

Network Topologies

CS 118

Computer Network Fundamentals

Peter Reiher

Outline

- What is network topology?
- Types of network topologies
- Issues in relaying messages

Network Topologies

- So you've got a bunch of nodes
- And you can connect the nodes with channels
- So you can communicate to everyone with fewer total channels
 - And avoid limitations of broadcast media
- Which nodes do you connect?
- That's the question of network topology

Parties and nodes

- Shannon talked about parties
 - Entities that wished to communicate
 - Generally a sender and receiver
 - Very general
 - Machines, people, whatever
- In networking, we usually talk about *nodes*
 - Physical machines used to move information
- We'll be changing to talk primarily about nodes

Channels and Links

- Shannon talked about channels
 - Logical connections between two parties
- Physical networks tend to talk about *links*, instead
 - A physical connection between two nodes
- We'll be changing to talk primarily about links
- In network topologies, links connect nodes

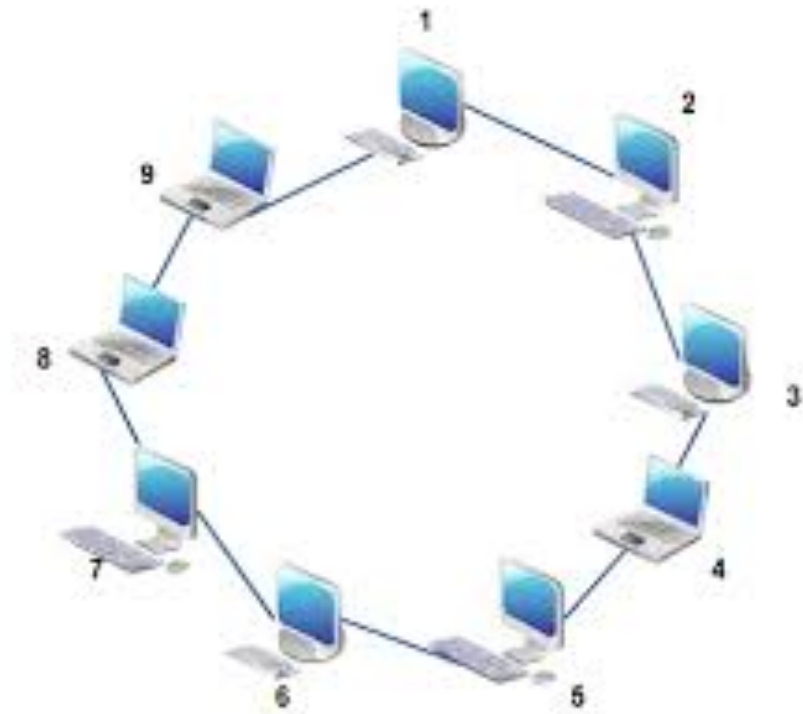
Some simple topologies

- Ring
- Hub and spoke
- Regular mesh
- Manhattan network
- Torus
- Hypercube

Ring

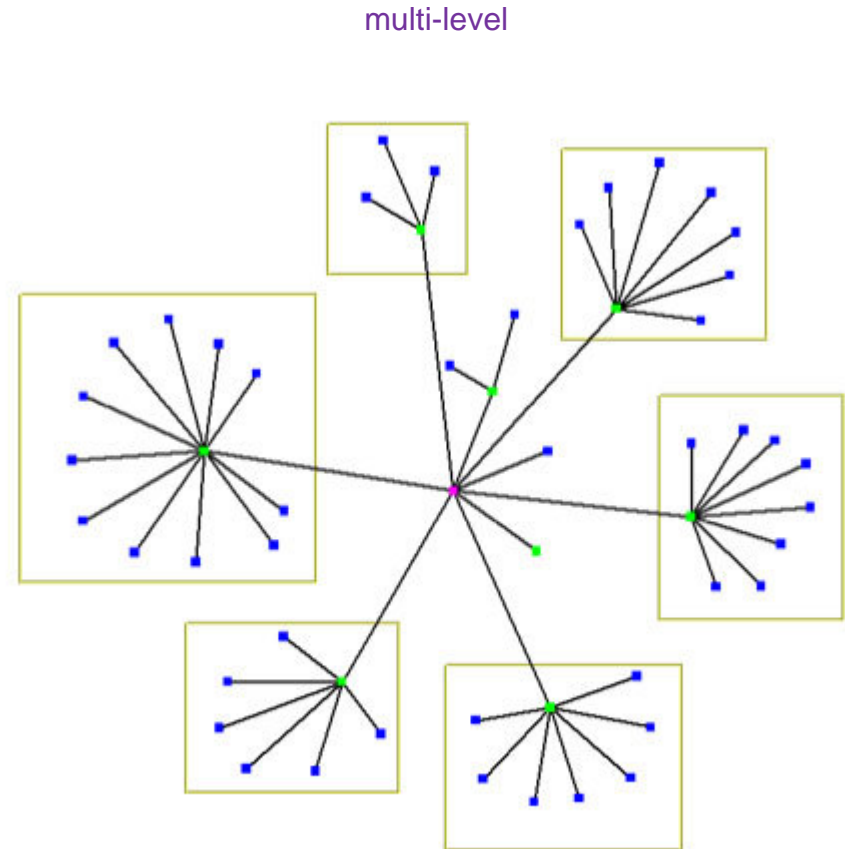
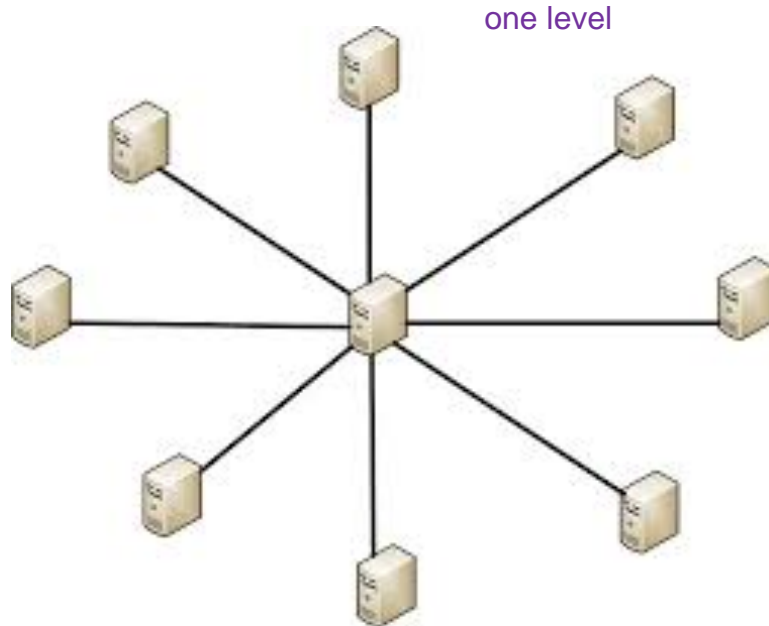
- Connected in a circle
 - N links
 - Can tolerate 1 link or node failure
 - Links can be simplex (one direction)
- Managing failures
 - “Pass through” on node failure
 - Dual-ring (duplex) allowing ends to “heal”

