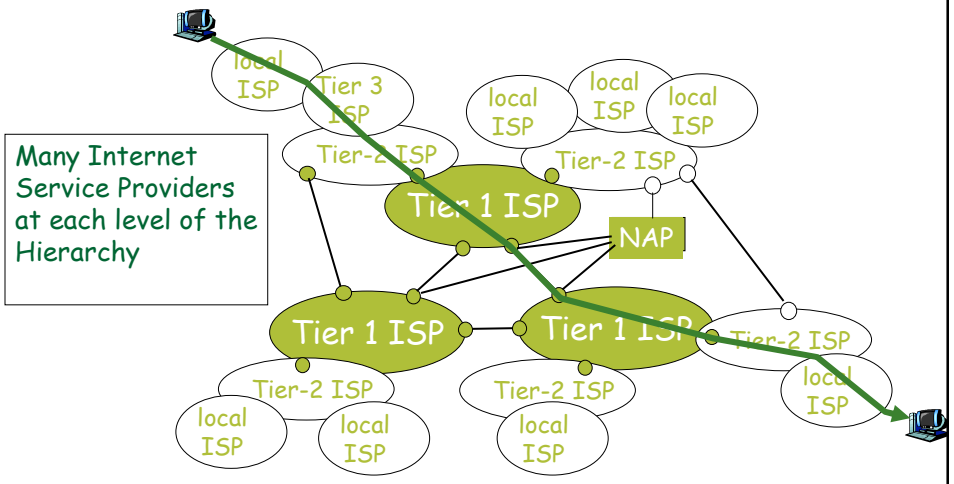


The Network Core

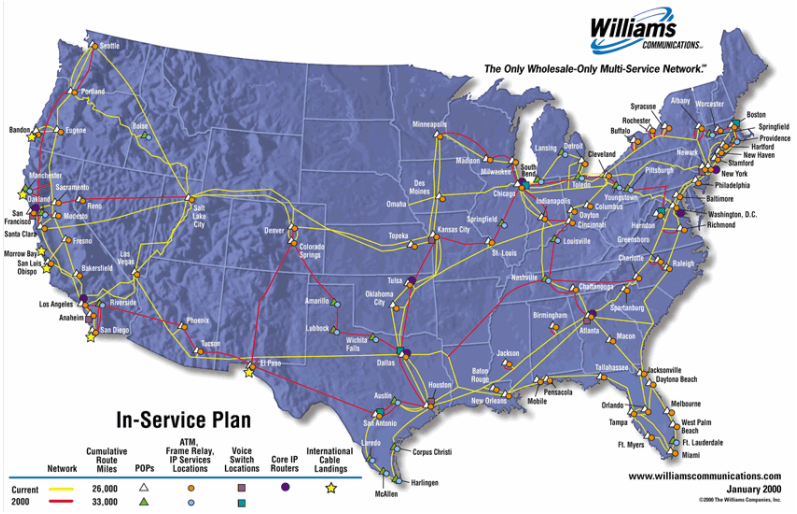
- Many interconnected "sub-networks"
- Many different architectures
- Advertises a "service" to the end devices
  - E.g. Phone network v/s the Internet



