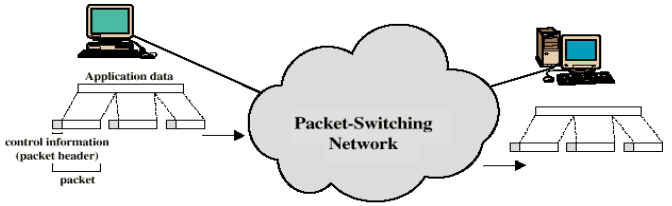
Chapter 10 Packet Switching

- Packet Switching Principles
  - Switching Techniques (***Datagram, Virtual Circuit***)
  - Packet Size
  - Comparison of Circuit Switching & Packet Switching
- Routing
  - Least-Cost Routing Algorithms
    - **Dijkstra's Algorithm**
    - **Bellman-Ford Algorithm**
  - Routing Strategies

Principles

- Circuit switching designed for voice
  - Resources dedicated to a particular call
  - Much of the time a data connection is idle
  - Data rate is fixed: Both ends must operate at the same rate
  - Not efficient for bursty type of data traffic
- Packet Switching: suit better for data network application



Basic Operation of Packet Switching

- Data transmitted in small packets
  - Maximum Size 1000 Bytes
  - Longer messages split into series of packets
  - Each packet contains a portion of user data plus some control info
- Control info: Routing (addressing) info
- Packets are received, stored briefly (buffered) and passed on to the next node
  - Store-and-forward

Advantages

- Line efficiency
  - Single node to node link can be shared by many packets over time (Statistic Time Division Multiplexing based)
  - Packets queued and transmitted as fast as possible
- Data rate conversion
  - Each station connects to the local node at its own speed
  - Nodes buffer data if required to equalize rates
- Packets are accepted even when network is busy
  - Delivery may slow down
- Priorities can be used

Switching Technique

- Host breaks long message into packets
- Packets sent one at a time to the network
- Packets handled in two ways
  - Datagram
  - Virtual circuit

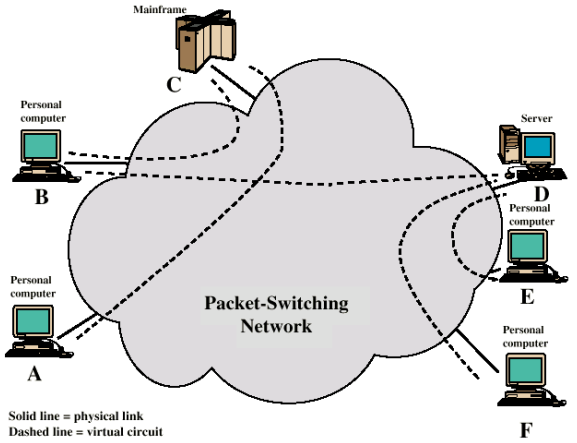
Datagram

- Each packet treated independently
- Packets can take any practical route
- Packets may arrive out of order
- Packets may go missing
- Up to receiver to re-order packets and recover from missing packets

Virtual Circuit

- Preplanned route established before any packets sent
- Call request and call accept packets establish connection (handshake)
- Each packet contains a virtual circuit identifier instead of destination address
- No routing decisions required for each packet
- Clear request to drop circuit
- Not a dedicated path though!

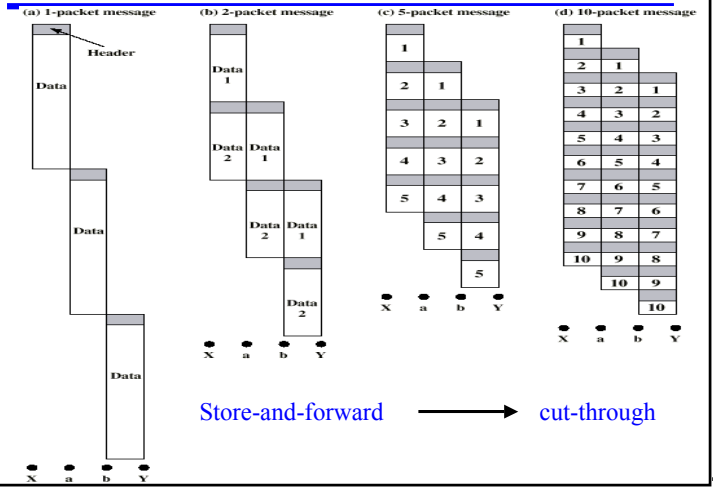
X.25: Use of Virtual Circuits



Virtual Circuits vs. Datagram

- Virtual circuits
  - Network can provide sequencing and error control
  - Packets are forwarded more quickly
    - No routing decisions to make
  - Less reliable
    - Loss of a node looses all circuits through that node
- Datagram
  - No call setup phase
    - Better if few packets
  - More flexible
    - Routing can be used to avoid congested parts of the network

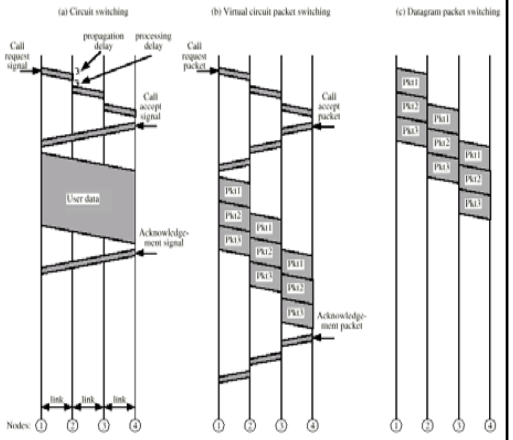
Packet Size



Circuit vs. Packet Switching

- CS based on **FDM & Synchronous TDM**; PS based on **Statistic TDM**
- Performance
  - Propagation delay
  - Transmission time
  - Node delay
- Reading Assignment:

