Has everyone taken a copy of each of these 4 handouts?

Welcome to this course! My name is Jasleen Kaur and I'm the instructor for this course. We'll spend today's class talking about the course outline, the course requirements, and your first homework. But before we do that, I'd like for us all to spend some time and introduce ourselves. So if each of you can say 3 things: (1) what is your name, (2) what is your major and year, and (3) what is it that you expect to learn from this course.

So let me begin: my name is Jasleen and I conduct research in the design and analysis of networks and distributed systems. And I'm looking forward to a semester-worth of exchanging ideas with all of you.

How many have NOT had socket programming?

What do you think is the “Internet”?

Intra-domain Routing

Formulation

- Intra-domain routing \(\Rightarrow\) \(\sim\) 100 routers

- Given:
  - Graph: where nodes are routers and edges are links
  - Cost: associated with each link

- Find:
  - Lowest-cost path between any two nodes

- Requirements:
  - Self-healing, traffic-sensitive, scalable

Need dynamic, distributed algorithms!

Two classes: based on “distance-vector” and “link-state”
There are at least two ways to think about the Internet (and also about what we will cover in this course): The first is in terms of the components that make up the Internet. So this would be a nuts-and-bolts view in which we can talk about several acronyms and terms that you may have come across, such as the:

- web
- TCP/IP
- inter-network
- routers
- switches

The second way to think about it is the one that is more common, which is in terms of the services and applications that run over the Internet. So all of us do web-browsing, exchange emails, share files.

The important point to note is that these applications we care about do not operate solely by themselves. There are numerous services that all work together to provide a seamless view that an application is doing everything by itself.

— In this class we expose some of the seams. The services that enable such applications are all part of the Internet. So we will spend time talking about how these services work?

### Link-state Routing

**Basic Idea**

- Speed of convergence is key advantage of link-state routing
- Approach: if each node has complete info about all links, it can:
  - Build complete map of network
  - And compute shortest path to any node
- Two key mechanisms:
  - Reliable dissemination (of complete link-state of the network)
  - Calculation of routes (from the sum of accumulated link-state)

### Link State Routing

**Reliable Flooding**

- On link-cost changes, and periodically, each node creates a link-state packet (LSP) that contains:
  - For enabling route-computation
    - ID of node that created it
    - List of directly-connected neighbors + cost of link to each
  - For ensuring reliability of flooding
    - Sequence number
    - TTL for this packet
- Transmission of LSPs between adjacent routers is made reliable
  - Using ACKs and retransmissions
- When K receives an LSP originated at Y, it stores it if:
  - Has no previous state (or has only smaller sequence number) from Y
  - If it stores, it also forwards to all neighbors (except one who forwarded LSP)
Link State Flooding Algorithm
Example

1. Step 1: A link fails and nodes C and D notice it.

2. Step 2: C and D send the updated link information to all attached neighbors with a timestamp.

3. Step 3: A floods the $D \rightarrow C$ update message it received from D and F floods the $C \rightarrow D$ update message it received from C.

4. Step 4: B floods the $C \rightarrow D$ update message it received from C and E floods the $D \rightarrow C$ update message it received from D.

5. Step 5: A floods the $C \rightarrow D$ update message it received from C, C floods the $D \rightarrow C$ update message it received from A (note that C received the same message from E in step 4), E floods the $D \rightarrow C$ update message it received from C (in Step 2).

6. Step 6: B floods the $D \rightarrow C$ update message it received from D in Step 2 (note that B received the same message from A in Step 3), F floods the $D \rightarrow C$ update message it received from E in Step 4, D floods the $C \rightarrow D$ update message it received from B in Step 4.
**Link State Routing**

**Dijkstra’s Algorithm**

1. **Initialization:**
   2. \( N = \{A\} \)
   3. for all nodes \( v \)
   4. if \( v \) adjacent to \( A \)
   5. then \( D(v) = c(A,v) \)
   6. else \( D(v) = \infty \)

8. **Loop**
   9. find node \( w \) not in \( N \) such that \( D(w) \) is a minimum
   10. add node \( w \) to \( N \)
   11. update \( D(v) \) for all nodes \( v \) adjacent to \( w \) and not in \( N \):
   12. \( D(v) = \min(D(v), D(w) + c(w,v)) \)
   13. /* new cost to node \( v \) is either old cost to \( v \) or known
   14. shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
   15. until all nodes in \( N \)

\( N \) is the set of nodes to which we have computed the minimum cost path

\( D(x) \) is the current minimum cost path to \( x \)

\( c(n,m) \) is the cost of the link from \( n \) to \( m \)

---

**Link State Routing**

**Dijkstra’s Algorithm: Example**

<table>
<thead>
<tr>
<th>Step</th>
<th>start</th>
<th>( D(B), p(B) )</th>
<th>( D(C), p(C) )</th>
<th>( D(D), p(D) )</th>
<th>( D(E), p(E) )</th>
<th>( D(F), p(F) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>\infty</td>
<td>\infty</td>
</tr>
</tbody>
</table>

\( N \) is the set of nodes to which we have computed the minimum cost path

\( D(x) \) is the current minimum cost path to \( x \)

\( p(x) \) is the predecessor of \( x \) on the current minimum cost path to \( x \).
**Link State Routing**

*Oscillating Routes*

- "Route oscillations" are possible in link state algorithms
- Let the link cost equal the amount of carried traffic
  - Assume the link cost is updated as traffic changes

Initially ... recompute ... recompute ... recompute

<table>
<thead>
<tr>
<th>Node</th>
<th>Path to A</th>
<th>Node</th>
<th>Path to A</th>
<th>Node</th>
<th>Path to A</th>
<th>Node</th>
<th>Path to A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>A</td>
<td>B</td>
<td></td>
<td>C</td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>D</td>
<td></td>
<td>B</td>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>A</td>
<td></td>
<td>C</td>
<td></td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>D</td>
<td></td>
<td>D</td>
<td></td>
<td>C</td>
<td>B</td>
</tr>
</tbody>
</table>

**Least Cost Path Computations**

*Link-state vs. Distance-vector Algorithms*

- **Message complexity:**
  - LS: With $N$ nodes, $E$ links, $O(NxE)$ messages sent for flooding
  - DV: Exchange between neighbors only (may trigger further exchanges)
    - Due to reliable flooding, LS considered to generate less traffic

- **Speed of Convergence:**
  - LS: $O(N^2)$ algorithm and $O(NxE)$ messages
    - May have oscillations depending on choice of metric
  - DV: Convergence time varies
    - Routing loops possible
    - Count-to-infinity problem

- **Robustness: what happens if there are failures?**
  - LS: Node can advertise incorrect link cost
    - Each node computes only its own table
  - DV: Node can advertise incorrect path cost
    - Each node's table used by others
    - Errors propagate through network