CS 268: Lecture 5 (Project Suggestions)

Ion Stoica

February 6, 2002

Project Related with Internet Indirection Infrastructure (*i*3)

- Goal: provide an uniform abstraction for basic communication primitives:
  - Unicast
  - Multicast
  - Anycast
- Next: overview of *i*3

Motivations

- Today’s Internet is built around a point-to-point communication abstraction:
  - Send packet “p” from host “A” to host “B”
- This abstraction allows Internet to be highly scalable and efficient, but…
- … not appropriate for applications that require other communication abstractions:
  - Multicast
  - Anycast
  - Mobility
  - …

Why?

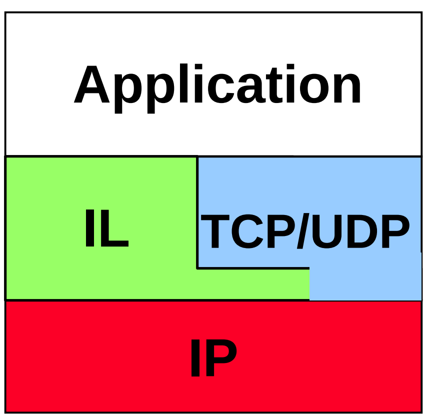
- Point-to-point communication abstraction implicitly assumes that there is one sender and one receiver, and that they are placed at fixed and well-known locations
    - E.g., a host identified by the IP address 128.32.xxx.xxx is most likely located in the Berkeley area

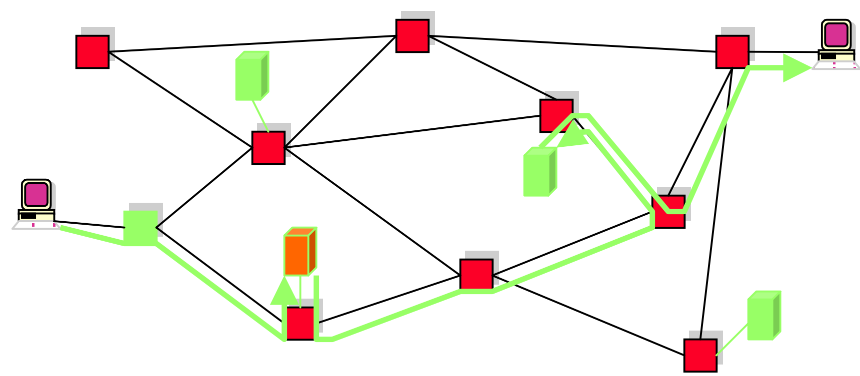
Key Observation

- All previous solutions use a simple but powerful technique: indirection
  - Assume a logical or physical indirection point interposed between sender(s) and receiver(s)
- Examples:
  - IP multicast assumes a logical indirection point: the IP multicast address
  - Mobile IP assumes a physical indirection point: the home agent

Our Solution

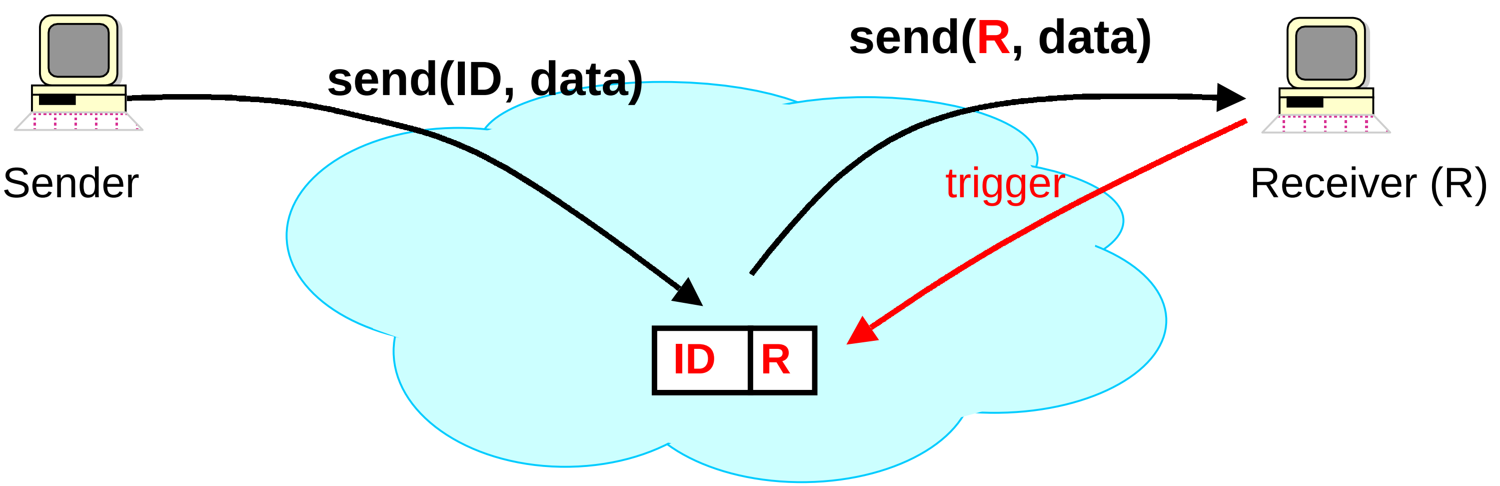
- Add an efficient indirection layer (IL) on top of IP
  - Transparent for legacy applications
- Use an overlay network to implement IL
  - Incrementally deployable; don't need to change IP