unidirectional links



Hub and spoke

- Node as a switch
 - One or multiple levels

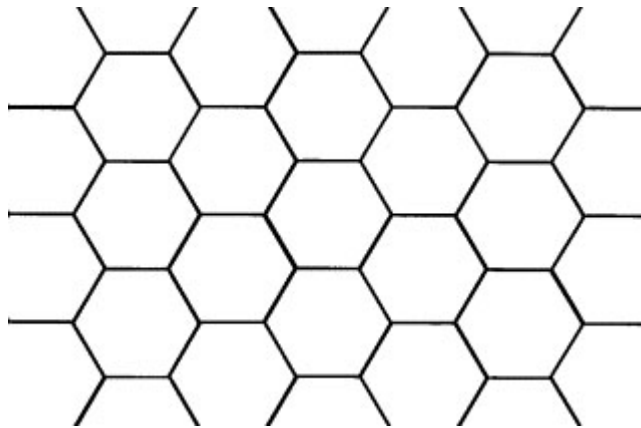


- Failure of the hub disconnects entire network

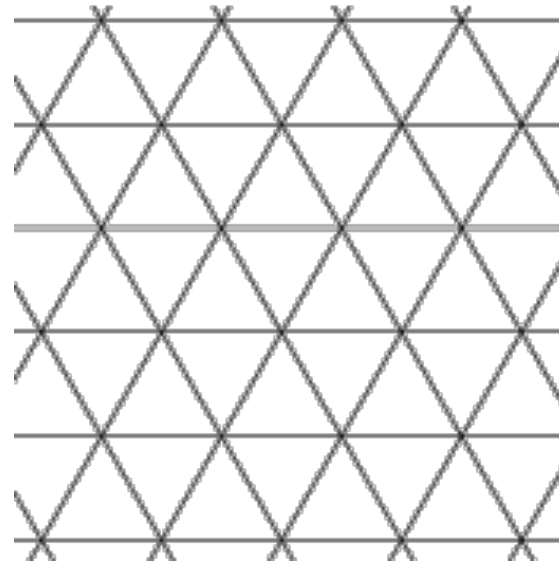
Regular mesh

match nodes into intersection of lines

- Any pattern
 - Nodes where lines meet



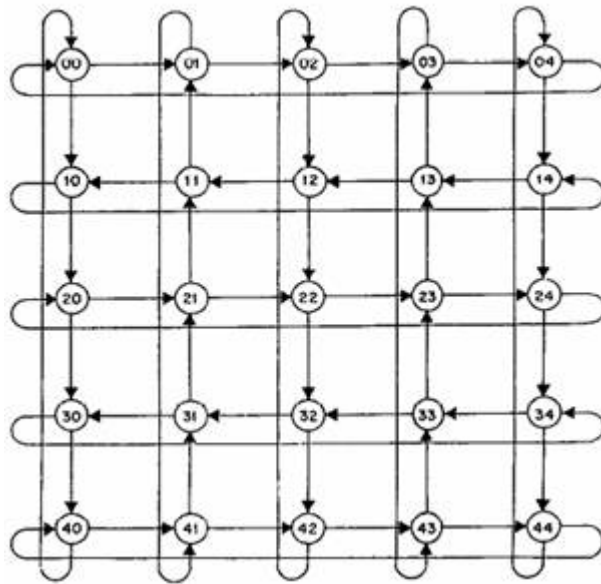
6 nodes on each hexagon



- Depending on pattern, tolerates multiple failures

Manhattan network

- Grid of squares
 - Tend to wrap ends around (2D torus)



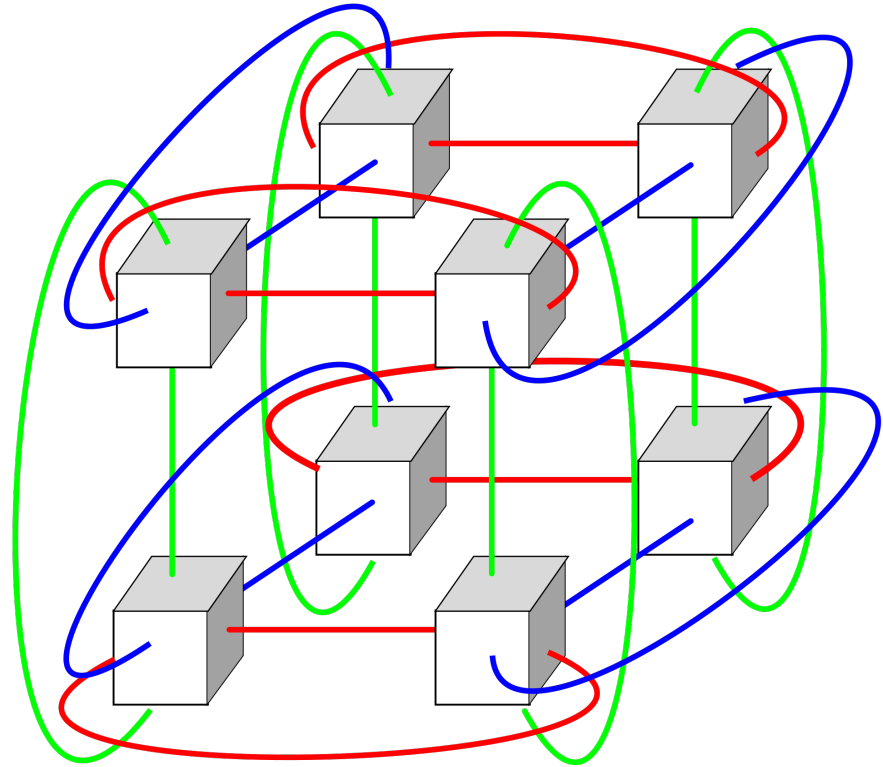
grid of squares - most streets are NS, EW



Torus

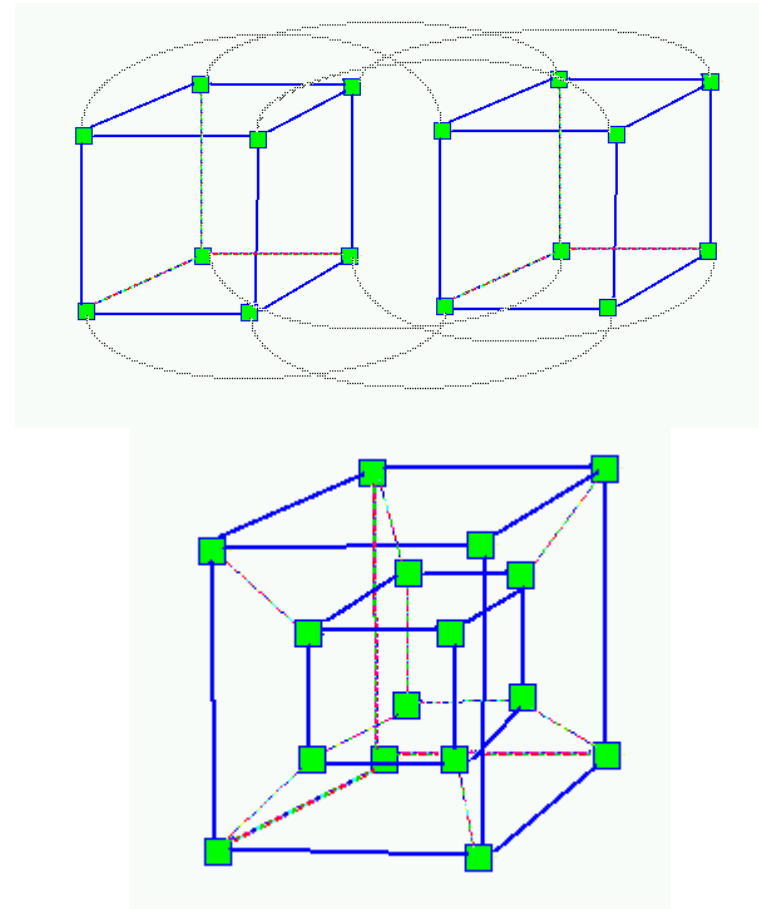
multi-dimension torus

- Each dimension is a set of rings
 - 1D = ring
 - 2D = Manhattan network
 - 3D (see right)
 - ...



Hypercube

- $\log_2 N$ links per node
 - 1D = 2-party (line)
 - Take 2 nodes, link corresponding nodes
 - 2D = square (ring)
 - Take 2 lines, link corresponding nodes
 - 3D = cube
 - Take two squares, link corresponding nodes
 - 4D (see right)
 - Take two cubes, link corresponding nodes
 - At each step, number of nodes doubles but links per node increases by 1



Multistage Interconnection Net (MIN)

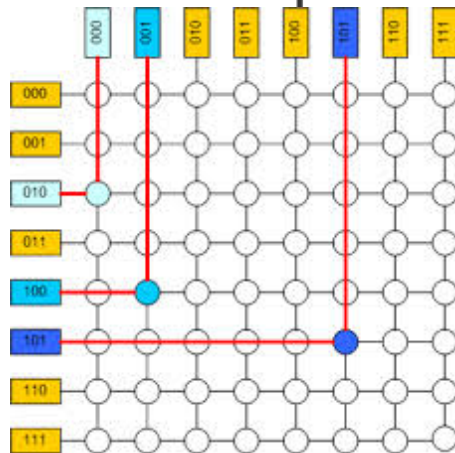
- Stages of nodes
 - Either new nodes, or nodes “reused”
 - Entire stage “switches” at the same time
- Phased switching implications
 - Fixed message size (“cells”)
 - “Rearrangeable” vs. “not”
- E.g., butterfly/banyan

Not usually used as general relays, but often used
“inside the black box” to emulate a link