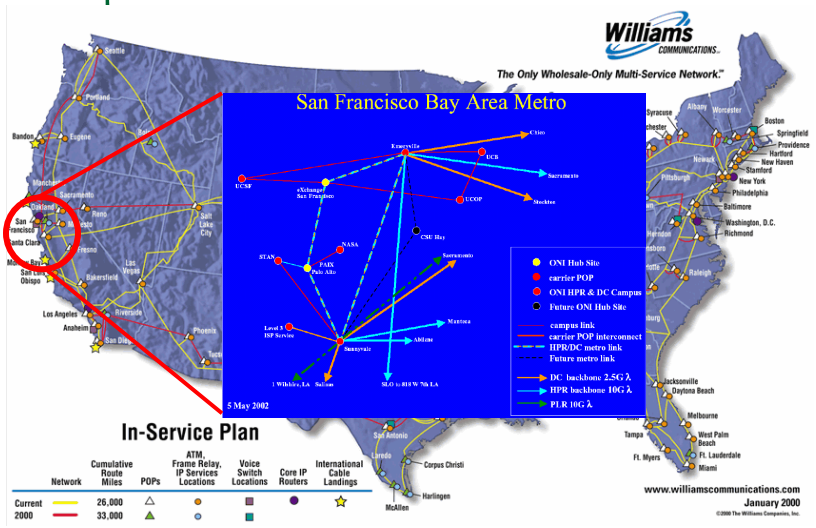
The internet consists of many networks



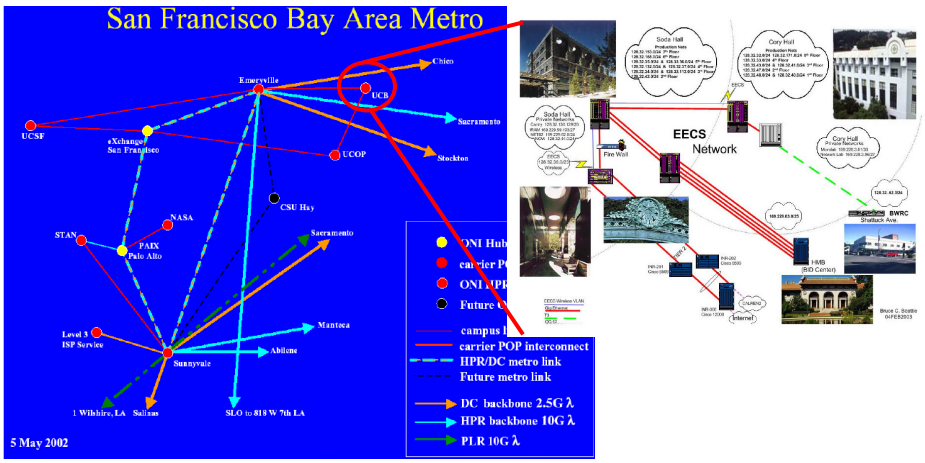
Example: Backbone Network



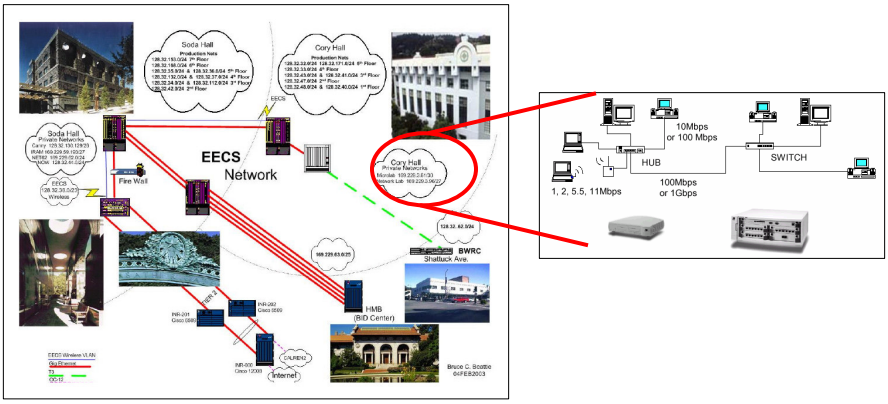
Metropolitan Area Network



Campus Network



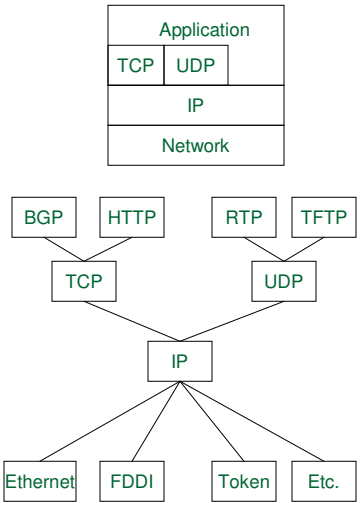
Local Area Network



Two fundamentally different ways to forward information

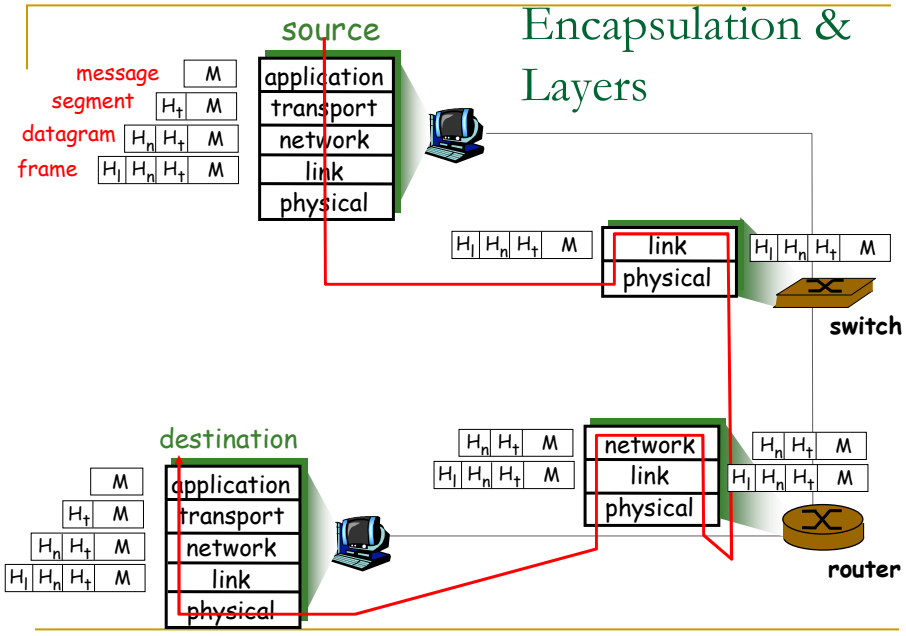
- **Circuit Switched**
  - Information is exchanged in units of "calls"
  - Network resources are reserved for the duration of the call
  - Example: The Phone Network
    - Once a call goes through, subsequent calls cannot degrade call quality
- **Packet Switched**
  - Information is exchanged in units of "packets"
  - Typically, no resources are reserved
  - **Datagram**: Each packet is forward independently
    - Example: The Internet
  - **Virtual Circuit**: All the packets from a given stream take the same path through the network
    - Example: ATM, ISDN, Intserv

Internet Layering

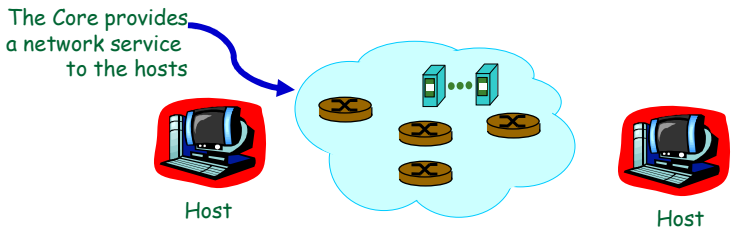


- Almost Any kind of application can write directly on IP
  - Including new transport protocols
- IP cannot be avoided
- As long as the routers speak IP, any application that can make do with datagram service can be written and implemented on the end devices.
  - No co-ordination, standards activity etc. is required!!

Encapsulation & Layers

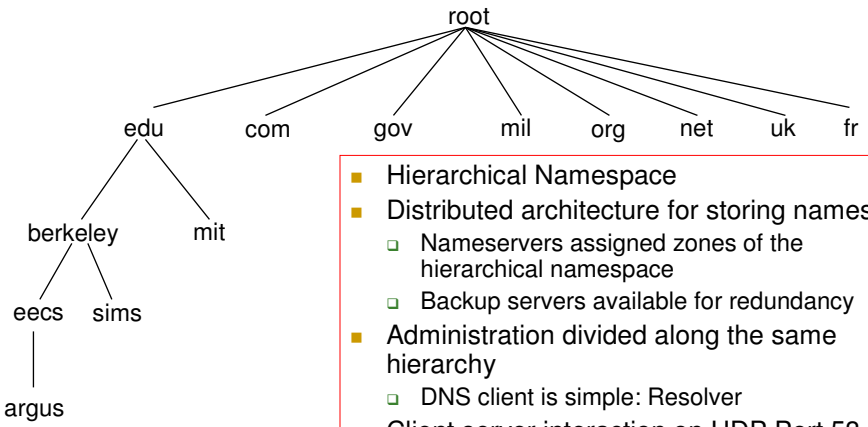


Application Protocols



- Host-Host:
  - HTTP, SMTP
- Host-Network:
  - DNS
- Network-Network:
  - Routing Protocols (e.g. OSPF)

DNS Features



- Hierarchical Namespace
- Distributed architecture for storing names
  - Nameservers assigned zones of the hierarchical namespace
  - Backup servers available for redundancy
- Administration divided along the same hierarchy
  - DNS client is simple: Resolver
- Client server interaction on UDP Port 53 (but can use TCP if desired)