Internet Indirection Infrastructure

- Change communication abstraction: instead of point-to-point, exchange data by name
  - Each packet is associated an identifier ID
  - To receive a packet with identifier ID, receiver R maintains a trigger (ID, R) into the overlay network

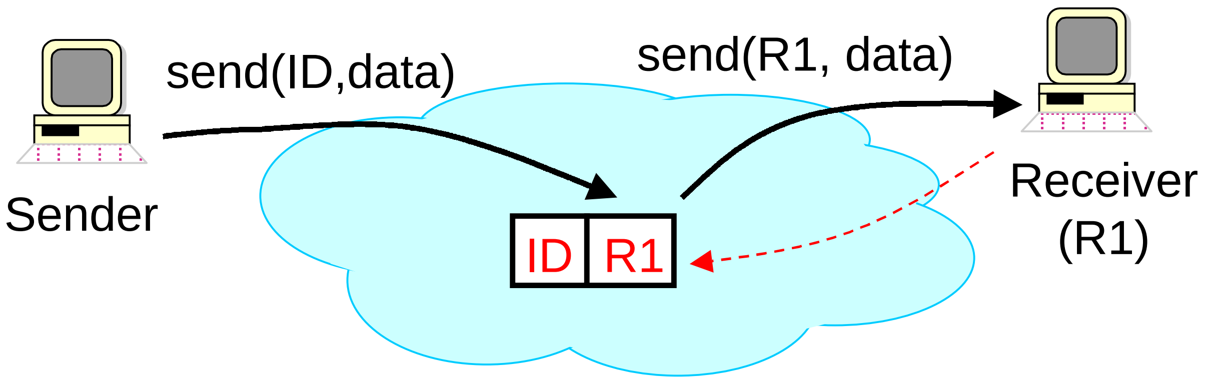


Service Model

- Best-effort service model (like IP)
- Triggers are periodically refreshed by end-hosts
- Reliability, congestion control, and flow-control implemented at end-hosts

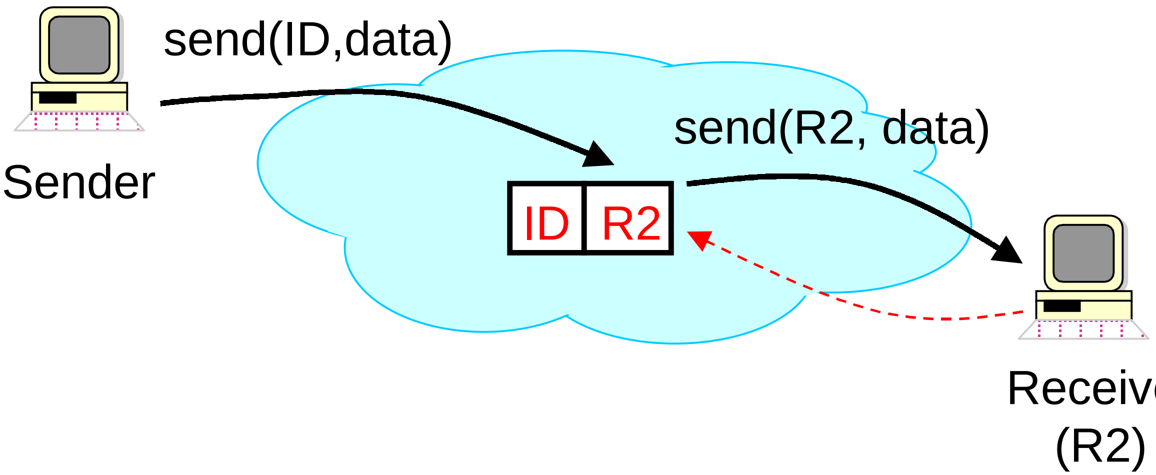
Mobility

- Host just needs to update its trigger as moves from one subnet to another
- Both sender and receiver can be mobile
- Can eliminate the “triangle routing problem”