Examples

- **Crossbar**

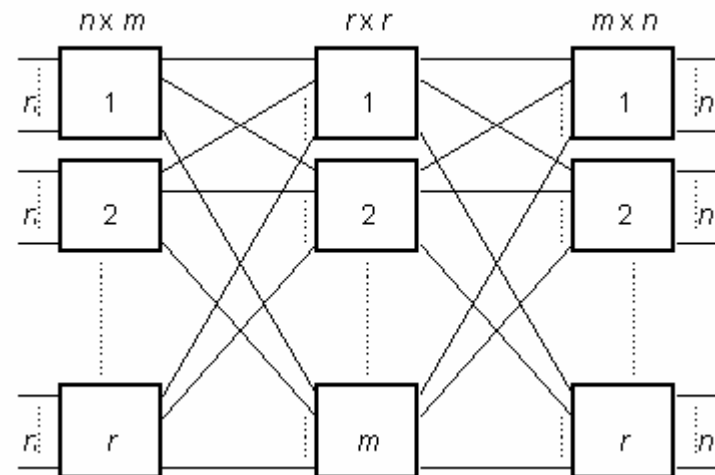
- Original phone switch
- The “anti-MIN”
- N^2 scale problem



use technology inside this box

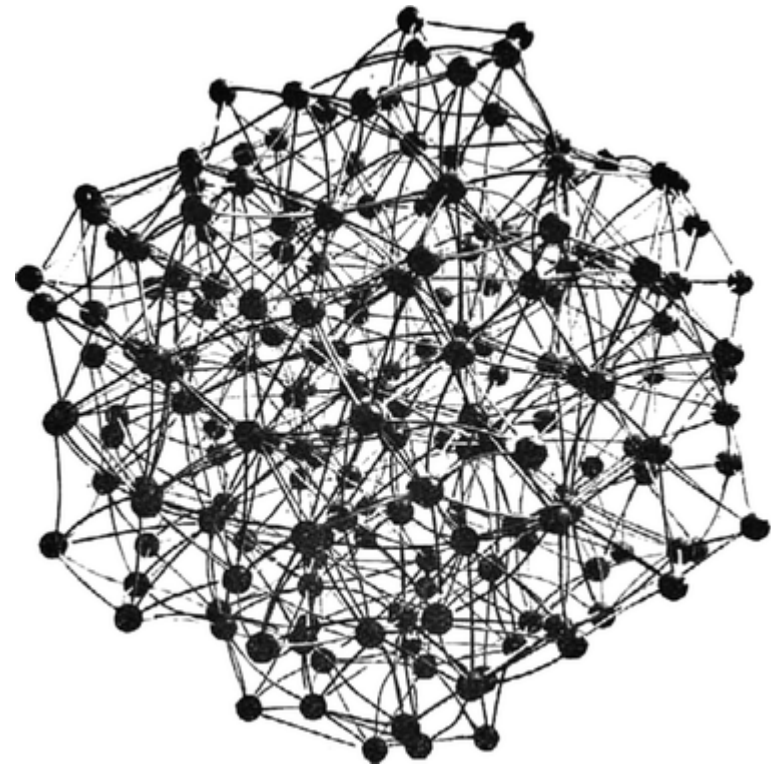
- Clos

- 3-stage: in, middle, out



Irregular mesh

- Lacking a pattern
 - Euclidean or NOT
- Recall from your math
 - Euclidean means the triangle inequality:
 $|A:B| + |C:D| \geq |A:C|$
 - Affects path selection



Impact of topology

topology of the node - know which key go on each enode

- Naming
 - Node labels can describe location
 - Location can indicate route
- Use cases
 - Distributed problems can be mapped to nodes
 - Communication patterns can align to topology
 - E.g., ATM cash machines and the bank map well to a hub-and-spoke

Scale and size

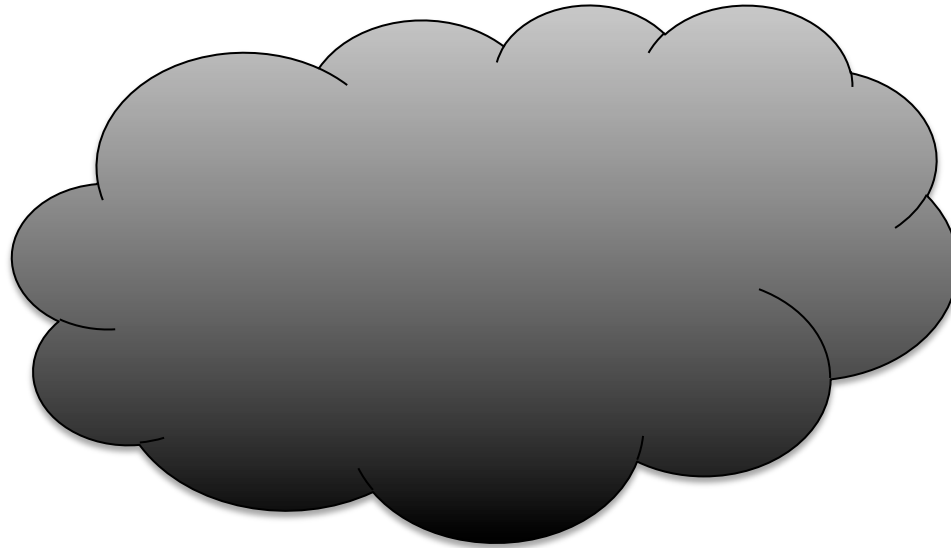
- Limits of a single link
 - Power limits distance
 - Can fix with repeaters, but they're not free
- Limits of a shared link
 - Protocol limits distance
 - Fan-out power limits node count complexity: more nodes we have more computer totlogiy calculatkoins
- Limits of relaying
 - No distance limit (relay using nodes)
 - No node count limit (fan-out using nodes)
 - Complexity (nodes do more work)
 - Overhead (communication takes more time)

detect the signal makes it bigger and send out

WAN

Internet is a WAN

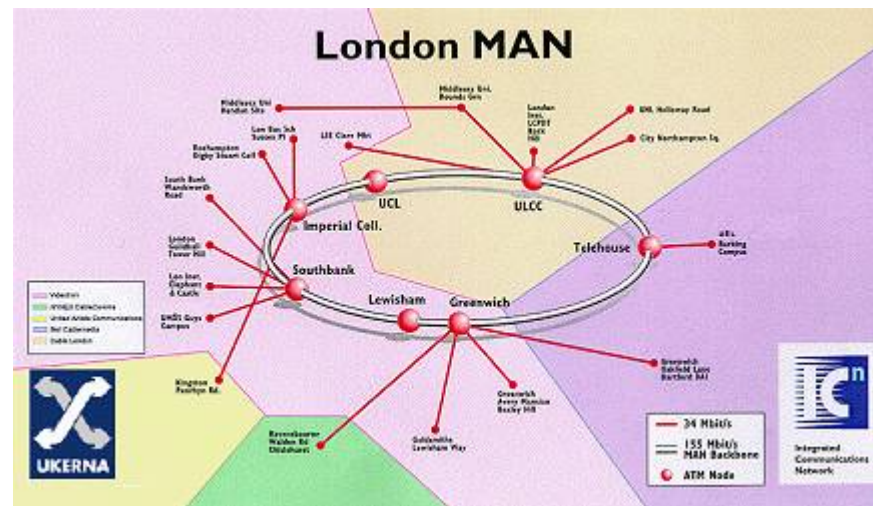
- Wide-area network
 - Typically inter-city to global
 - 100-20,000 km



MAN

typically up to a city

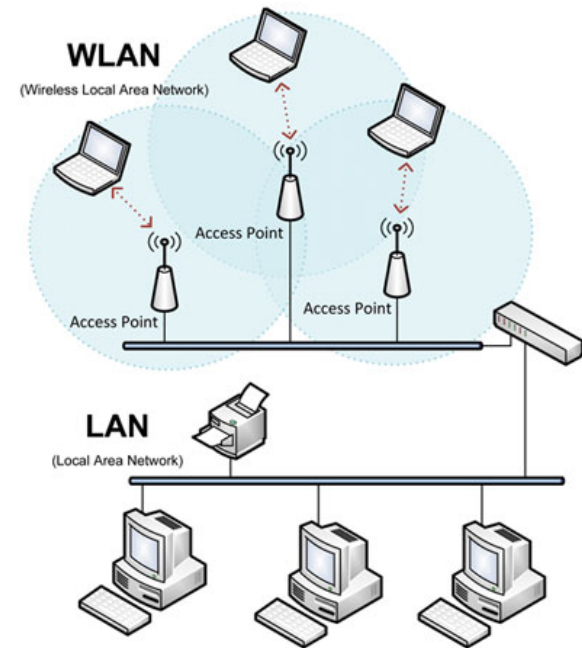
- Metropolitan ...
 - Several buildings to a city and its suburbs
 - 0.1-100 km