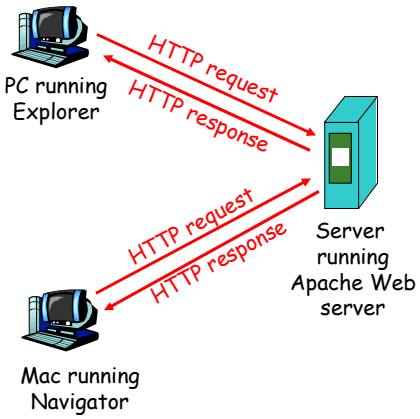
How does a name get resolved

- Query "walks" its way up and down the hierarchy
  - Iterated query
    - I don't know, but here's who to ask next
  - Recursive query
    - I don't know right now, but I'll get back to you…

HTTP

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.0: Non Persistent
- HTTP 1.1: Persistent



Persistent HTTP

- server leaves connection open after sending response
  - TCP overhead minimized
- subsequent HTTP messages between same client/server sent over open connection

No pipelining:

- client issues new request only when previous response has been received
- one RTT for each referenced object

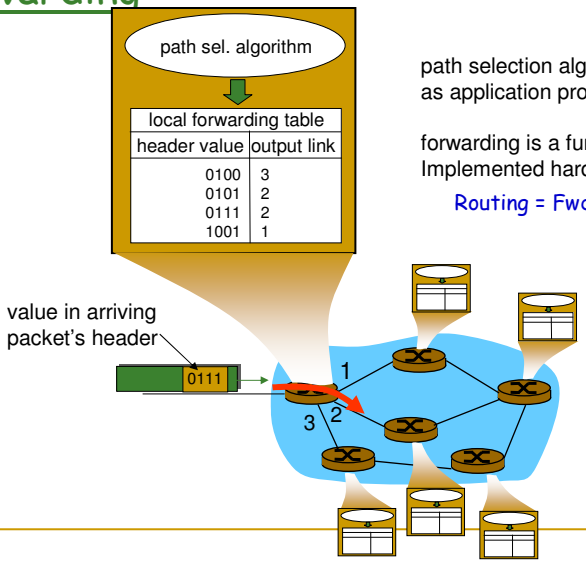
Pipelining:

- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects
- default in HTTP/1.1

Network Layer

- Control Functions: Ensure that routers are configured to deliver packets correctly to the destination
  - Path Selection (called routing in the book)
  - Connection Setup: required in virtual circuit routing.
- Data Functions: Ensure that arriving packets are forwarded correctly within a router with minimum delay
  - Forwarding

Interplay between path selection and forwarding



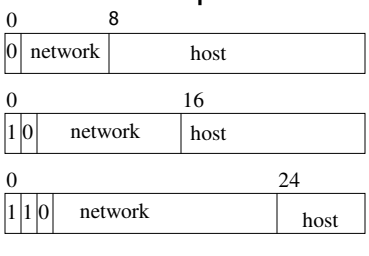
path selection algorithms run as application protocols

forwarding is a function mostly Implemented hardware

Routing = Fwding + Path Selection

Class-base Addressing

- Addressing reflects internet hierarchy
  - 32 bits divided into 2 parts:
  - Class A
  - Class B
  - Class C

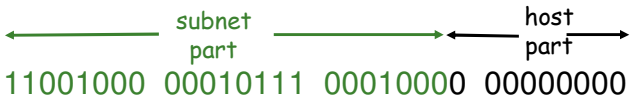


~2 million nets
256 hosts

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- net portion of address of arbitrary length: subnet
- address format: a.b.c.d/x, where x is # bits in subnet portion of address



200.23.16.0/23

CIDR: Example

- Example 128.5.10/23
  - Common prefix is 23 bits: 01000000 00000101 0000101
  - Number of addresses: $2^9 = 512$
- Prefix aggregation
  - Combine two address ranges
  - 128.5.10/24 and 128.5.11/24:
  - 01000000 00000101 00001010
    01000000 00000101 00001011

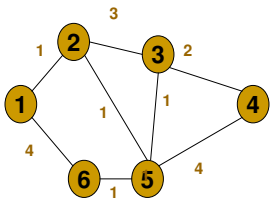
  gives 128.5.10/23
- Routers match to longest prefix

Link State Protocols

1. Every node learns the topology of the network
   - Flooding of Link State Packets (LSP)
2. An efficient shortest path algorithm computes routes to every other node
3. Node updates Forwarding Table

Route Computation: Dijkstra

- Every node knows the graph
  - All link weights are >= 0
- Goal at node 1: Find the shortest paths from 1 to all the other nodes.
- Each node computes the same shortest paths so they all agree on the routes
- Strategy at node 1: Find the shortest paths in order of increasing path length
  - List the nodes in increasing order of (shortest) distance
  - S(k): closest k nodes
  - Iteration k yields S(k) and a way to get there



S(1)={1}
S(2)={1,2}
S(3)={1,2,5}

Distance Vector Algorithms

- Nodes communicate distance estimates to their neighbors, not topology information
- Based on the Bellman Ford Equation:

Define D(x,y) to the shortest distance from x to y.

$$D(x,y) = \min_{v \in N(x)} \{c(x,v) + D(v,y)\}$$

where N(x) are the neighbors of node x.

- Why is this true?

Let D(x,v,y) be the shortest path from x to y where the first node after x is v.

Then D(x,v,y) = c(x,v) + D(v,y).

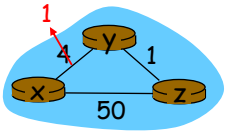
$D(x,y) = \min_v D(x,v,y)$

$= \min_v \{c(x,v) + D(v,y)\}$

Distance Vector: link cost changes

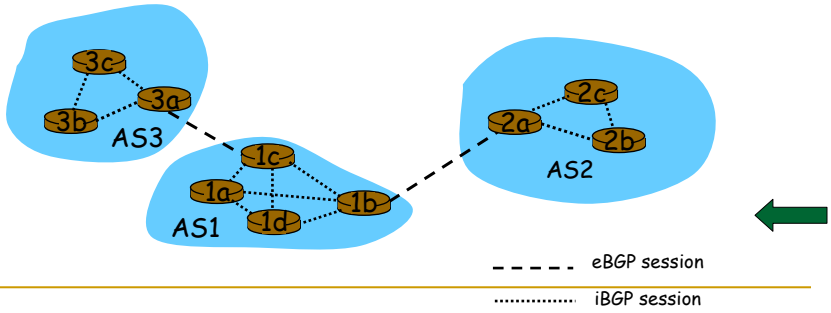
Link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- Good news travels fast but Bad news can travel very slowly….Counting to infinity



