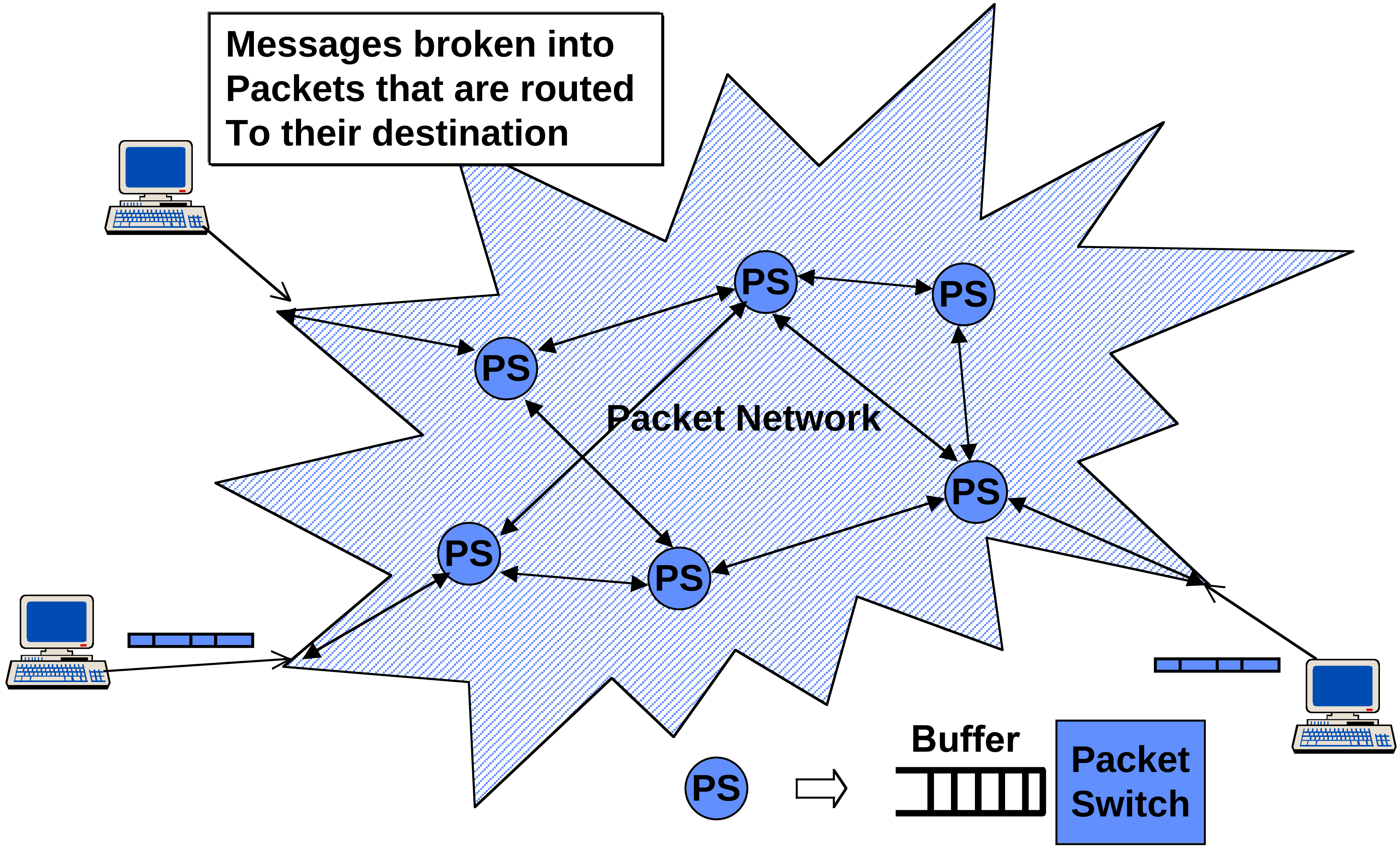
Routing in Data Networks

Eytan Modiano

Packet Switched Networks



Routing

- **Must choose routes for various origin destination pairs (O/D pairs) or for various sessions**
  - **Datagram routing: route chosen on a packet by packet basis**