P312: table 10.1



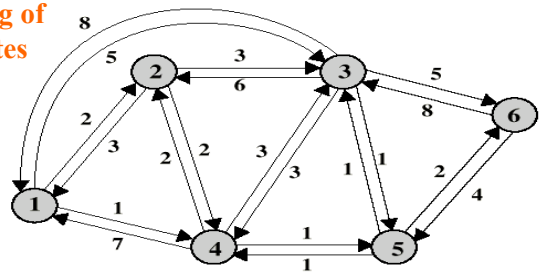
Routing in Packet Switching Networks

- **Complex, crucial aspect of packet switched networks**
- **Characteristics required**
  - **Correctness**
  - **Simplicity**
  - **Robustness**
  - **Stability**
  - **Fairness**
  - **Optimality**
  - **Efficiency**

Performance Criteria

- Used for selection of route: e.g. Minimum hop

Costing of Routes



- Least cost (appendix 10A)
  - *Dijkstra's Algorithm*
  - *Bellman-Ford Algorithm*

Dijkstra's Algorithm

- A greedy algorithm finding shortest paths from source node to all other nodes
- **Define:**

N: set of nodes in the network

S: source node

T: set of nodes so far incorporated in the algorithm

W(i,j): link cost from node i to node j;

W(i,i)=0;

w(i,j)=∞ for two none directly connected nodes

w(i,j) =c

L(n): least cost path from node s to node n that is currently known to the algorithm. At the end, this is the final least cost path

Dijkstra's Algorithm (Con't)

Step 1: Initialization

T = {s} : at initial stage, only source node included

L(n) = w(i,j) for all n ≠s

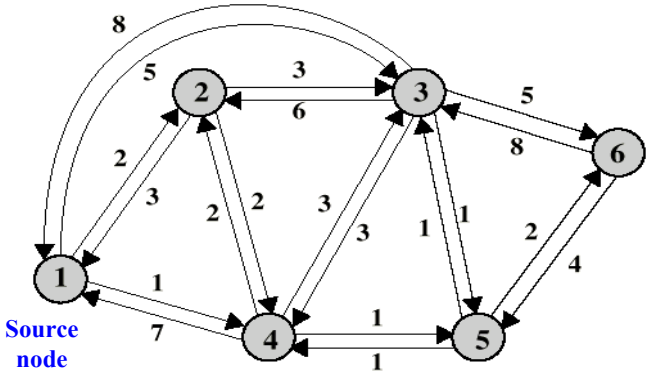
Step 2: Find the neighboring node not in T that has the least-cost path from node s and include that node into T, i.e., add node x to T

Step 3: Update least-cost paths due to node x included

L(n) = min [L(n), L(x) +w(x,n)]

Step 4: algorithm terminates when all nodes included in T, otherwise, go back to step 2

Dijkstra's Algorithm



Bellman-Ford Algorithm

Idea: Find the shortest paths from a given source node subject to the constraint that the paths contain at most one link; then find the shortest paths with a constraint of at most two links, and so on. Algorithm terminates when the increase of the maximum link number brings no improvement.

Define:

S: source node

W(i,j): link cost from node i to node j;

W(i,i)=0;

w(i,j)=∞ for two none directly connected nodes

w(i,j) =c

$L_h(n)$: least cost path from node s to node n under the constraint of no more than h links

Bellman-Ford Algorithm

Algorithm:

Step 1: Initialization

$L_0(n) = \infty$, for all $n \neq s$

$L_h(s) = 0$, for all h

Step 1: Update when $L_{h+1}(n)$ less than $L_h(n)$

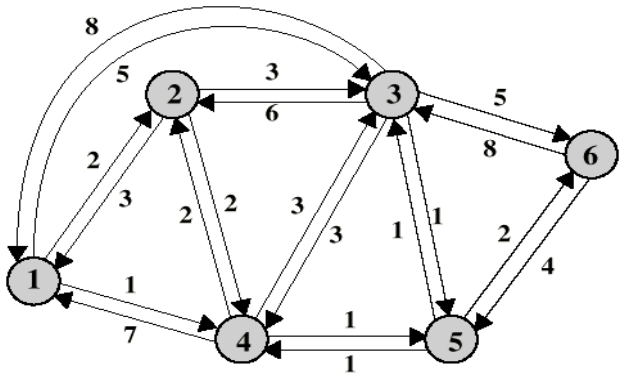
For each successive $h \geq 0$:

For each $n \neq s$, compute

$L_{h+1}(n) = \min [L_h(j) + w(j,n)]$ for all predecessor node j

Step 2: Terminate when increase of h brings no improvement

Bellman-Ford Algorithm



Network Information Source and Update Timing

- Routing decisions usually based on knowledge of network (not always)
- Distributed routing
  - Nodes use local knowledge
  - May collect info from adjacent nodes
  - May collect info from all nodes on a potential route
- Central routing: Collect info from all nodes
- Update timing
  - When network info nodes updated
  - Fixed - never updated
  - Adaptive - regular updates

Routing Strategies

- Fixed
- Flooding
- Random
- Adaptive

Reading Assignment: p318 – pp329