BGP

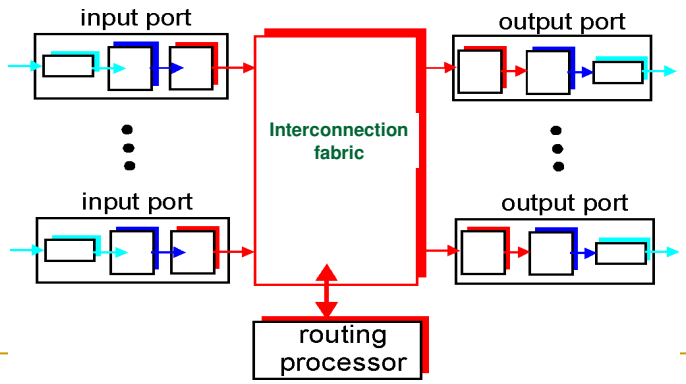
- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement



Router Architecture Overview

Two key router functions:

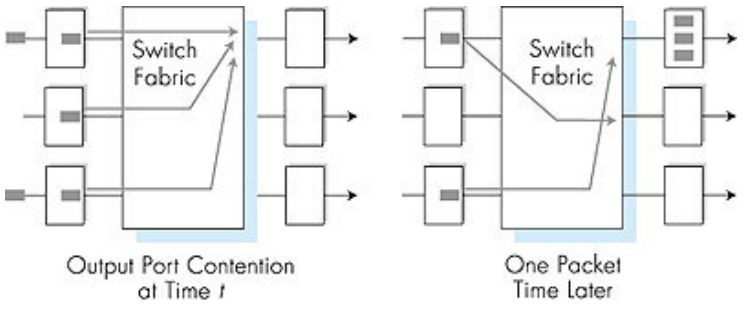
- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



The Forwarding Decision Process

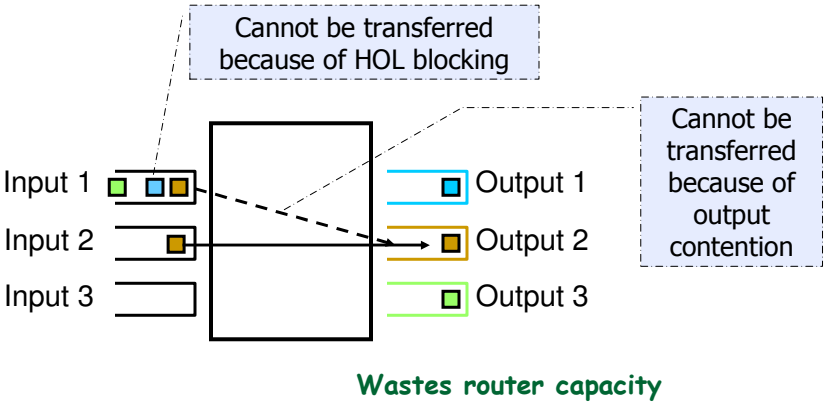
- Datagram Routing: Each packet is independently forwarded at each router
  - Must look up IP address ranges
  - Match Longest Prefix
- Virtual Circuit Routing:
  - call setup, teardown for each call *before* data can flow
  - each packet carries VC identifier (not destination host address)
  - *every* router on source-dest path maintains “state” for each passing connection
  - link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

Output Queued Routers



Input Queues: Head-of-line Blocking

- The packet at the head of an input queue cannot be transferred, thus blocking the following packets



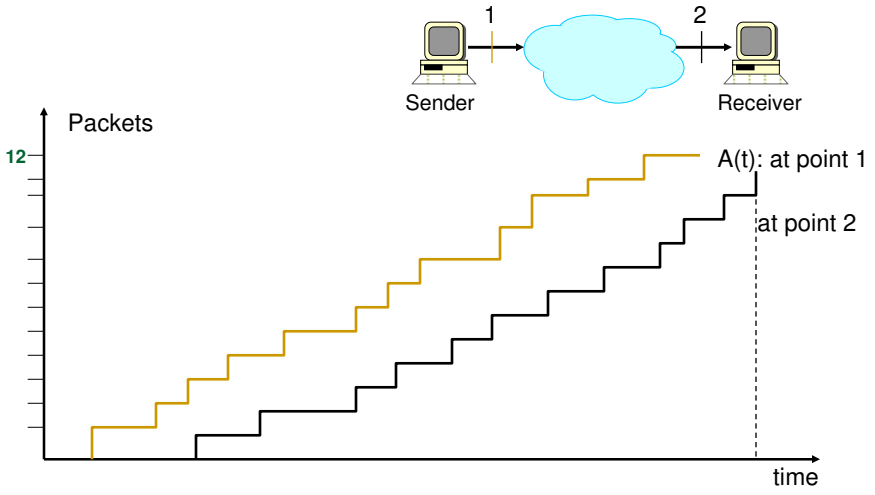
Wastes router capacity

Components of Per Hop Delay

- **Propagation delay:** time it takes the signal to t from source to destination
- **Packet transmission time:** time it takes the sender to transmit all bits of the packet
- **Queuing delay:** time the packet need to wait before being transmitted because the queue was not empty when it arrived
- **Processing Time:** time it takes a router/switch to process the packet header, manage memory, etc