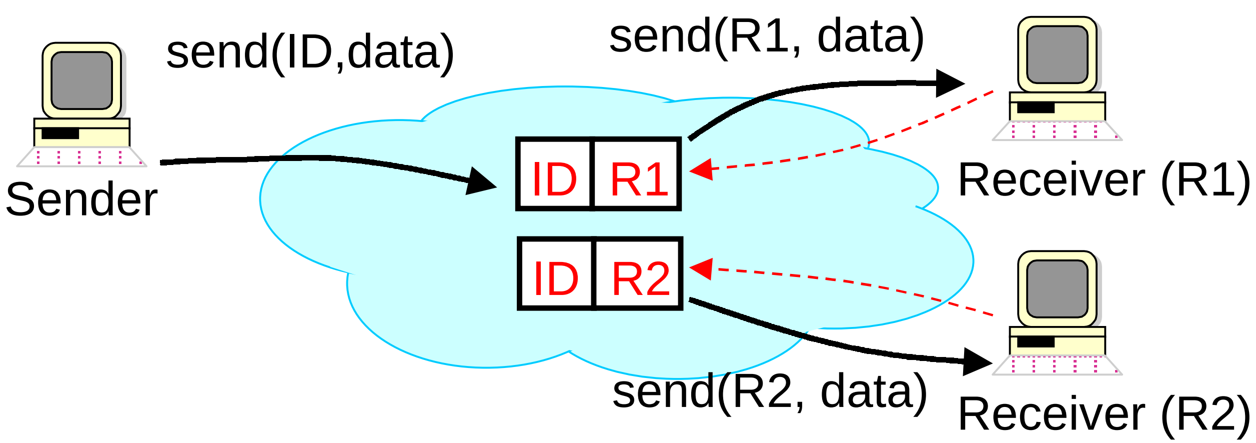
Mobility

- Host just needs to update its trigger as moves from one subnet to another
- Both sender and receiver can be mobile
- Can eliminate the "triangle routing problem"



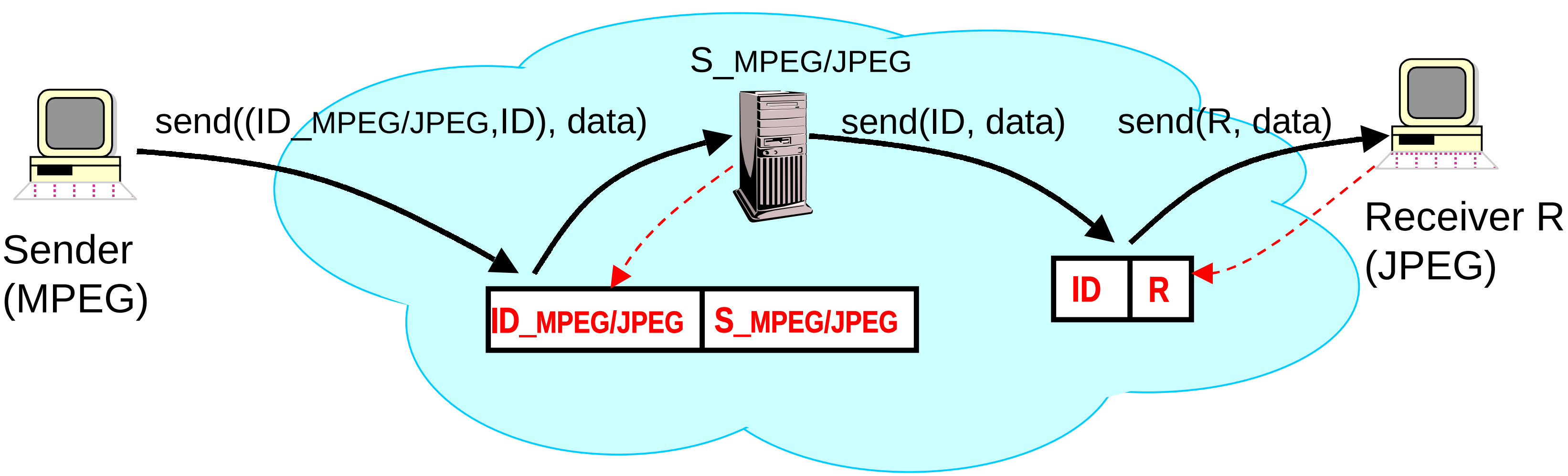
Multicast

- Unifies multicast and unicast abstraction
  - Multicast: receivers insert triggers with the same identifier
- An application can dynamically switch between multicast and unicast



