Today: routing

* Routers forward a packet for one hop by looking at the IP address
* How to send the packet without knowing anything about the topology?
  + Send it to a random next hop.
  + Flooding: send it to everybody.
  + Source Routing: the original sender list all routers along the path
  + Distributed algorithm: routers construct a forwarding table by talking with each other
* Flooding: routers forward a packet to every interface, except for the one it comes from
  + Cons:
    - infinite loops. TTL field was originally designed to avoid the infinite loops.
    - Inefficient usage of links: a packet would use all links at least once
    - Packets are delivered to everyone
  + Pros: packets arrive the destination at the shortest possible path
* Source Routing: the source hosts specify the whole path (A -> R1 -> R2 -> R3 -> B) or part of the path (the path has to go through R2)
  + Cons:
    - Revealing the underlying network topology causes potential vulnerabilities, and ISPs do not want to reveal that to users (due to security/competition)
    - Potentially large headers and all sources need to know the networking topology to do this
  + Pros:
    - No loops
    - No need for tables at routers
* If (the packet is for this router’s Ethernet address) {  
   Check the IP version number and length of datagram;  
   Decrement TTL, and update IP header checksum;  
   If (TTL == 0) drop;  
   If (IP destination address is in the forwarding table) {  
   Forward to the correct port;
* Routing tables: map prefix of IP destination addresses to next hop IP address
  + X.Y/Z is a Z-bit prefix IP destination address
  + Looking up in the routing tables looks for the longest prefix match
* How to construct routing tables?
  + If packets from any source router should be delivered to router X exactly once along the shortest path, the routing table for destination router X, is a spanning tree with root at router X.
  + Routers build a spanning tree for each destination.
* Bellman-Ford algorithm: every router advising its # hops from the destination, and update that number when they received a smaller # hops (new number + 1 < old number) from its neighbors, and breaks ties at random.