Only random component

Delays and Queues

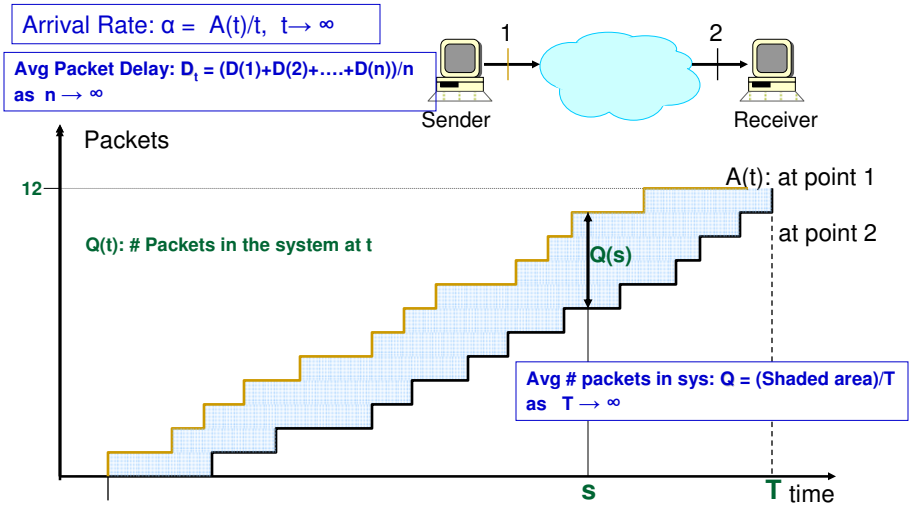


Delays and Queues

Avg Occupancy = Area

Arrival Rate: $\alpha = A(t)/t, \ t \rightarrow \infty$

Avg Packet Delay: $D_t = (D(1)+D(2)+....+D(n))/n$ as $n \rightarrow \infty$



Little's Law

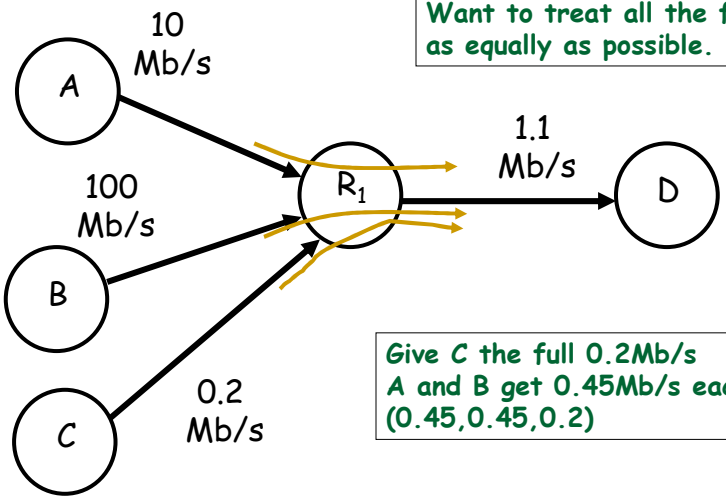
- Shaded Area up to time T is equal to both
1. D(1)+D(2)+….+D(A(T))
2. $\int_0^T$ Q(t) dt
- Divide and multiple 1 by A(T):
  1. [D(1)+D(2)+….+D(A(T)) / A(T)] A(T)
- Divide both (rewritten) 1 and 2 by T and take limits
  - Q = D α

average occupancy = (average Delay) X (average arrival rate)

Flows and Fairness

Max-Min Fair Allocation

Want to treat all the flows as equally as possible.



Give C the full 0.2Mb/s
A and B get 0.45Mb/s each
(0.45,0.45,0.2)

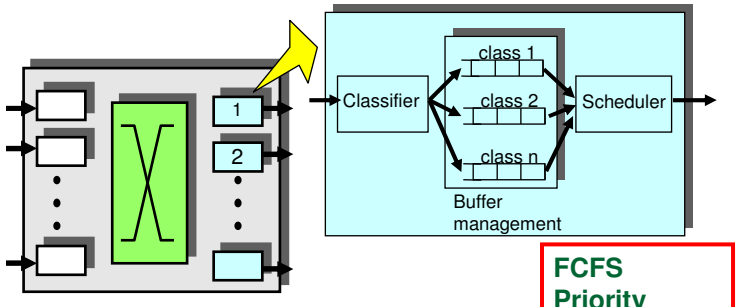
Mechanisms to Improve Best Effort

- Classification and Scheduling
- Drop Policies
- Call admission
- Policing

- Implementing even a subset of these can help!

Advanced Queuing Functions

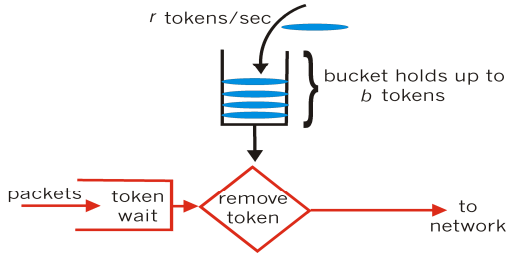
- Packet classification: map each packet to a predefined class
  - use to implement more sophisticated services (e.g., QoS)
- Flow: a subset of packets between any two endpoints in the network



FCFS
Priority
Round Robin
WFQ

Policing Mechanisms

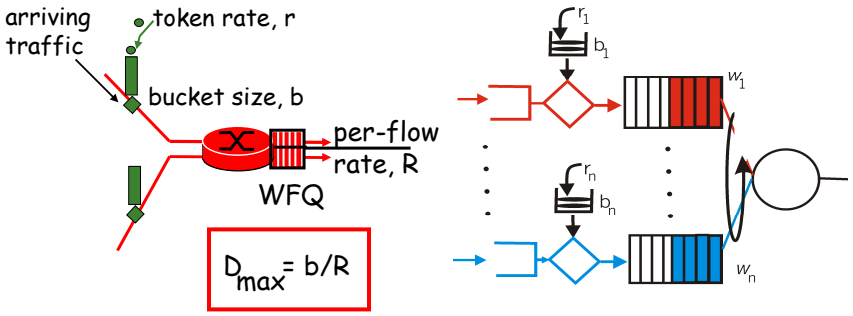
Token Bucket: limit input to specified Burst Size and Average Rate.



- bucket can hold b tokens
- tokens generated at rate *r token/sec* unless bucket full
- *over interval of length t: number of packets admitted less than or equal to (r t + b).*

Performance Guarantees: Flows+Policing+Scheduling

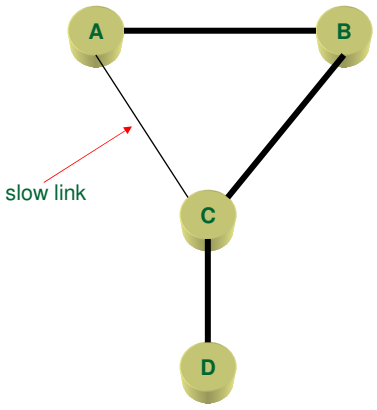
- Policing
- Scheduling



Modeling Issues

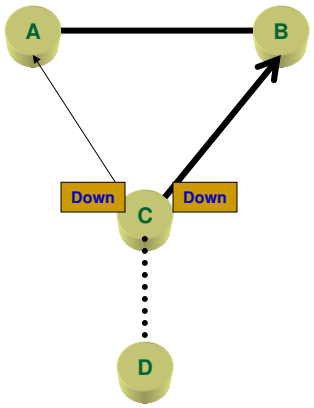
- Error correction
  - Assume that errors can "eventually" corrected
- Propagation Delay
  - Fixed
  - Variable but no more than d
  - Variable with no upper bound
- Other components of delay
  - Queueing Delay
  - Transmission Delay
- Packet order
  - FIFO
  - Can be delivered in arbitrary order

Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

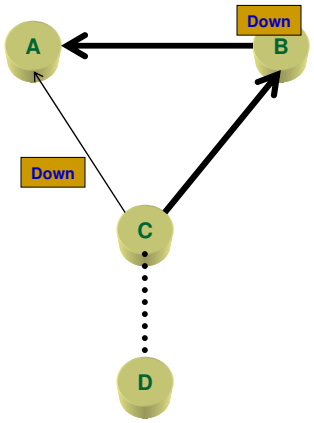
Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood

