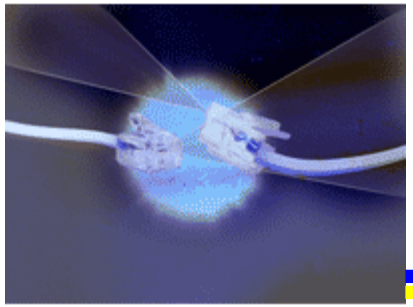
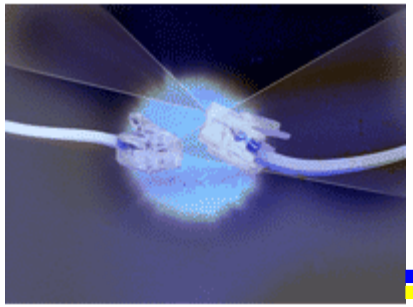


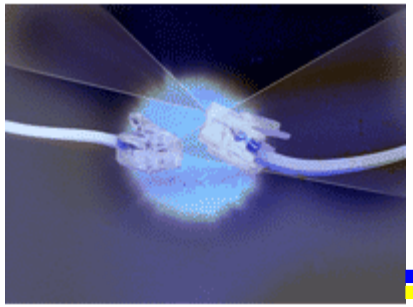
Routing

- Process of distributing information through network so routers can build forwarding tables
- Forwarding vs. routing tables
  - Forwarding table maps network number to interface, data link address
  - Routing table maps network number to next hop router
  - May be combined in implementation



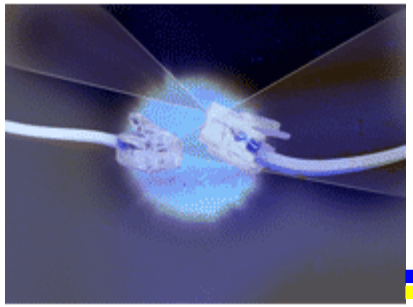
Routing (cont.)

- To scale, organize routing into hierarchy
  - Intradomain routing – interior gateway protocols (IGPs)
  - Interdomain routing
- Domain is a relatively small collection of networks, typically under the same administrative control



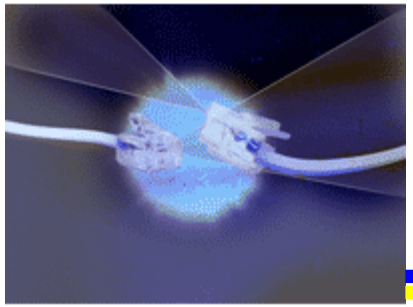
Networks as Graphs

- Vertices are routers, edges are links.
  - Each edge has a cost that indicates how *expensive* that link is
- Routing is graph problem – find lowest-cost path between two vertices
- Want to maintain information dynamically, to tolerate link or node failures, handle additions, and allow costs to change



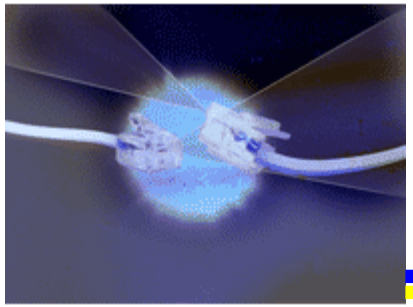
Network Graphs (cont.)

- Graph problem is distributed among routers, which creates potential problems
  - Different routers can disagree about the current state of the network graph
  - Can create loops between routers until difference is resolved



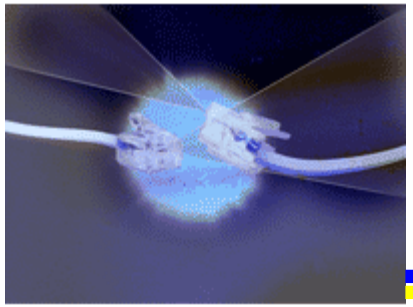
Distance Vector Routing

- Also known as Bellman-Ford algorithm
- Each node builds table of distances to other nodes
  - Initially, assume node knows cost to each directly connected neighbor; simple measure is to assign value of 1 to each link
  - Broken links or links with unknown cost assigned infinite cost



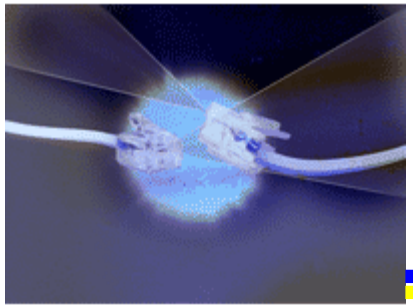
Distance Vector Routing (cont.)

