

Distributed Systems

Communication and models

Rik Sarkar
Spring 2018

University of Edinburgh

Models

- Expectations/assumptions about things
- Every idea or action anywhere is based on a model
- Determines what can or cannot happen

Communication & modeling

- Modeling distributed systems:
 - How we can think about them
- Communication between nearby nodes
- Communication between distant nodes
- Communication with many nodes

Some terminology:

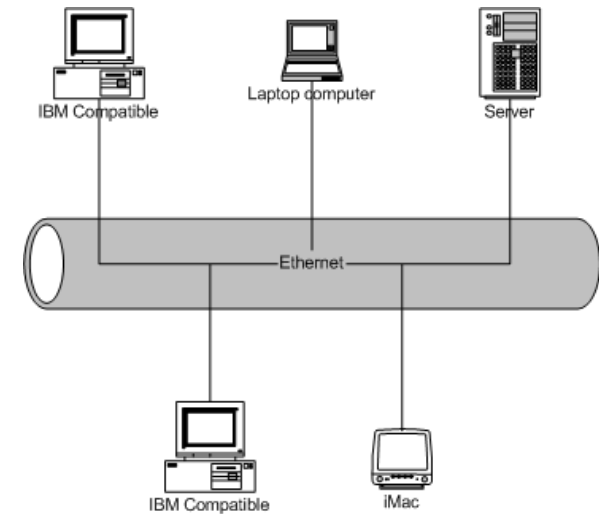
- One to “all”: Broadcast
 - All: in some set of interest
- One to one: point to point

Packets

- Networks communicate data in messages of fixed (bounded) size – called packets
- More data requires more packets
- Number of messages or packets transmitted is a measure of communication used

Local area networks

- **Medium: Broadcast**
 - Message goes from one computer to *all* other computers (restricted to some set)
 - For example, all other computers in the LAN, or some other system in consideration
 - Ethernet LAN is a broadcast medium
 - All computers are connected to a wire. They transmit messages on the wire and all can receive
 - Wireless LAN (WiFi) is a broadcast medium
 - Electromagnetic waves is the common medium



Local area networks

Advantages:

- Sending a common message to everyone is easy
- Finding destination is easy
 - Message goes to everyone
 - Just have a “destination” field

Main issue: Medium access

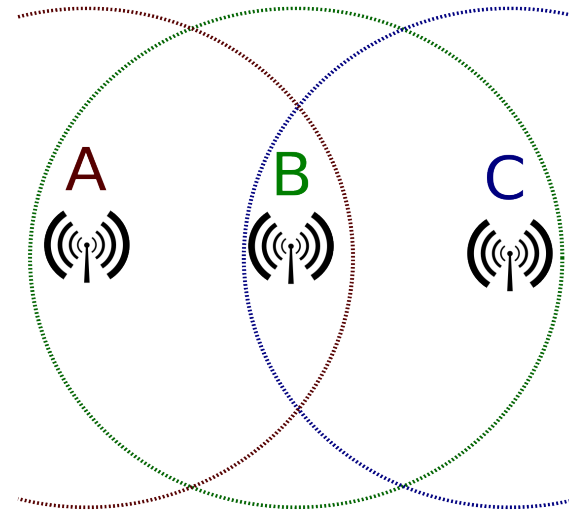
- Since medium is broadcast, two people transmitting at the same time garbles message

Medium access

- Only one transmission at a time can be allowed
 - Mutual exclusion problem (shared resource of communication wire)
 - We cannot use messages to solve it
 - Protocols:
 - TDMA: Everyone has a periodic slot
 - CSMA: See if anyone else is transmitting. If so, defer.
 - Usually acks are also used to ensure transmission
 - Retransmit if necessary
 - Bandwidth reduces with number of nodes trying to transmit.
 - One LAN should not have too many nodes

Medium access

- Wireless: more complicated
- Hidden terminal problem
- More complex protocol using acks