1. CD fails

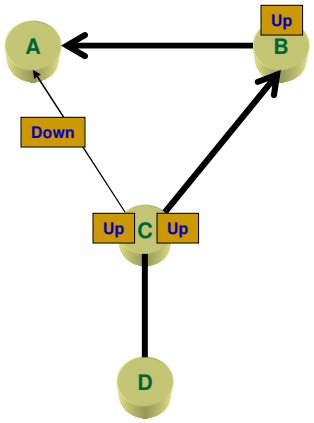
Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

1. CD fails
   - A marks the link down

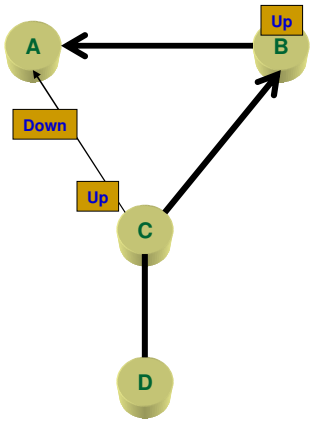
Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

1. CD fails
   - A marks the link down
2. CD comes back up

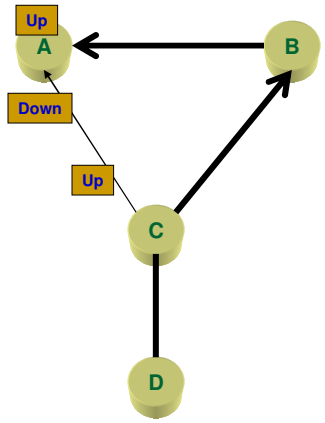
Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

1. CD fails
   - A marks the link down
2. CD comes back up

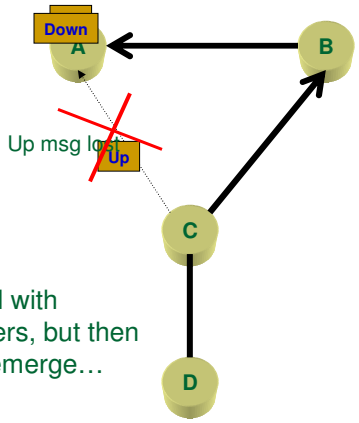
Maintaining accurate topology information



Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

1. CD fails
   - A marks the link down
2. CD comes back up
   - A marks the link up
3. A marks the link down

Maintaining accurate topology information



This can be fixed with sequence numbers, but then other problems emerge…

Whenever a link goes down/up, its end points send messages to all their neighbors who then flood.

1. CD fails
   - A marks the link down
2. CD comes back up
   - A marks the link up
3. A marks the link down
4. CA fails
   - Up message lost

A thinks CD is down when it is actually up!

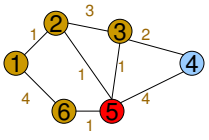
Synchronous v/s Asynchronous Algorithms

- Synchronous algorithms can be described in terms of global iterations. The time taken for a given iteration is the time taken for the slowest processor to complete that iteration: *time driven*
  - E.g. TDM or SONET
- Asynchronous algorithms execute at a processor based on received messages and internal state: *event driven*
  - *E.g.* IP protocols which must run over heterogeneous systems

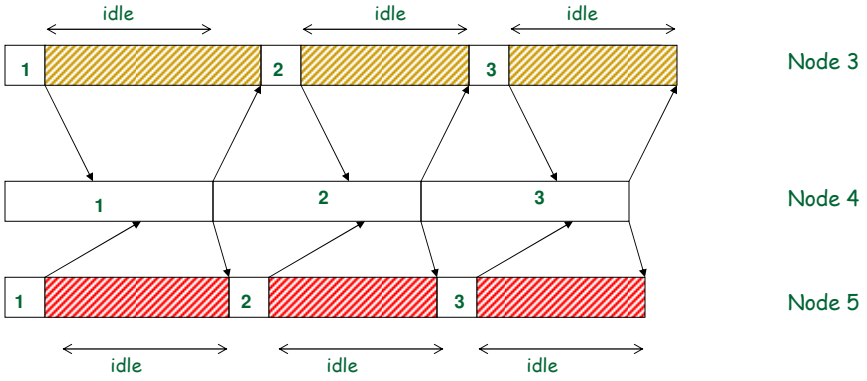
Implementing a Synchronous Algorithm

- Suppose the slowest process can complete an iteration in time T_p
- Link delay is always less than T_l
- Then a slot size of T_p+T_l or more is sufficient
  - But most processors may be idle most of the time
- What if T_p and or T_l are not known?

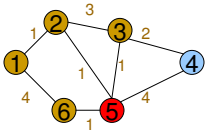
Local Synchronization



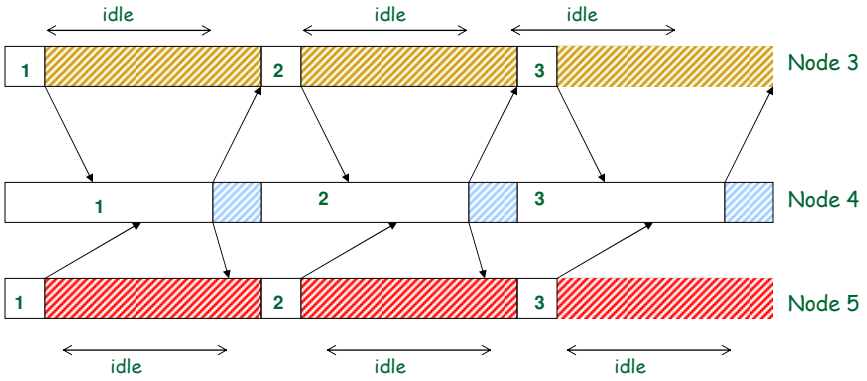
Send update k after you've heard update k-1 from all neighbors.



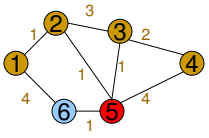
Compare with Synchronous



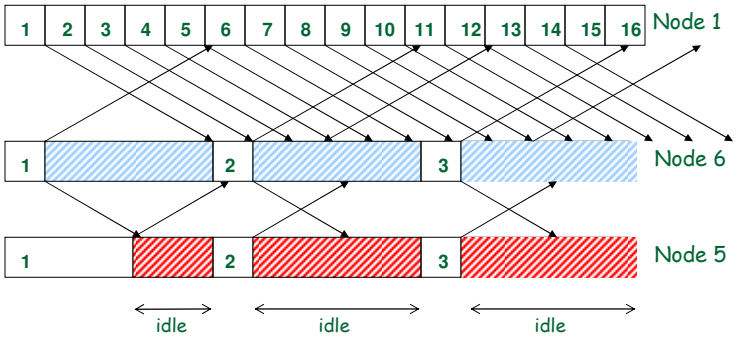
Slot size is affected by the slow node 4



Asynchronous computation



No notion of "slot size" at all!