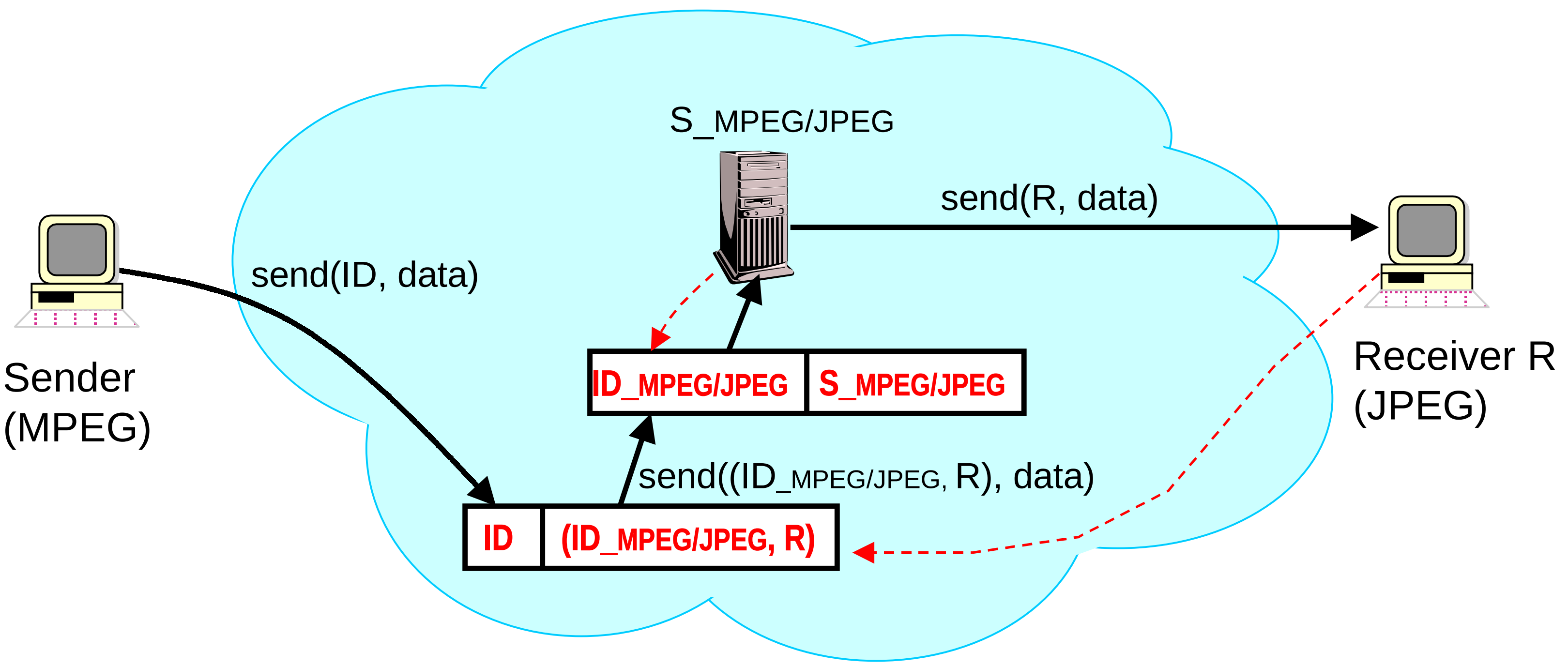
Composable Services

- Use a stack of IDs to encode the successions of operations to be performed on data (e.g., transcoding)
- Advantages
  - Don't need to configure path
  - Load balancing and robustness easy to achieve