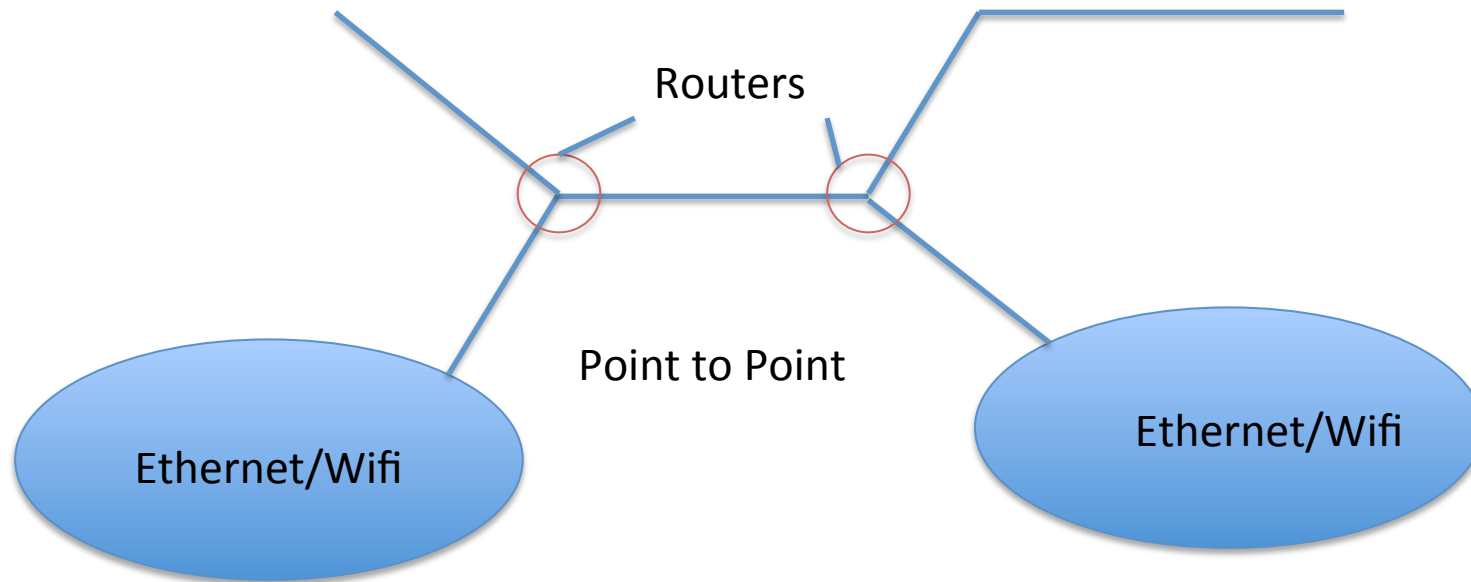


Our models of LAN

- Graph: Every node has an edge to every other
- We often assume that to send a message (packet) to a node on the same network takes one unit of time (or, at most a constant)
- This may not be true if there can be many nodes in the same LAN
 - But usually the number is not very large