- Each node then distributes this table to all immediate neighbors
- When router receives table from neighbor, it merges the table with its own
  - If entry in neighbor's table plus cost of link from neighbor is smaller than node's current entry, replace
  - If value is larger, ignore



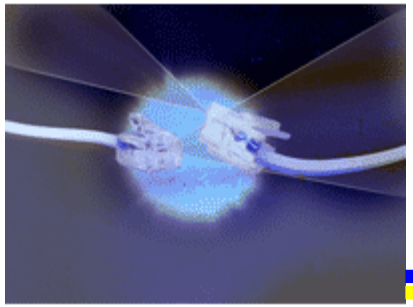
Distance Vector Routing (cont.)

- After exchanging updates repeatedly, every node's table will *converge* to a static state
- Two update mechanism
  - Periodic: node sends information at regular intervals, even if nothing has changed (frequency seconds to minutes)
  - Triggered: whenever table changes, send out update



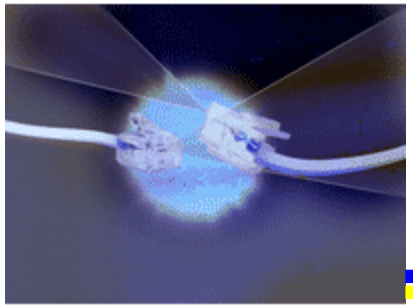
Link or Node Failure

- Nodes monitor links
  - Actively, by sending control message
  - Passively, by watching for routing updates
- If link is down, node changes cost to ∞ and sends out update
- If a neighbor's path to some destination is through the node, neighbor needs to update its own table



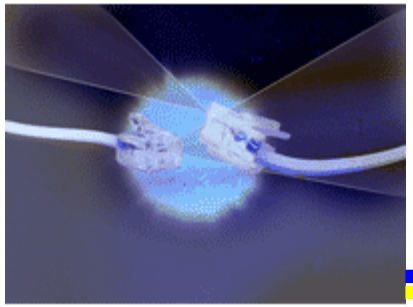
Link or Node Failure (cont.)

- This update might trigger additional cascading updates; network will eventually reconverge
- Can create *count-to-infinity* problem
  - Cycle in graph can lead to nodes on cycle constantly updating counts, never realizing that a node is down



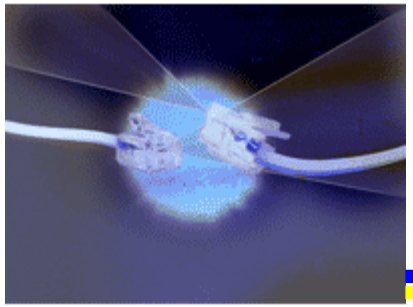
Count-to-Infinity Solutions

- Simplistic: pick a relatively small number and consider anything > that as infinity
  - Breaks down if network size exceeds that limit
- Split horizon solution
  - Keep track of where node learned about route
  - Don’t send updates to a route to neighbor from which node learned about route
  - Poison reverse – actually send update with infinite cost to guarantee neighbor won’t use



Count-to-Infinity (cont.)

- Only works for cycle between two nodes
- For larger cycles, can introduce delay; node doesn't send out update to table immediately
  - Can slow convergence
  - Alternative is link-state routing



Routing Information Protocol (RIP)

- Standard distance vector-based protocol widely used in IP networks
  - Entries are networks rather than nodes
  - Can support other address families besides IP
- RIP sends updates every 30 sec.
- Every link's cost is 1 (so distance is just hop count)
- 16 is considered infinite