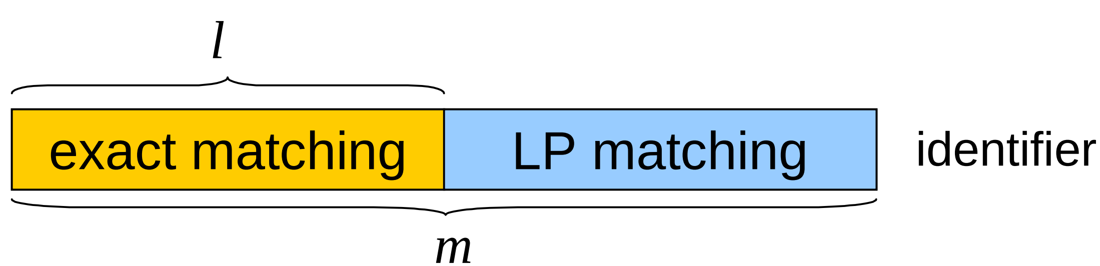
Composable Services (cont'd)

- Both receivers and senders can specify the operations to be performed on data



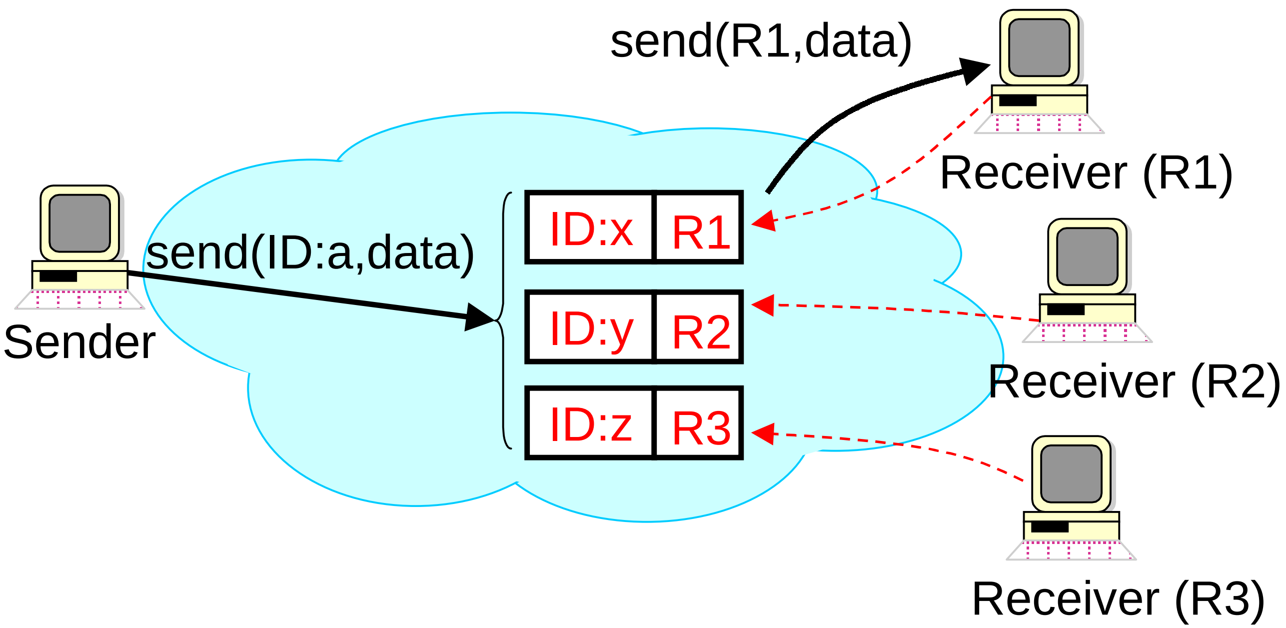
Anycast

- Generalize the matching scheme used to forward a packet
  - Until now we assumed exact matching
- Next, we assume:
  - Exact matching on the most significant l bits of ID
  - Longest prefix matching on the remaining bits (ID size = m)



Anycast (cont'd)

- Anycast is simply a byproduct of the new matching scheme
  - Each receiver in the anycast group inserts IDs that differ only in the last *l*-*m* bits



Anycast (cont'd)

- Highly flexibile: the least significant l-m bits of ID are application specific
- Two examples:
  - Load balancing
  - Proximity

Idea 1: Load Balancing

- Assumptions:
  - N servers of capacity C_i, 1 <= i <= N
  - M clients downloading files from these servers
- Goal: come up with an algorithm to insert triggers and set up their identifiers such that to balance the load in the presence of server failures

Idea 2: Multicast

- Problem: triggers with the same identifiers are stored at the same server → not scalable
- Problem: extend *i*3 infrastructure to support large scale multicasts