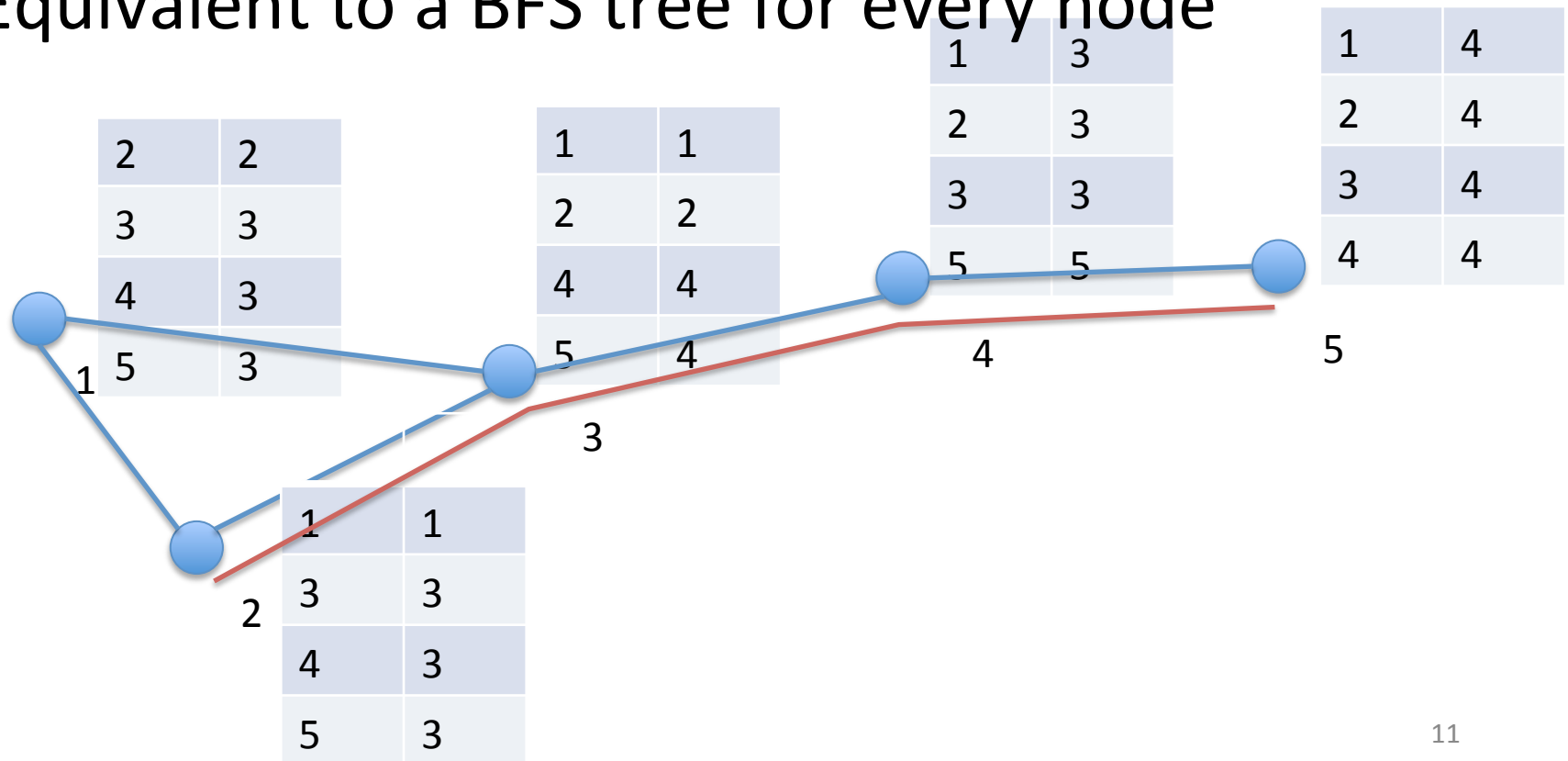
Real life networks

- LANs connected by routers



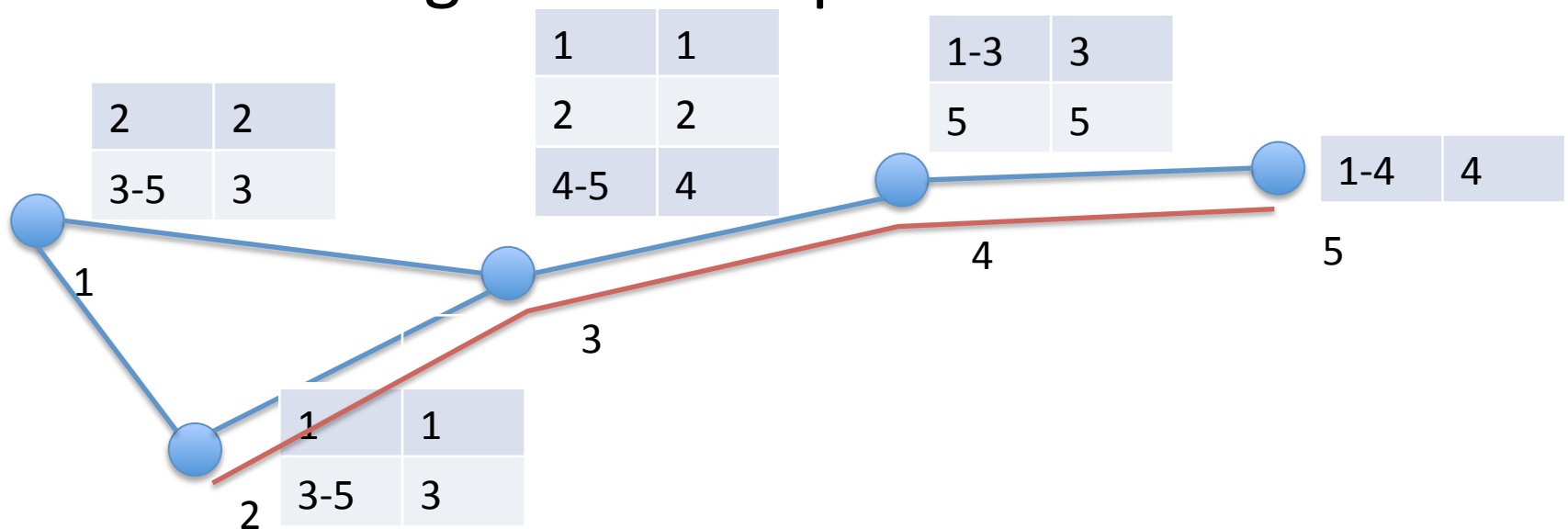
Routing

- Finding a path in the network
- Every node has a routing table
- Equivalent to a BFS tree for every node



Routing: Distributed search for a path