LAN

- Local area network
 - Inter-desktop to intra-building
 - The first *AN
 - 1-100 m
 - WLAN = wireless

wireless LAN



Other *ANs

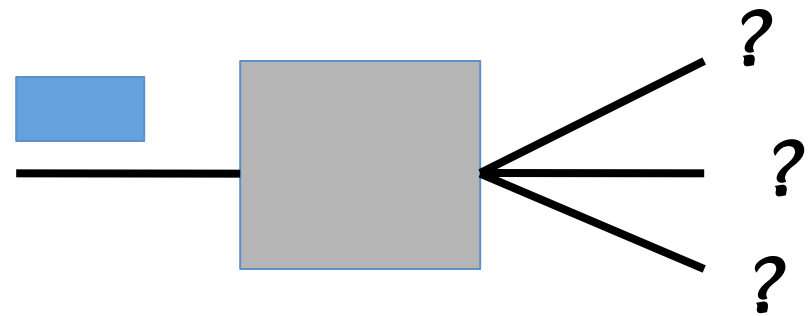
- DAN
 - Desktop
 - Interconnecting devices inside a PC
- PAN / BAN
 - Personal / body
 - Devices within one person's space
 - Phone, watch, tablet, sensors
- SAN
 - Storage
 - LAN or DAN focused on storage devices
- CAN
 - Car or campus
 - Car is maxi PAN
 - Campus is mini MAN
- HAN
 - Home
 - LAN with privacy limits

Networks without a “AN”

- Intranet
 - Network internal to a corporation
- Internet network of networks
 - Network of heterogeneous networks (“inter”)
- Access / Aggregation
 - Connects multiple LANs and WAN
- Interplanetary
 - Very large delays
 - Intermittent links

Relaying and choices

- Nodes in a topology must relay messages
- Except for simple topologies, choices must be made
 - Multiple outgoing links
 - Which to send it on?



Relaying rules

- What do you do when you get a message?
 - Deliver it if it's for you
 - Otherwise, relay it!
- How?
 - (for now, assume everyone knows a-priori)
- When?
- What about collisions?

The when question

- A message starts to come in
 - And it isn't for you
- *When* do you start sending it out?
if incoming message is not for you, when do you send it out?
- As soon as you know where to send it?
 - Possibly before you've gotten all of it
- Or only when the entire message has arrived?

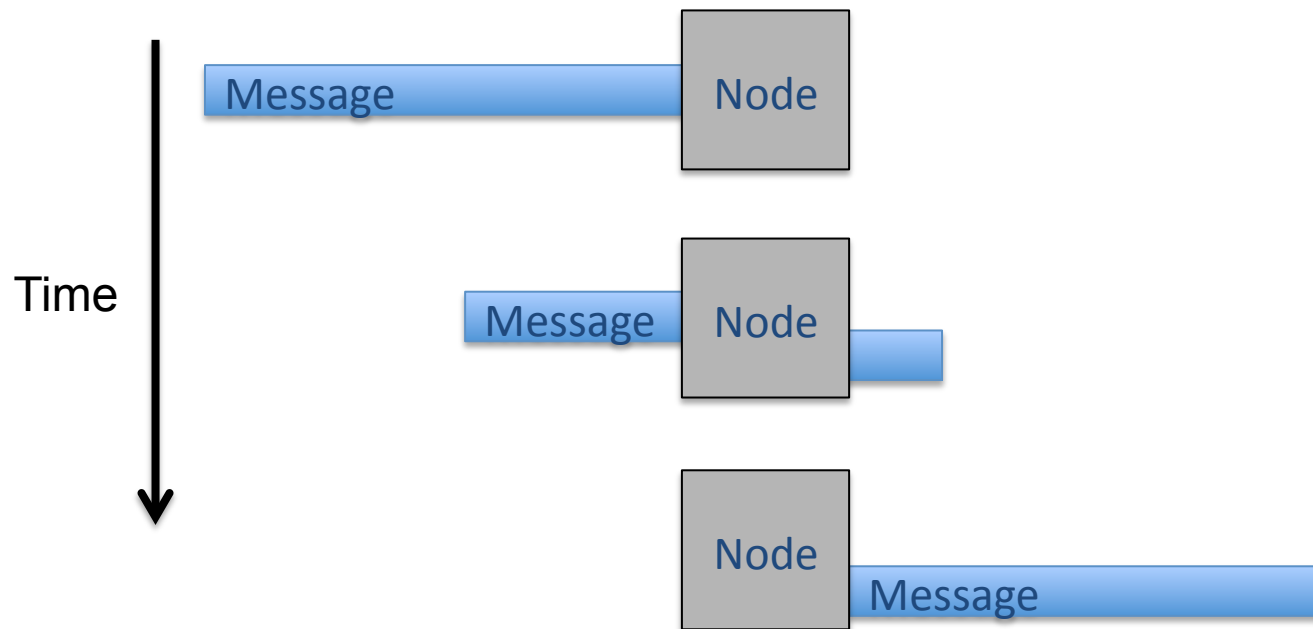
checksum can only be used when i have all parts of the packet

i should wait on the entire packet and wait until more info to dceck

Cut-through

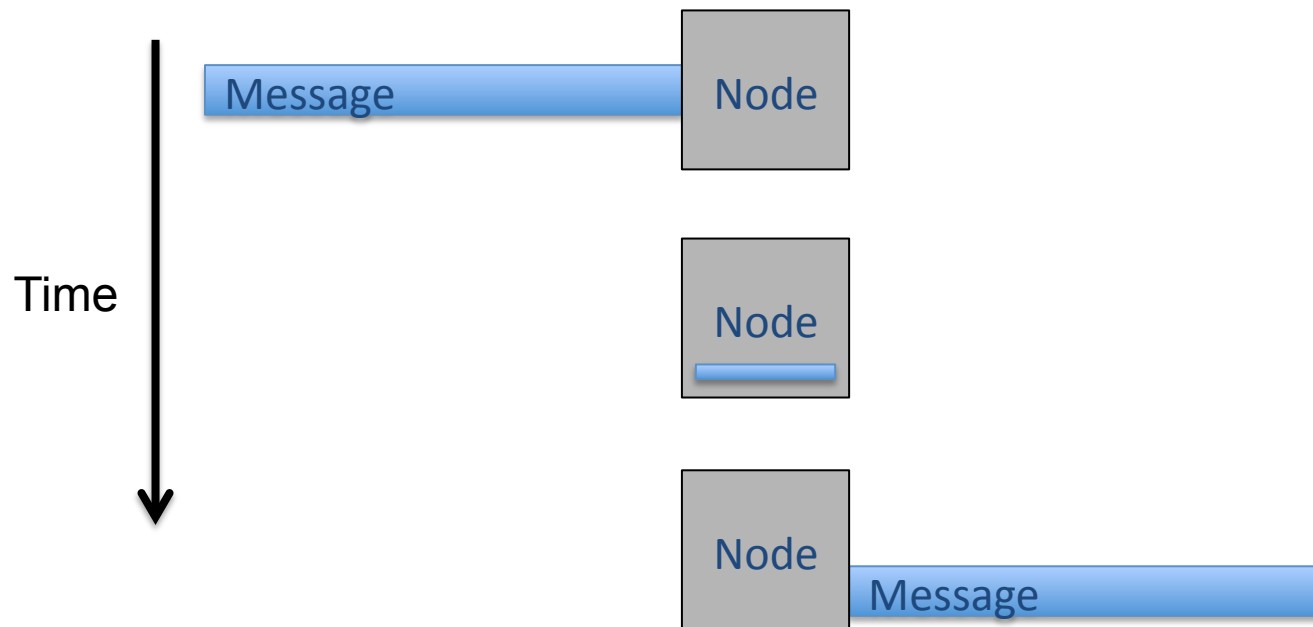
- Delay only long enough to decide

do a little bit of waiting - because first part of packet does not tell me where to go