Why should this work?

Why bother with Asynchronous Algorithms

- To reduce the synchronization penalty
- Difficult to get the synchronous algorithm to start
- The network is dynamic
  - Flows
  - Topology
    - Think of the algorithm having to "restart" with a new set of initial conditions, every time there is a failure
- Changes create "events" which may or may not have global impact
  - Event-driven algorithms better suited

Soft State

- State with Time-Out
- Example: A host joins a group by sending a "join" message to a "host manager". The manager adds the host to the group for the next T seconds. If the host wants to stay in the group it must send a refresh message within T seconds to the manager. Otherwise it is dropped.
- Advantage: Manager robust to host failure
- Disadvantage: Too many messages
- Most internet protocols use this way of communicating
- Trades of simplicity of correctness with complexity of communication

Kinds of Overlay Networks

- Two kinds of Overlays
  1. Only Hosts: Peer to Peer Networks (P2P)
     - Example: Napster, Gnutella, KaZa
  2. Only Gateway nodes: Infrastructure Overlays
     - Content Distribution Networks (CDNs)
       - Example: Akamai
- Overlay node structure
  - Regular: CAN
  - Adhoc: Gnutella
  - Hybrid: KaZa
- Functions
  - Route Enhancement: Better QoS, Application Level Multicast
  - Resource Discovery: P2P

Content Addressable P2P Networks (CAN)

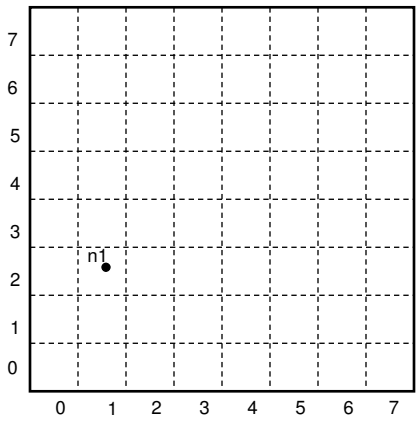
- CAN is one of several recent P2P architectures that
  - imposes a structure on the virtual topology
  - uses a distributed hash-table data structure abstraction
    - Note: item can be anything: a data object, document, file, pointer to a file…
  - routes queries through the structured overlay
  - attempts to distribute (object, location) pairs uniformly throughout the network
  - supports object lookup, insertion and deletion of objects efficiently.
- Others: Chord, Pastry, Tapestry

Content Addressable Network (CAN)

- Associate with each node and item, a unique *id* in an *d*-dimensional space
  - Example for d=3: A node might be called (1,11,5)
  - Example for d=3: A song might be called (2,3,11)
- Properties
  - Routing table size O(*d*)
  - Guarantee that a file is found in at most $d*n^{1/d}$ steps, where *n* is the total number of nodes

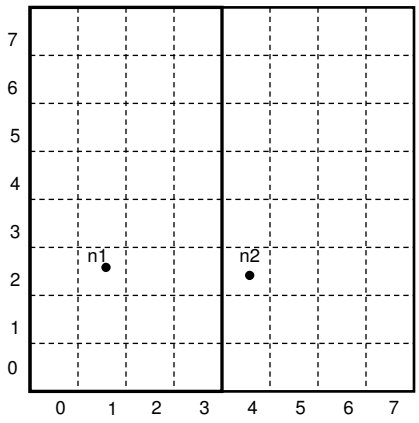
CAN Example: d=2

- Space divided between nodes
- All nodes collectively "cover" the entire space
- Each node covers either a square or a rectangular area of ratios 1:2 or 2:1
- Example:
  - Assume space size (8 x 8)
  - Node n1:(1, 2) first node that joins → cover the entire space



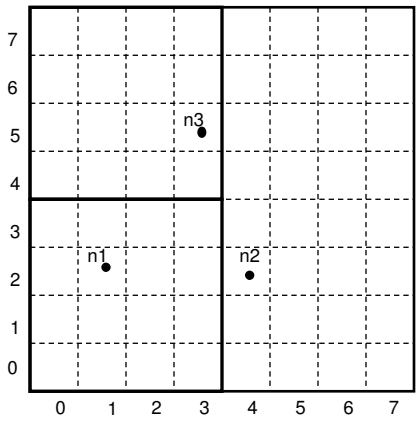
Covered space divided between nodes

- Node n2:(4, 2) joins → space is divided between n1 and n2



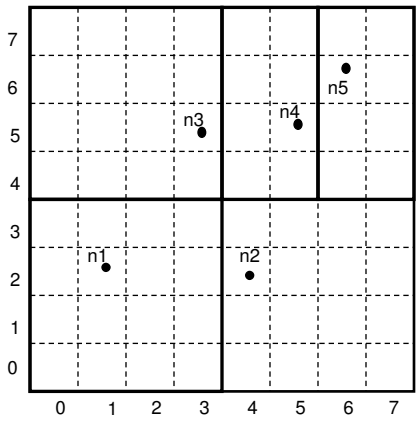
Nodes continue to join

- Node n2:(4, 2) joins → space is divided between n1 and n2
- Node n3: (3,5)



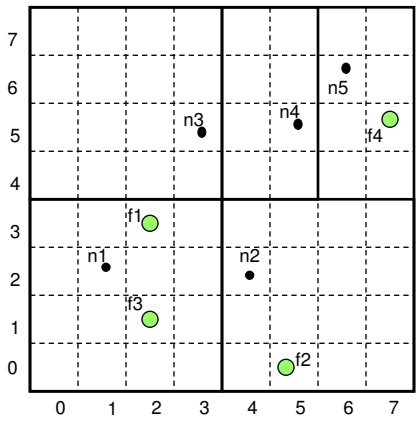
Nodes continue to join

- Nodes n4:(5, 5) and n5:(6,6) join



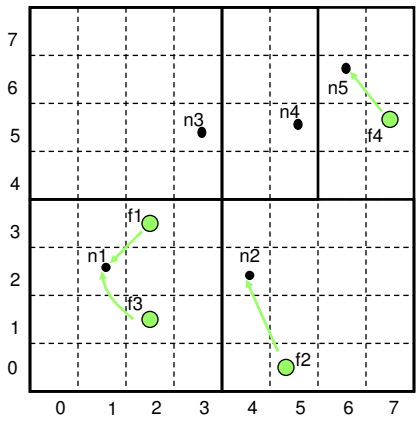
Items are also mapped in the same space

- Items: f1:(2,3); f2:(5,1); f3:(2,1); f4:(7,5);