- Smaller routing tables by combining addresses
- Used in IP (Internet) routing
- Smaller routing tables are preferable

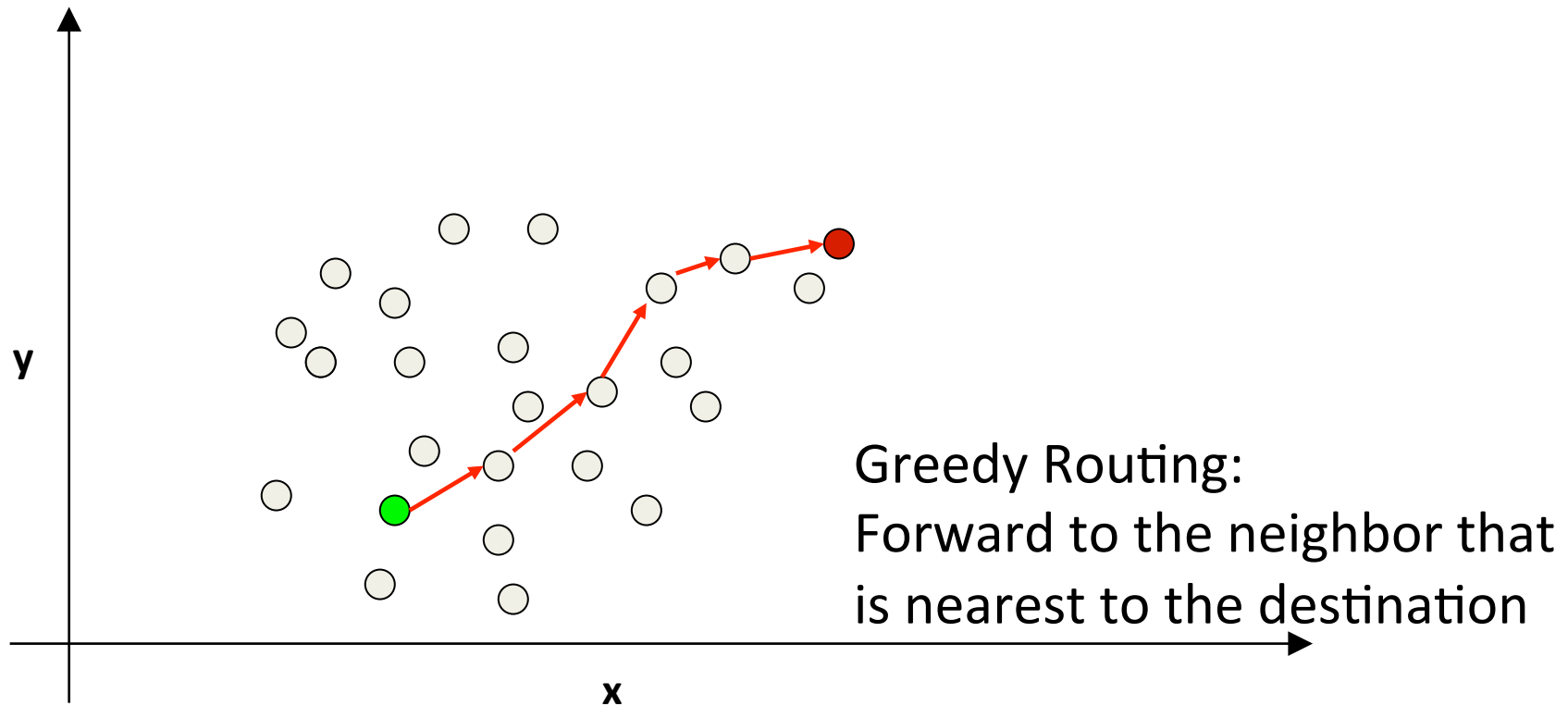


Routing

- Real routing is more complicated
- With more than one path to a destination, backups etc