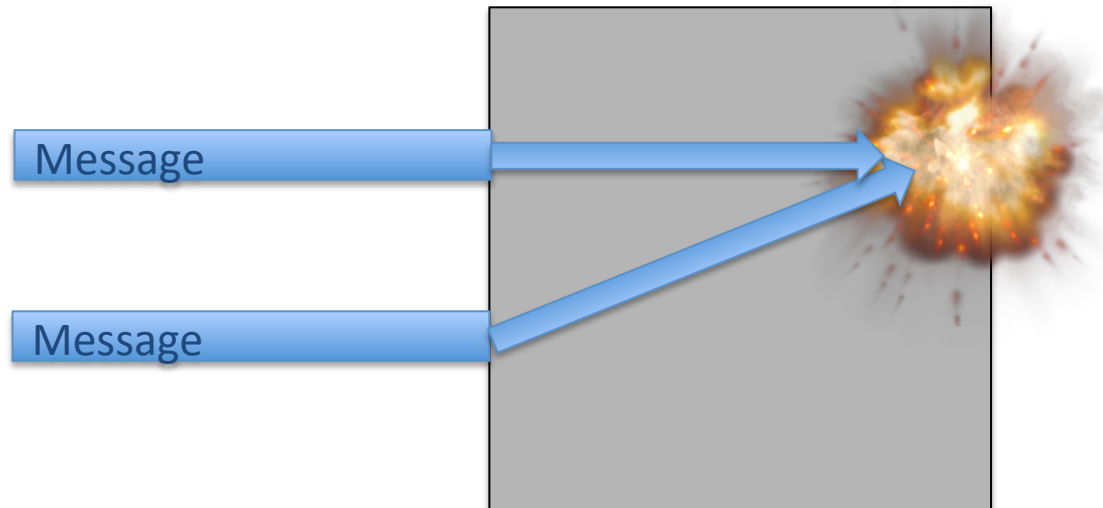
Store and forward

- Message “stops” inside the node



Collisions

- Multiple messages arrive at a node
 - More than one seeks the same output
 - (output) contention, collision

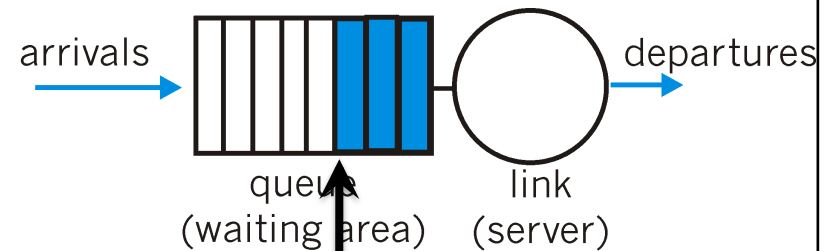


Avoiding collisions

- Delete one
 - Which one?

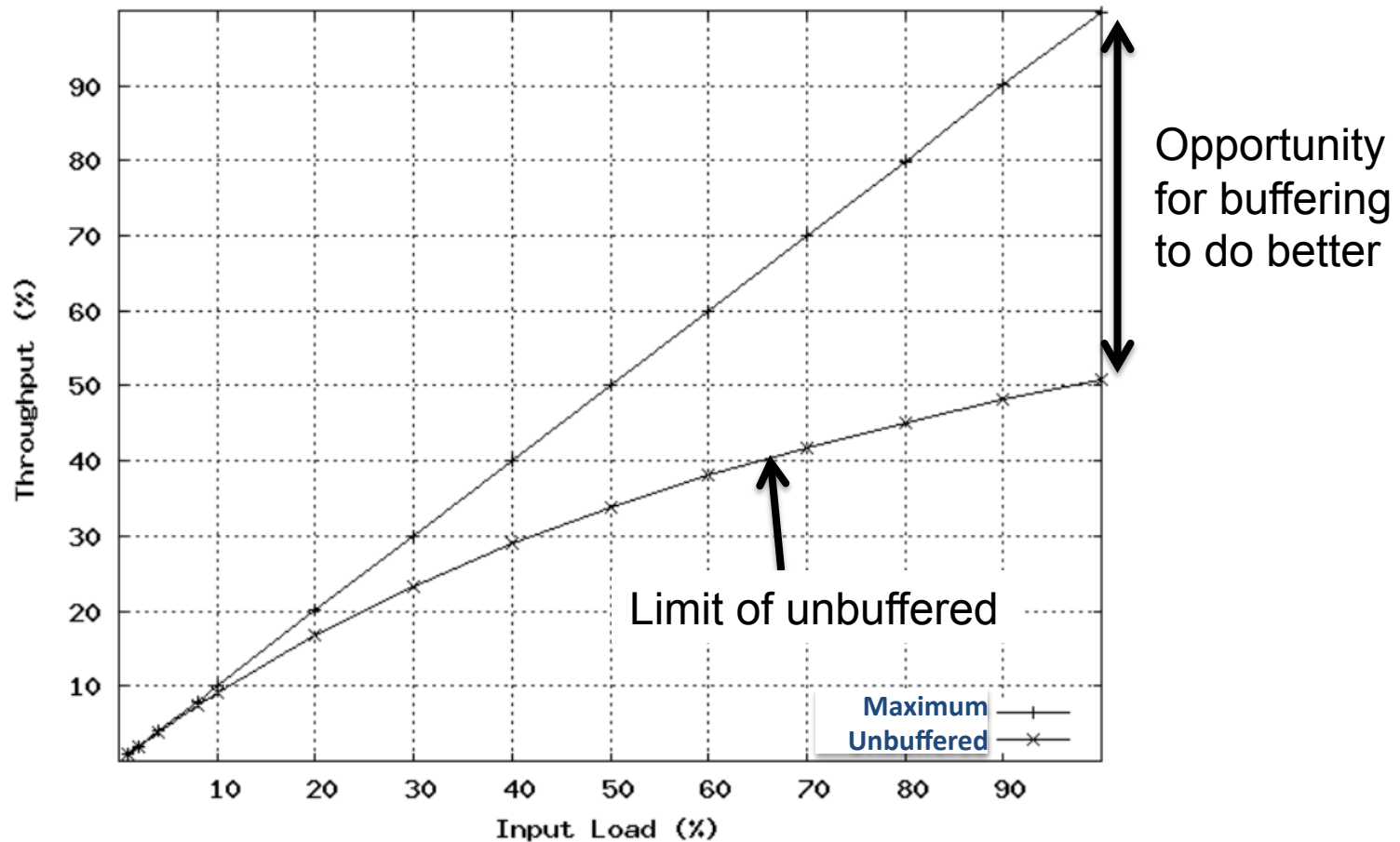


- Save one
 - Which one?
 - How long?



- Require storage
 - *A buffer*

Buffered vs. bufferless



Queuing

- Store and forward provides an opportunity
 - Messages can be managed as a set
 - Messages can be reordered
 - Messages can be dropped



- Critical issue – what do you do when buffer is full?

A short foray into queueing theory

- Queueing theory is mathematical theory of handling lines
- Highly applicable to networking
- In particular, to the issue of handling messages as they arrive at a node
- A fundamental question –
 - Under what circumstances will we need to drop a message?

Answering that question

- Informally, if you can't keep up with the work



- How can we formalize that idea?
- Through queueing theory

The basics of queueing theory

- We have a server
- It has an input queue
- The server takes work out of the input queue and sends it to an output
 - Taking some amount of time to do so
 - On average, W seconds to complete a piece of work
- Work arrives at some rate λ items per second
- When are we in trouble?