Idea 3: Transcoding Application

- Design a transcoding application
  - From one video format to another (e.g., MPEG → H.263), or
  - From one data format to another (e.g., HTML → WML)
- Note: the goal of the project is not to design the transcoder, but to demonstrate the service composition function

Idea 4: Migrate-able End-to-End Protocols

- Modify TCP such that it is possible to change the receiving machine in the middle of the transfer!
- A and B open a TCP connection (A receiver; B source)
- A changes to A’
- B continue to send data to A; without creating a new connection
- Challenge: transparently transfer the receiver state from A to A’

Idea 5: *i*3 in Sensornets

- Design and implement *i*3 in Sensornets
- Challenge: there is no undelying point to point communication in sensornets

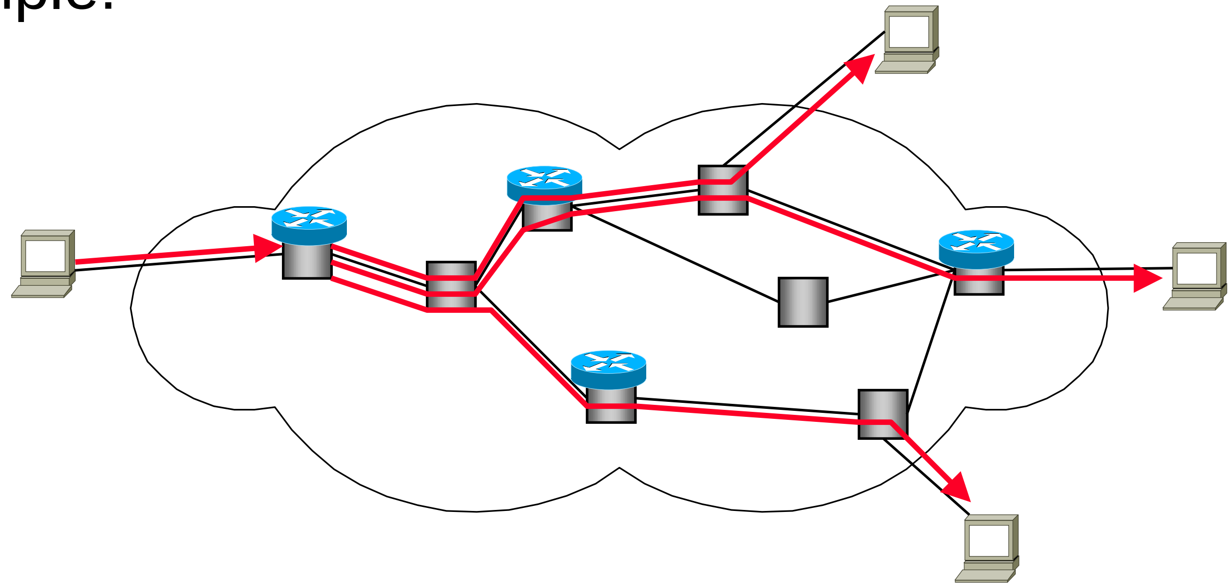
Other Project Ideas

Idea 6: Reducing (elimination) Multicast State in Routers

- Today each router maintain state for each multicast group that has traffic traversing it
- Problem: state is hard to maintain and manage → not scalable
- Extreme solution: maintain all receiver addresses in each packet
  - Routers don't need to maintain any state, but
  - Packet headers can become very large → huge overhead
- Solution: design an algorithm in between
  - Maintain some state in routers and some in packets
- Note: you can think either at the IP or application layer