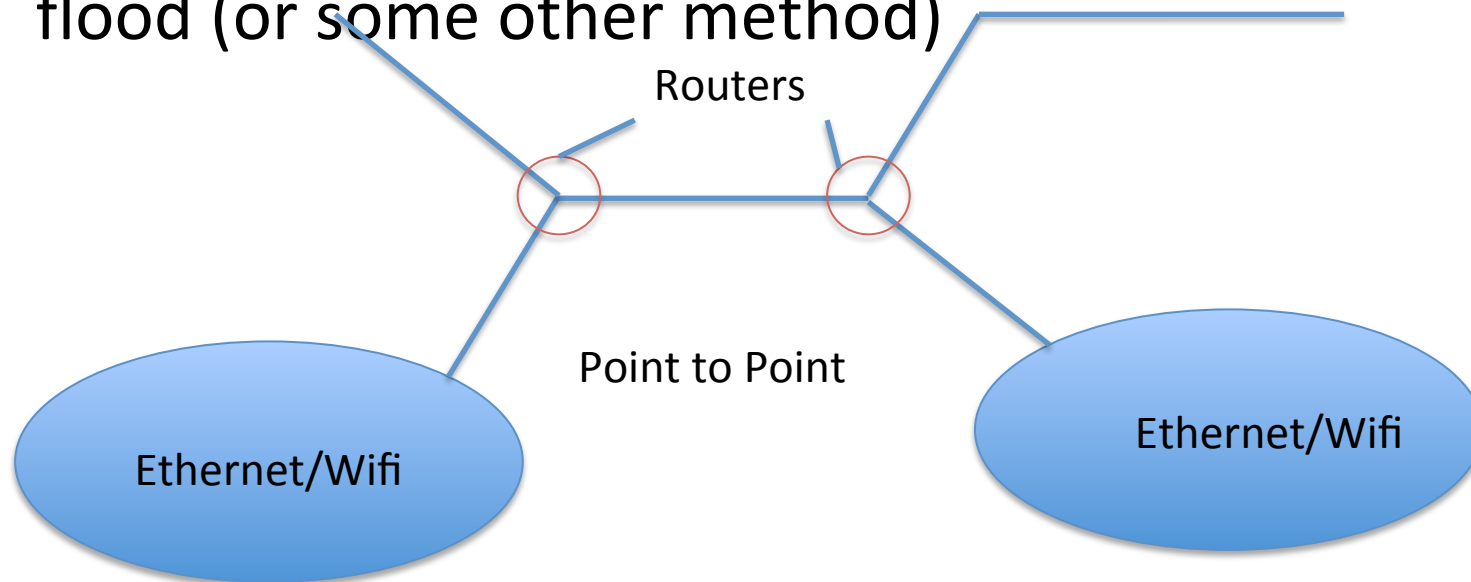
Related: Location based routing

- Geographical routing uses a node's **location** to discover path to that node.



Large networks

- Communication is typically point-to-point using routing
- Broadcast is not automatic
 - If we need broadcast, we will have to arrange a flood (or some other method)