What's trouble?

trouble - things are not stable

- Trouble could be dropping any message
- Or it could be getting into a state where we'll never catch up
 - And messages will always need to be dropped
- Clearly, the latter is worse than the former
- If we occasionally drop a message, maybe we can recover
- If we always are dropping messages, that implies an unstable system

Little's result

- Under what conditions will our system be stable?
- Say our arrival rate at the network element in question is λ messages per second
- And we can forward a message in time W
- Both on average
- How many messages will be in the system, on average? $L = \lambda W$ --> if you can handle L messages, you are going to be stable
- $L = \lambda W$
- So if we can handle L messages, we'll be stable
 - BUT we might still drop some messages

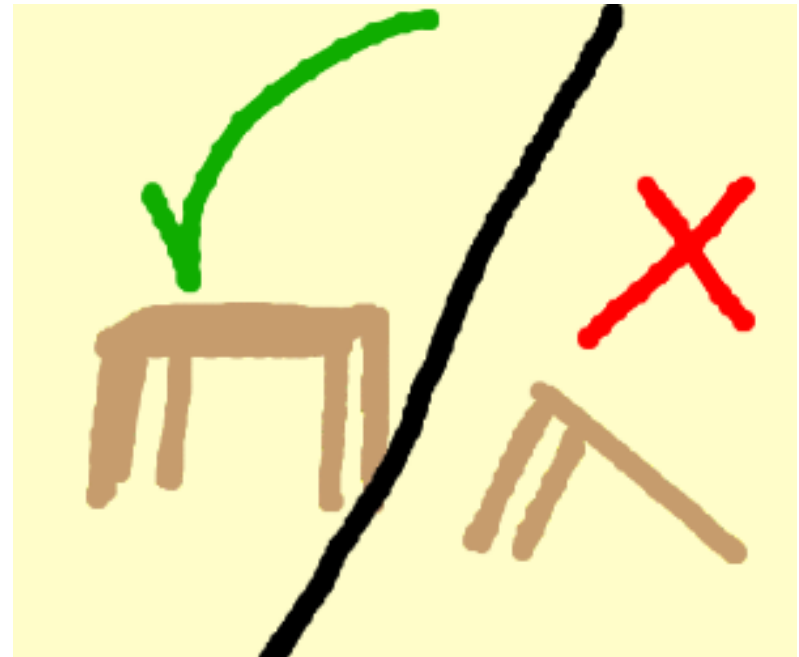
Drop behaviors

- Tail drop the N messages we already got deserve better treatments, so drop the most recent one
 - Delete arrivals to a full queue
- Head drop drop the ones that were sitting there for a very long time
 - New arrivals push oldest members out
- Random drop
 - Delete randomly from both new arrivals and existing members
- Random early drop
 - Random drop before the queue reaches its limit
 - Increase probability of drop as space decreases

RED - random early drop - let's not wait until we are full, let's use certain messages to do not

Priority

- Some messages are “more equal” than others
 - See George Orwell
Animal Farm
 - Messages are associated with priority
 - Priority biases decisions



QoS

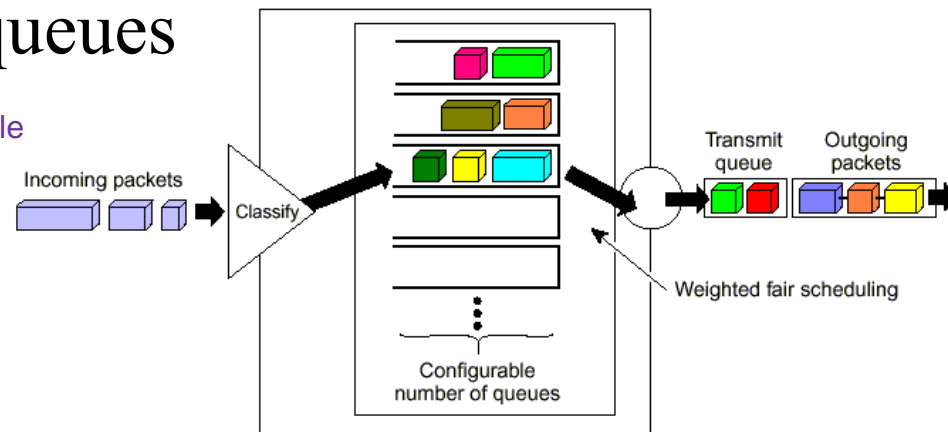
guarantees about messages

- Quality of service
 - Guarantees to a node about their messages
 - Per-message or long-term average behavior
 - Fixed guarantees or statistical
 - Complex set of queues

we dont have one queue, but multiple
people with higher quality of
service goes to higher queue

if you are blue collar ober

aniamls - gett 4 eil...



Returning to the question of choice

- Where do we relay the message to?
- Post office does it by looking at the address
- Implies a need for naming
 - Specifying who to deliver something to



know the packet name to see

Naming and relaying

- So a message arrives at a relay node
- It's not for that node
- How does the relay node know where to send it?
- A problem (partially) in naming
- The message can be associated with a name
- Which can allow the relay node to know how to relay it

Terminology

- Name space set of possible numbers or string
 - A set of possible names
 - Usually indicated by syntax or semantic range
 - E.g., N.N.N.N, where N=0..255
- Name
 - An instance within a name space
an instance within the name space
- Identifier naming is assigning a
teaee
ee
 - Another way to say “name”

Naming vs. identification

- Naming: for a particular item associated with name, how do you get it
 - The act of assigning an identifier to an item
- Identification:
 - The act of selecting a subset of items based on a name

What can be “named”?

- **Place** (source or sink of data)
 - Endpoint, node, process (inside an OS)
 - Location
- **Communications medium** (path of data)
 - Channel / link
 - Path (sequence of adjacent links)
- **Information** (the data itself)
 - Group of data, e.g., a record or file
- **A behavior** (the way data evolves)
 - A sequence of events, an algorithm, etc.

Finding names

- You have to know these things...
 - A-priori knowledge
 - A directory



Basically, some other way

Not associated directly with the message itself

Purpose of Names

- Support shared channels
- Support relaying
- Support layering
- Provide other information
- Support naming itself

Support shared channels

- Source identifier (N:1)
- Destination identifier (1:N)
- Group identifier (all, some, any)

Source vs. Destination ID

- Distinct
 - Source in the context of a destination
 - “Reza's professor named Peter”
 - Destination in the context of a source
 - “Peter's student named Nora”
 - Allows localized naming
 - Easier to ensure unique names
- One name for both
 - Avoids need for context
 - Requires coordination of all names

Types of group identifiers

- All
 - Broadcast
- Some
 - Multicast
- Any
 - Anycast

Broadcast

- All everywhere
 - Very rarely actually true
 - At best, true within part of a network
- All within a range of names
 - I.e., all names between Jenny and Morton
 - The metric is internal to the name space
 - Assumes an ordered name space
- All within some metric
 - Where the metric is external to the names
 - E.g., within 3 hops, within 2 ms, within 600m

Multicast

- A group of selected members
 - Indicated by the source
 - Indicated by an external membership protocol
- A subset of such a group (N of M)
 - Exactly N of M
 - At least N of M

Anycast

- Any member
 - Any available party responds
 - More than one can “offer”; originator selects
- Any ONE member
 - At most one party responds
 - The network (or the receivers) select
 - Originator is given no choice

Support relaying

- Structure in the name can help
 - Name space is a hierarchy that matches the network topology
 - E.g., IPv6
 - E.g., telephone country codes, area codes, exchanges
 - Name space can match geographic areas
 - Where you are determines part of your name
 - E.g., DNS country code suffixes (.us, .jp, .in)
 - Zip codes on letters
 - E.g., telephone country codes, area codes

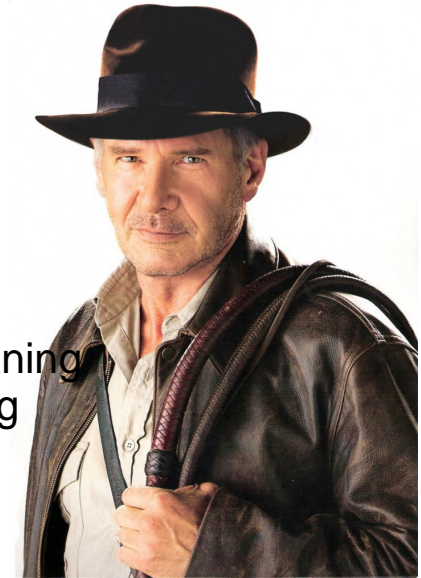
Types of Names

- Flat
 - I.e., unstructured
- Structured
 - Syntax
 - Semantics
- Single-level
 - One organization for the entire name space
- Multi-level
 - A sequence of organizations
 - Organizations can repeat or vary
 - Levels can represent a hierarchy (subdivision)

People

- Size
 - 1-N Characters (var. charset)
- Organization
 - 2-3 level hierarchy of flat
 - Right, left, middle in US, most of EU
 - Left, right in Japan
- Content
 - Surname (family)
 - Usually flat
 - Can change (marriage)
 - Given name
 - Flat
 - Nickname
 - Flat
- Broad/multicast support?
 - Geographically limited (hey you all!); geo-limited multicast (all the brown-eyed people)
- Example
 - Henry Jones, Jr.

structure: there is special meaning
flat: there's no special meaning



Houses

- Size
 - 1-N Characters (var. charset)
- Organization
 - 4-6 level hierarchy of flat
 - Country, region, city, street, unit, subunit
- Content
 - Flat in general
 - Units
 - Geographic in US
 - Chronological in Japan
- Broad/multicast support?
 - No broadcast; no multicast
- Example
 - 123 Main Street, Apt. 33, Boston, MA, USA