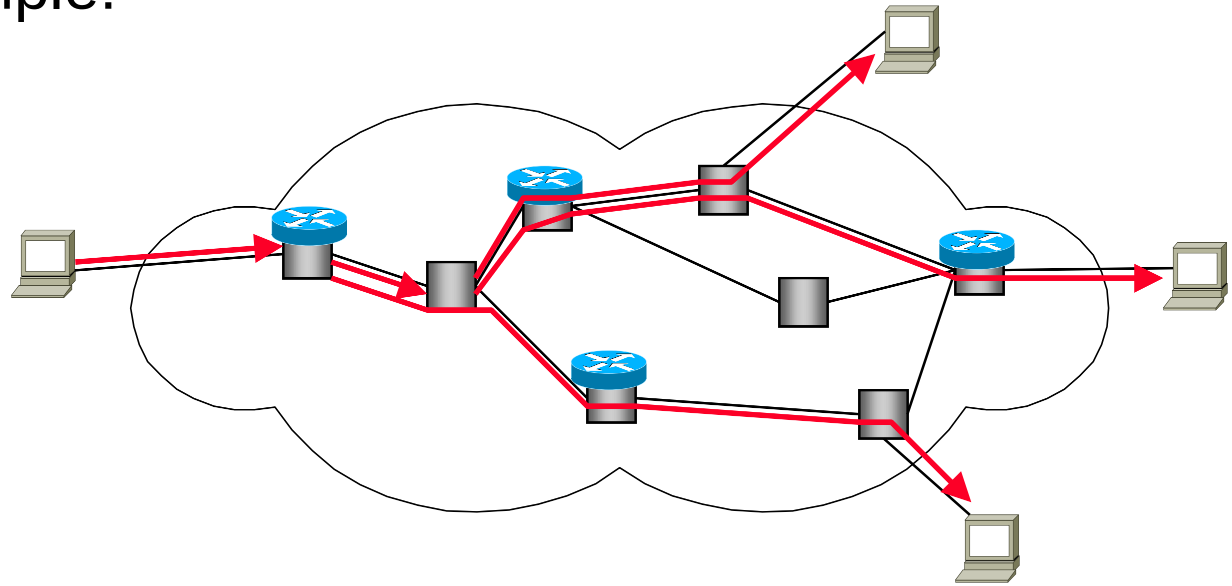
Idea 7: A Self-Organizing Overlay Multicast Tree Algorithm

- Goal: design and simulate a self-organizing multicast tree algorithm for overlay networks
- Algorithm idea: have overlay nodes decide to add/collapse branches in the multicast tree
- Example:



Idea 7: A Self-Organizing Overlay Multicast Tree Algorithm

- Goal: design and simulate a self-organizing multicast tree algorithm for overlay networks
- Algorithm idea: have overlay nodes decide to add/collapse branches in the multicast tree
- Example:



Forwarding in Low Energy Wireless Networks

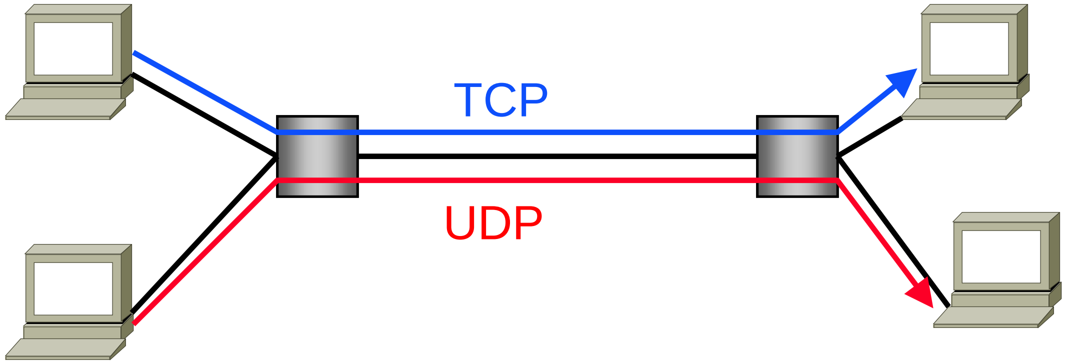
- Problem: each node cannot afford to remain ON all the time
  - a node can communicate/receive data only when it is ON
- Two nodes can communicate only when both of them are simultaneously ON
- A node stores a packet in transit until it finds the next hop ON