Transport management

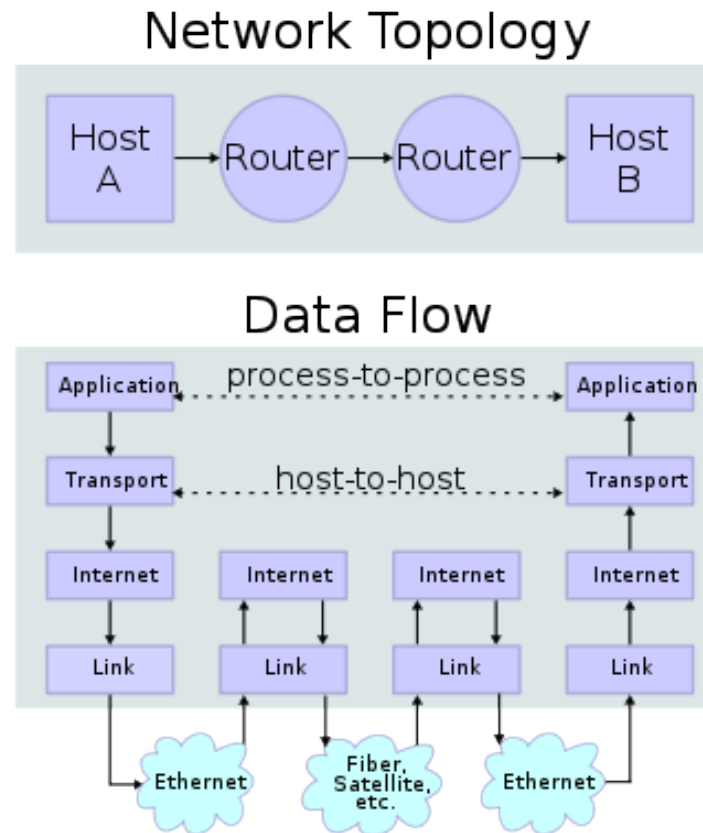
- UDP:
 - Send a packet, hope that the network routes and delivers it, in time
 - No Sequence number
 - Not necessarily FIFO
 - Useful in streaming audio/video. Not for important data.
- TCP:
 - Send a packet (or few packets)
 - Packets have sequence number
 - FIFO
 - If no acks arrive, resend packets
 - If no acks are found after many tries, return error

TCP

- Does distributed congestion control
 - When packets don't get delivered, TCP slows down the stream
 - Assumption: routers drop packets when there are too many
- Difficulty
 - Acks may not arrive due to other factors
 - Some connection failed temporarily
 - User moved from one network to another

Network stack

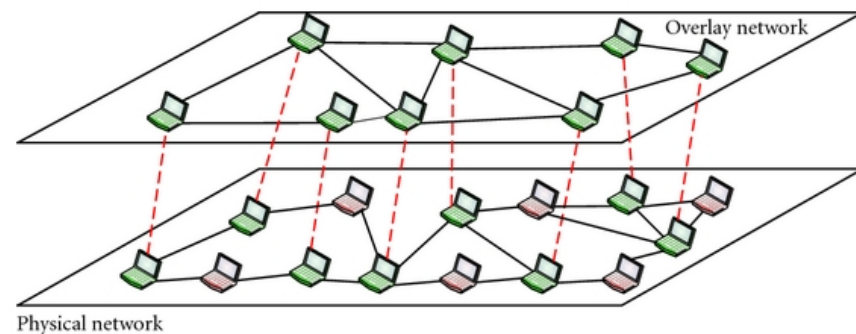
- Each layer solves a different distributed problem



Form wikipedia

Communication: Overlay network

- We may sometimes ignore parts of the network
 - Nodes that carry messages but do not directly participate eg. routers
 - Or edges that exist but we are not using
 - Or we don't know about
- Often used in peer-to-peer networks
 - Not every node knows all other nodes in the network
 - But communicates to known nodes through routing