Geographic

- Size
 - 3 groups: Latitude, longitude, altitude
- Organization
 - 2-level independent
- Content
 - Latitude and longitude
 - Degrees, minutes, seconds
 - Or degrees and fractions
 - Altitude
 - Meters
- Broad/multicast support?
 - Geographic broadcast; no multicast
- Example
 - $34^{\circ} 1' 16.86''$, $-118^{\circ} 17' 27.5352''$ (elevation unknown)

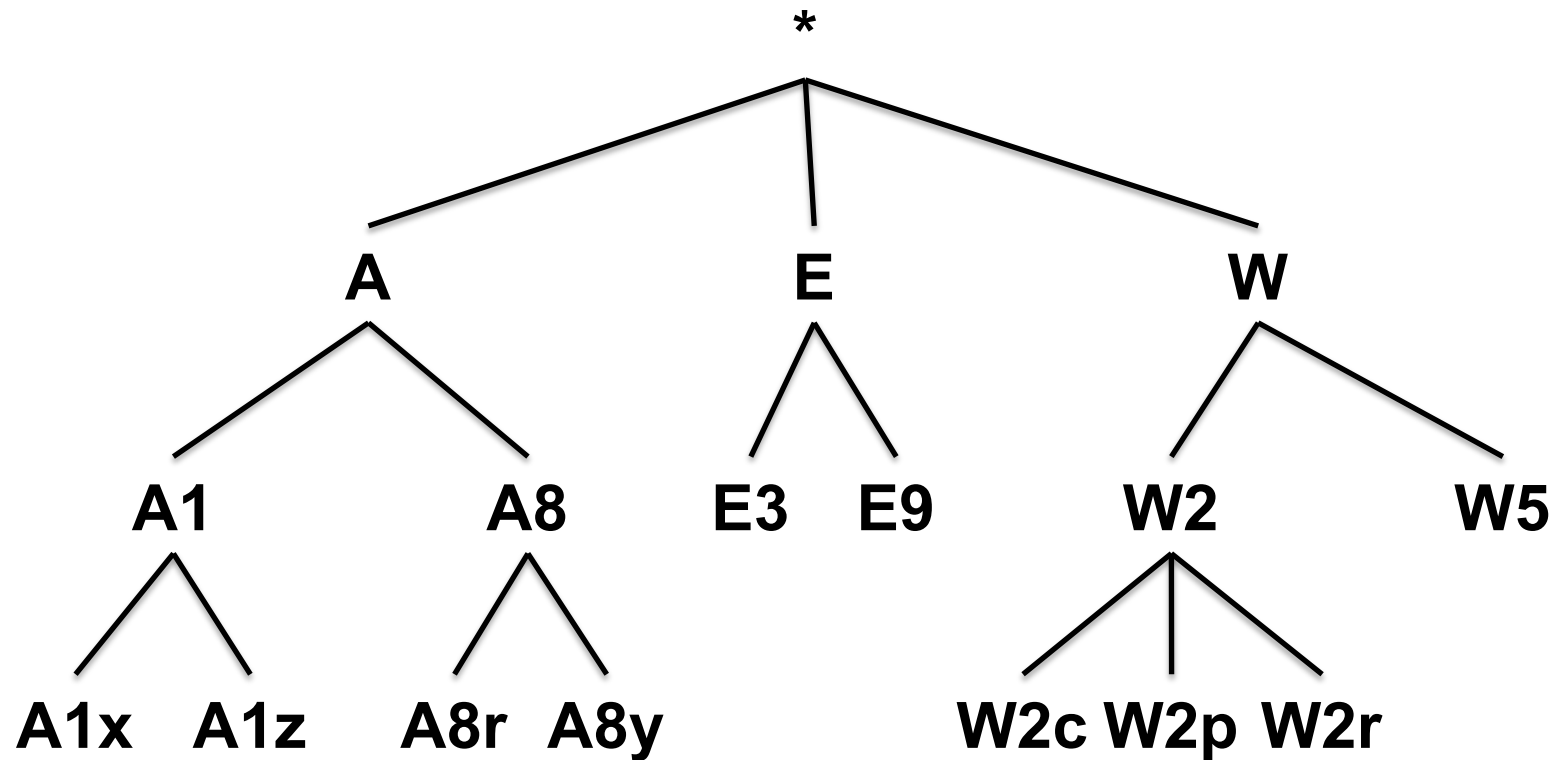


Geographic names



DNS created a portion of namespace for these kinds of name

Topological names



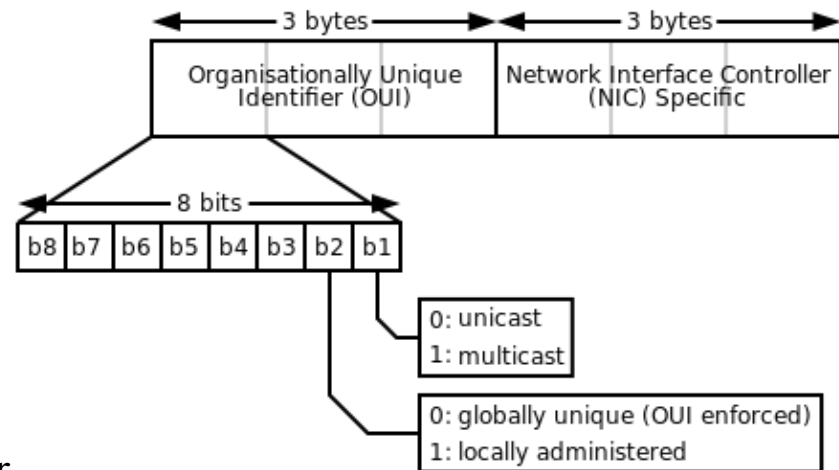
Telephone

- Size
 - 6-N digits
- Organization
 - 3-4 level hierarchy of flat, variable
- Content
 - Flat variable country code
 - Flat variable region code
 - Flat variable exchange code
 - Flat variable line number
- Broad/multicast support?
 - No broadcast; no multicast
- Example
 - +49-89-636-48018



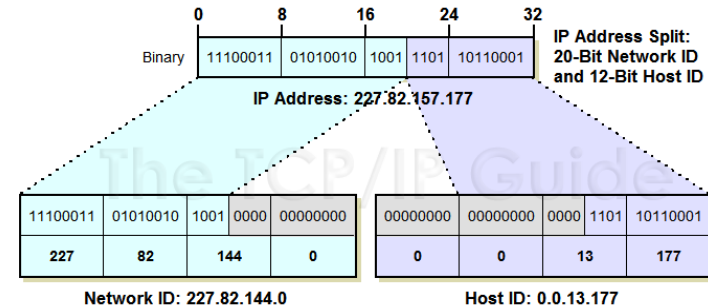
Ethernet

- Size
 - 6 bytes
- Organization
 - 2-level hierarchy of flat, fixed
- Content
 - Flat 3-byte OUI
 - Organizationally Unique Identifier
 - E.g., F0-F6-1C = Apple (one of many!)
 - Flat 3-byte NIC
 - Network Interface Controller
- Broad/multicast support?
 - Global broadcast; global multicast as configured
- Example
 - F0-F6-1C-24-E3-15



IPv4

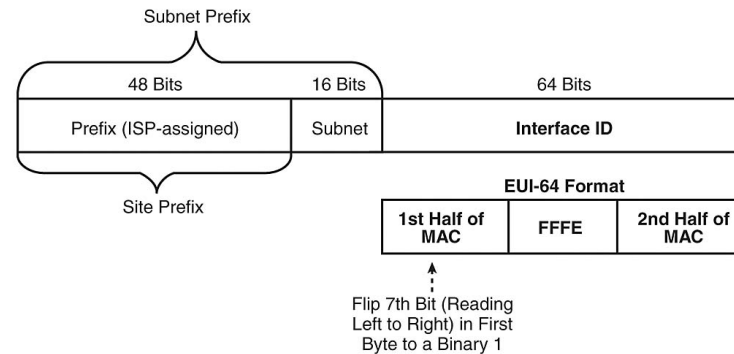
- Size
 - 4 bytes
- Organization
 - 2-level hierarchy of flat, variable
- Content
 - Flat high-order network part
 - Assigned by IANA
 - Of variable length
 - Flat low-order host part
 - Assigned locally, either statically or via automation
 - Of variable length
- Broad/multicast support?
 - One-hop broadcast; global multicast as configured
- Example
 - 128.9.160.161