Ideas 8 & 9

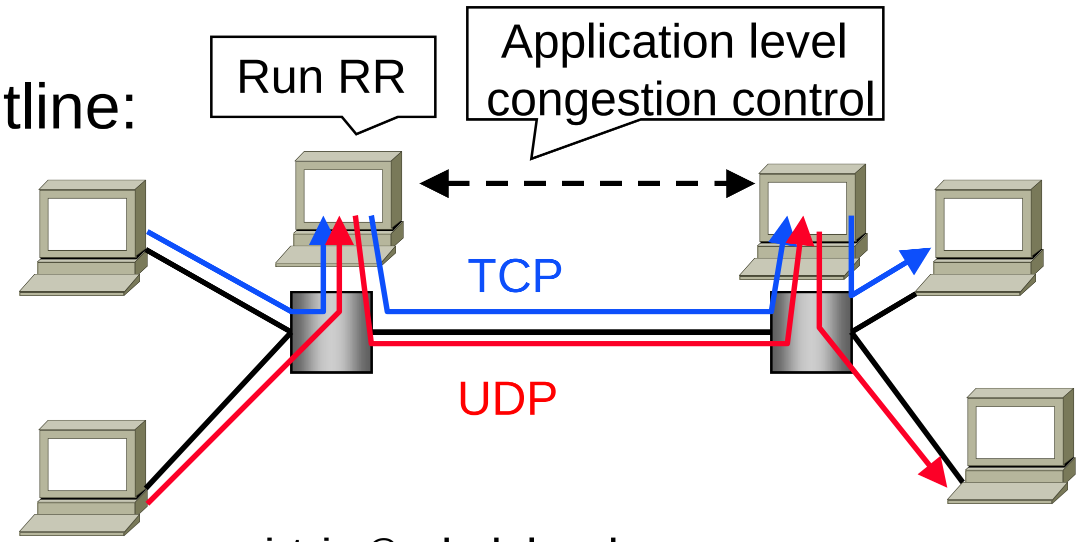
- Assume routing tables are known
- Assume that each node is independently switching between ON and OFF states
- Idea 8:
  - Study the tradeoff between the fraction of time a node is ON and the time to deliver a message and the amount of storage required by a node
- Idea 9:
  - Design a self-synchronization algorithm and study its properties (i.e., a distributed algorithm that will result in all nodes being ON at the same time)

Idea 10: Implement Round Robin at the Application Layer

- Problem: flow isolation (UDP can kill TCP)

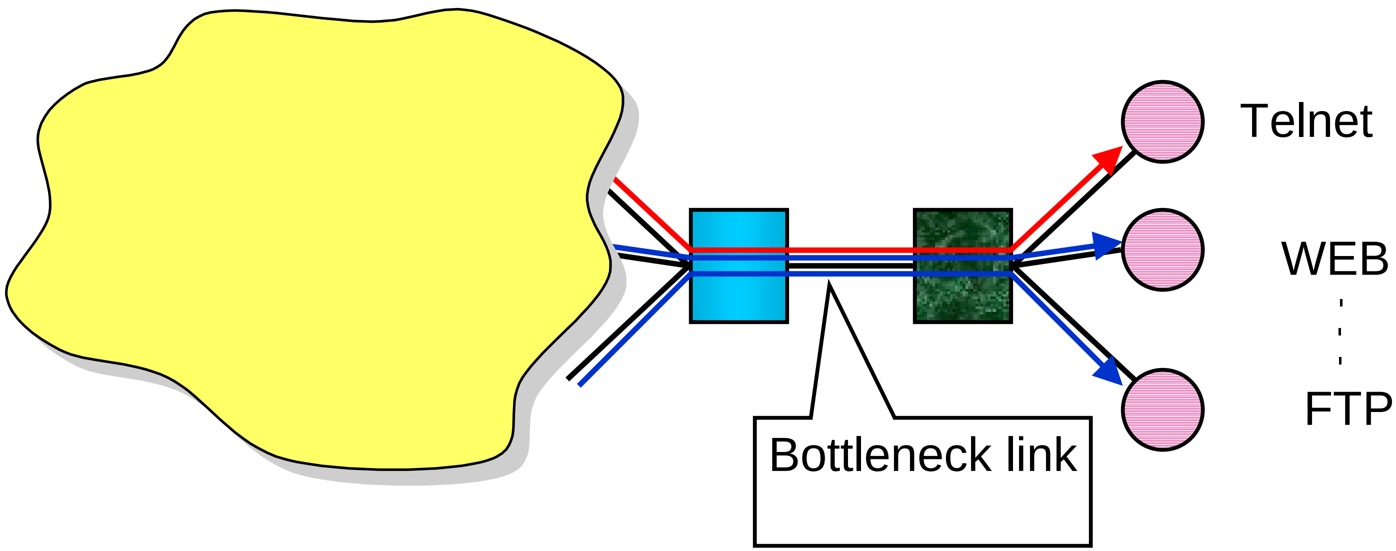


- Solution outline:



Idea 11: Receiver-Controlled Cooperative Sharing

- You control the downstream router or the end host
- You want to control the bandwidth allocation policy
  - Manipulate TCP packets to adjust the sender’s transmission rate



Next Step

- You can either choose one of the projects we discussed during this lecture, or come up with your own
- Pick your partner, and submit a one page proposal by February 13. The proposal needs to contain:
  - The problem you are solving
  - Your plan of attack with milestones and dates
  - Any special resources you may need