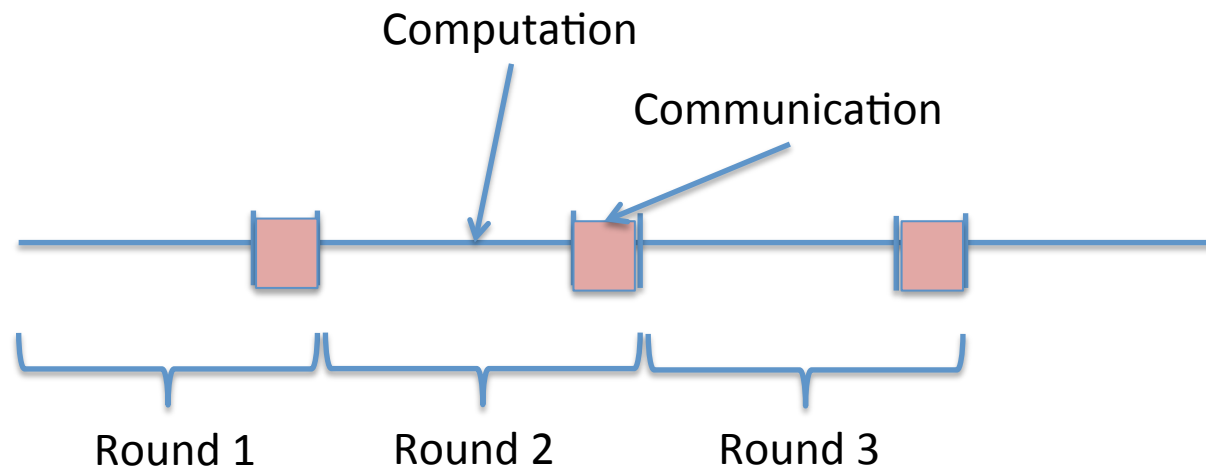


Blocking and non-blocking communication

- Blocking communication
 - Sender sends message, waits until receiver replies
 - Does not do anything in the mean time
- Non – blocking communication
 - Sender sends message, then continues its own work without waiting
 - When receiver replies, or some other message arrives, node interrupts current work to handle message
- Sometimes these are called synchronous/asynchronous, but we will try not to use that

Computation

- Synchronous:
 - Operation in rounds
 - In a round, a node performs some computation, and then sends some messages
 - All messages sent at the end of round x are available to recipients at start of round $x+1$
 - But not earlier



Communication

- Synchronous
 - Can be implemented if message transmission time is bounded by some constant say m
 - Computation times for all nodes are bounded by some constant c
 - Clocks are synchronized (sufficiently)
 - Then set each round to be $m+c$ in duration

Asynchronous Communication

- No synchronization or rounds
 - Nodes compute at different and arbitrary speeds
 - Messages proceed at different speeds: may be arbitrarily delayed, may be received at any time
- Worst case model
 - No assumption about speeds of processes or channel
 - (But does not include communication/computation errors)

Asynchronous Communication

- Harder to manage
 - Message can arrive at any time after being sent, must be handled suitably
 - Possible to make some simplifying assumptions
 - E.g.:
 - Channels are FIFO: order of messages on a channel are preserved
 - Some code blocks are atomic (not interrupted by messages)
 - Either communication or computation times bounded

Synchronous communication in Real systems

- Synchronous communication can be a fair model
- Modern computers and networks are fast
 - (though not arbitrarily fast)
- Easier to design algorithms and analyze
- Well designed algorithms are faster and more efficient
- Often can be adapted to asynchronous systems
 - Often a starting point for design

Failures

- Nodes may fail
 - Hardware failure
 - Run out of energy or power failure
 - Software failure (crash)
 - Permanent
 - Temporary (what happens when it restarts?
Recovers the state? Starts from initial state?)
 - Model depends on system. E.g. different types of failures occur with corresponding probabilities

Node failures

- Common abstract models
 - Stopping failure: node just stops working
 - May need assumptions about which computation/communication it finishes before stopping
 - May need assumption about neighbors knowing of failure
 - Byzantine failure: node behaves as an adversary
 - Imagine your enemy has taken control of the node
 - Is trying to spoil your computation
- Nodes may fail individually
 - E.g. each node fails with probability p
- Nodes may have correlated failure
 - E.g. all nodes fail in a region (data center, sensor field)

Link/communication failure

- May be temporary/permanent
- May happen due to
 - Hardware failure
 - Noise: electronic devices (microwaves etc) may transmit radio waves at similar frequencies and disrupt communication
 - Interference: Other communicating nodes nearby may disrupt communication
- Effects
 - Channel silent and unusable (hardware failure)
 - Channel active, but unusable due to noise and interference
 - Channel active, but may contain erroneous message (may be detected by error correcting codes)

Security

- Issues:
 - Unauthorized access, modification. Making systems unavailable (DOS)
 - Attack on one or more nodes
 - Causing to it fail
 - Read data
 - Taking control to read future data, disrupt operation
 - Attack on communication links/channel
 - Block communication
 - Read data in the channel (easy in wireless without encryption)
 - Corrupt data in the channel

Security

- Solutions usually have specific assumptions of what the adversary can do
- E.g. If adversary has access to channel
 - Cryptography may be able to prevent reading/corrupting data

Mobility

- Movement makes it harder to design distributed systems
 - Communication is difficult
 - Delays, lost messages
 - Edge weights can change
 - Applications that depend on location must adapt to movement
- How do people move? What is a model of movement?
 - Not yet well understood

Modeling distributed systems

- Many possibilities
- Choose your assumptions carefully for your problem
- Pay close attention to what is known about communication/network
- Start with simpler models
 - Usually more assumptions, fewer parameters
 - See what can be achieved
 - Then try to drop/relax assumptions