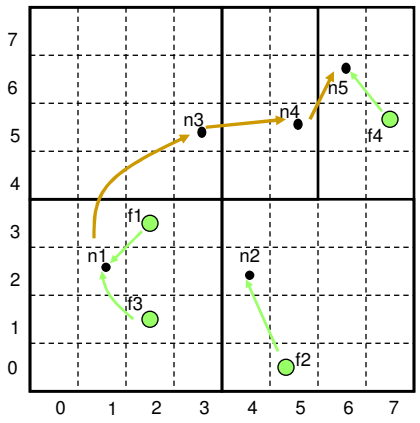
CAN Example: Two Dimensional Space

- Each item is stored by the node who owns its mapping in the space

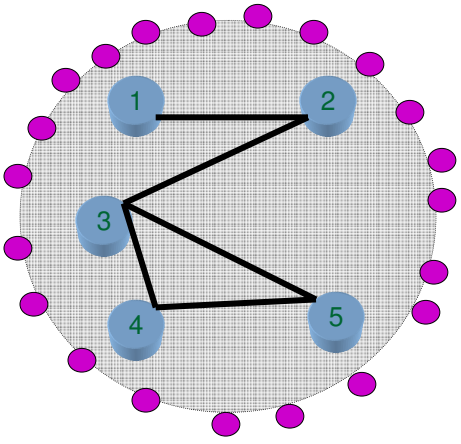


CAN: Query Example

- Each node knows its neighbors in the *d*-space
- Also knows the d-space controlled by its neighbors
- Forward query to the neighbor that is closest to the query *id*
- Example: assume n1 queries f4

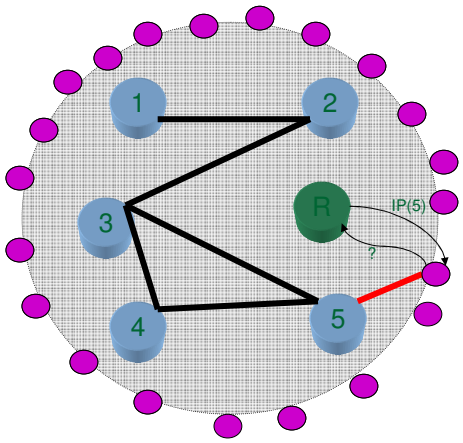


Infrastructure Overlays



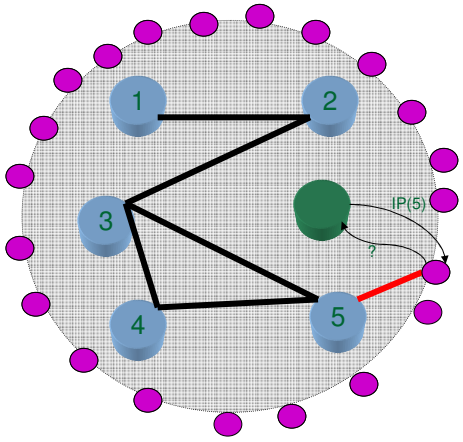
- Overlay network users are not directly connected to the overlay nodes
  - E.g. Akamai

Overlay Routing: Edge Mapping



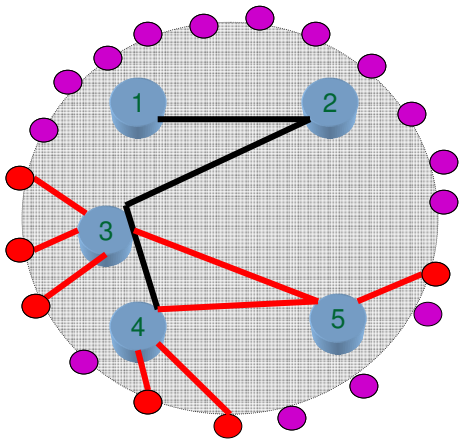
- Overlay network users are not directly connected to the overlay nodes
  - E.g. Akamai
- User must be redirected to a "close by" overlay node
- Edge-Mapping, or redirection function is hard since
  - # potential users enormous
  - User clients not under direct control

Overlay Routing: Edge Mapping



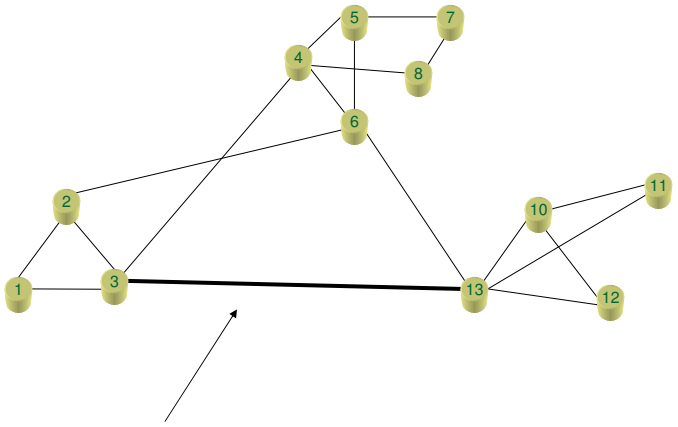
- Overlay nodes interconnect clients
- Enhance nature of connection
  - Multicast
  - Secure
  - Low Loss
- Much easier to add functionality than to integrate into a router

Overlay Routing: Adding Function to the route



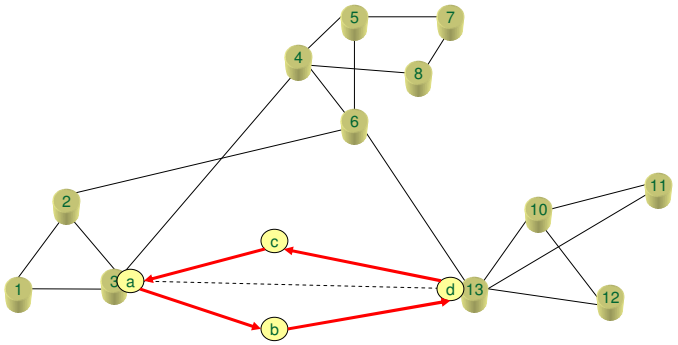
- Overlay nodes interconnect clients
- Enhance nature of connection
  - Multicast
  - Secure
  - Low Loss
- Much easier to add functionality than to integrate into a router
- Overlay nodes can become bottlenecks

Overlay Concept: Going Down



Need this link to be very reliable and fast!

IP Network is the Overlay…



IP Routers 3 and 13 attach to a virtual circuit network
e.g. ATM
The IP network "sees" the virtual circuit network as a link
This is called "Link Virtualization" and is commonly deployed