IPv6

64 bit addresses

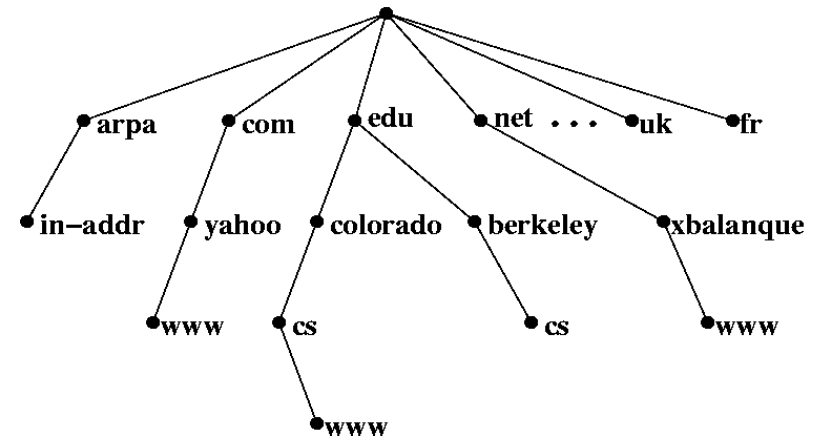
- Size
 - 8 bytes
- Organization
 - 3-level hierarchy of flat, fixed
- Content
 - Flat, multi-level 6-byte site prefix
 - Assigned by IANA, variable
 - Assigned to ISPs hierarchically
 - Flat 2-byte subnet
 - Assigned locally, either statically or via automation
 - Flat 8-byte interface ID
 - Derived from Ethernet MAC address
- Broad/multicast support?
 - One-hop broadcast; multicast to preassigned groups or as configured
- Example
 - 2001:0:9d38:6ab8:3c1f:108d:b898:6b35



Host names (hosts.txt)

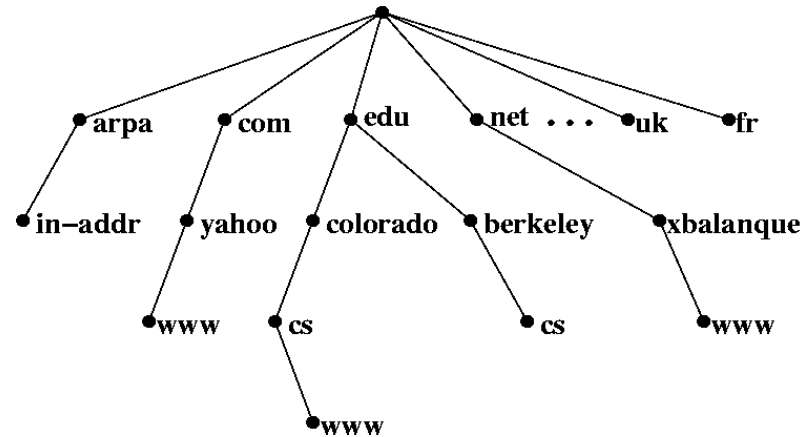
- Size
 - 1-N characters
- Organization
 - Variable-level sequence of flat, variable
- Content
 - Labels separated by “.”
 - Organized right to left
 - Assigned and managed *globally*
- Broad/multicast support?
 - No broadcast; no multicast
- Example
 - [lever](#).cs.ucla.edu
 - Each level can have different number of characters

translation between IP address and host name



DNS

- Size
 - 1-63 byte labels, 1-253 bytes total
- Organization
 - Variable-level hierarchy of flat, variable
- Content
 - Labels separated by “.”
 - Delegated right to left
 - Assigned and managed locally
- Broad/multicast support?
 - No broadcast; no multicast
- Example
 - [lever](#).cs.ucla.edu
 - Again, possibly different number of characters at each level



TCP, UDP, SCTP, etc.

like TCP, less reliable

TCP: transport layer

- Size
 - 2 4 byte labels, plus IPv4 or IPv6

- Organization

- Three groups of flat

- Content

- System (registered)
 - 0-1023, “privileged”

- User (registered)

- 1024-49151

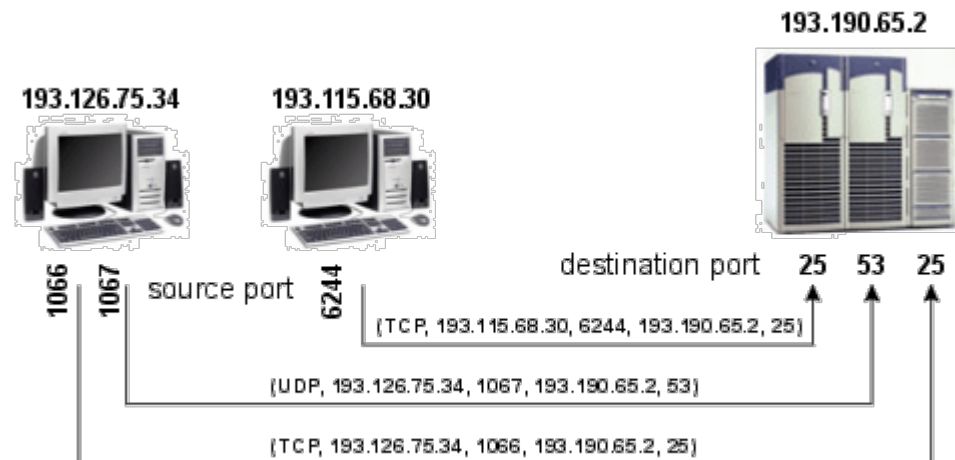
- Dynamic (self-assigned)
 - 49152-65535

- Broad/multicast support?

- No broadcast; no multicast

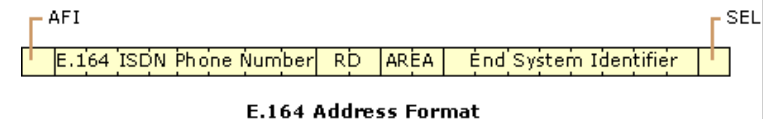
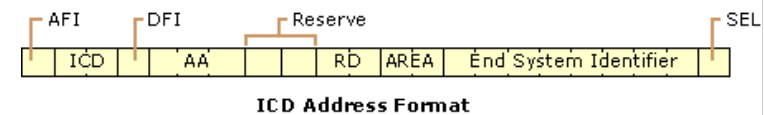
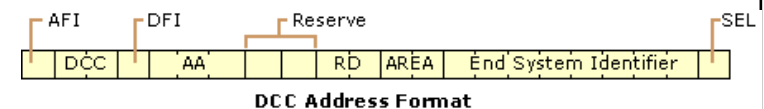
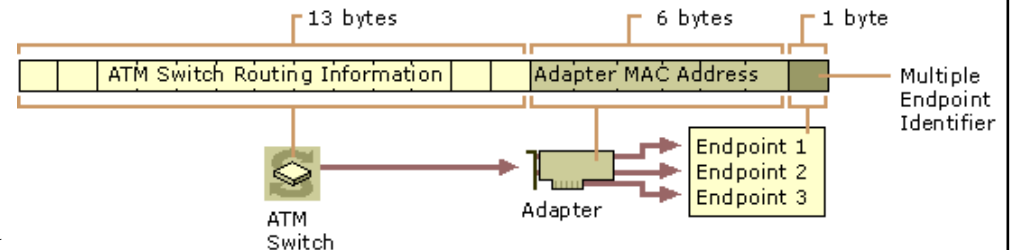
- Example

- 80 for HTTP, 53 for DNS



ATM

- Size
 - 20 bytes
- Organization
 - 3-level hierarchy of flat, fixed
- Content
 - 13-byte switch part
 - Three hierarchies
 - 6-byte interface part
 - Borrows Ethernet MAC
 - 1-byte selector part
 - Multiple endpoints within an interface
- Broad/multicast support?
 - No broadcast (emulation); no multicast
- Example
 - 47.00918100000001604799FD01.0050A219F03B.0



Names also provide other information

- Names do more than identify
 - Cell area code – where/when you first got it
 - Social security – where/when you were born
 - People – lineage (typ. paternal)
 - Ethernet – who made your device and when
 - IPv6 – who your ISP is

Summary

- We can build more scalable networks by connecting nodes and relaying messages
- Nodes are connected in some topology
 - Different topologies have different characteristics
- Relaying needs more shared rules
 - And often also a place to wait
- Naming is one way to enable relaying