    Using datagram routing is an easy way to split paths
  - **Virtual circuit routing: route chosen a session by session basis**
  - **Static routing: route chosen in a prearranged way based on O/D pairs**

Broadcast Routing

- **Route a packet from a source to all nodes in the network**

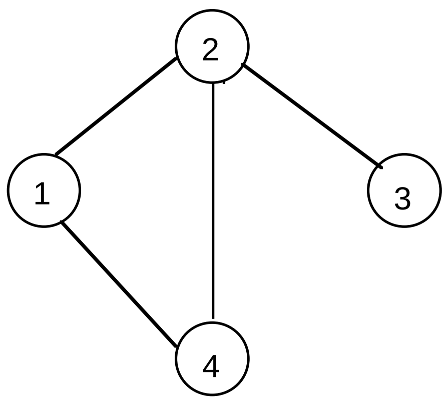
- **Possible solutions:**

  - **Flooding: Each node sends packet on all outgoing links**
    Discard packets received a second time

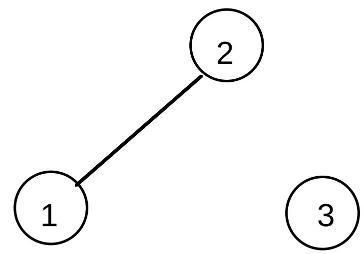
  - **Spanning Tree Routing: Send packet along a tree that includes all of the nodes in the network**

Graphs

- **A graph G = (N,A) is a finite nonempty set of nodes and a set of node pairs A called arcs (or links or edges)**



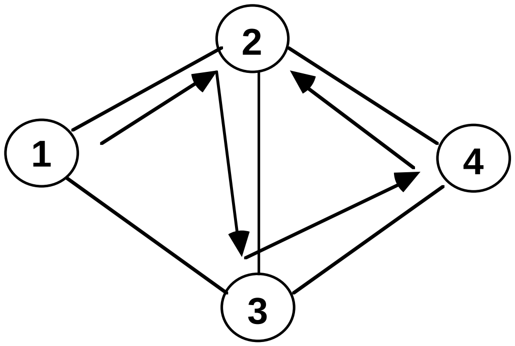
N = {1,2,3,4}
A = {(1,2),(2,3),(1,4),(2,4)}



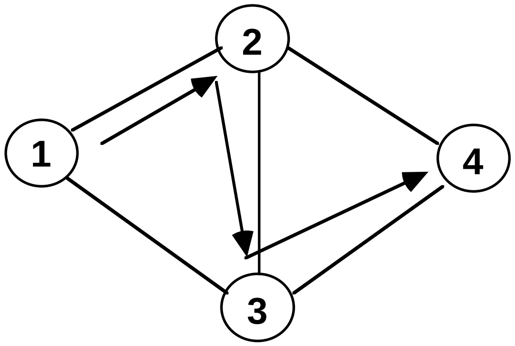
N = {1,2,3}
A = {(1,2)}

Walks and paths

- A walk is a sequence of nodes (n1, n2, ...,nk) in which each adjacent node pair is an arc.
- A path is a walk with no repeated nodes.



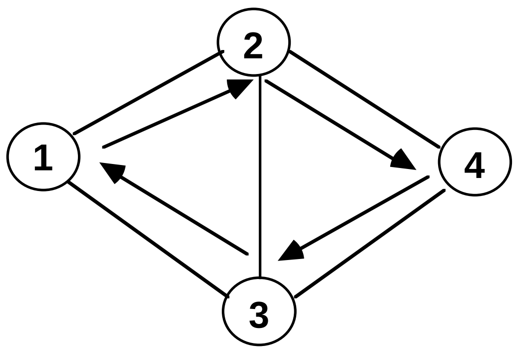
Walk (1,2,3,4,2)



Path (1,2,3,4)

Cycles

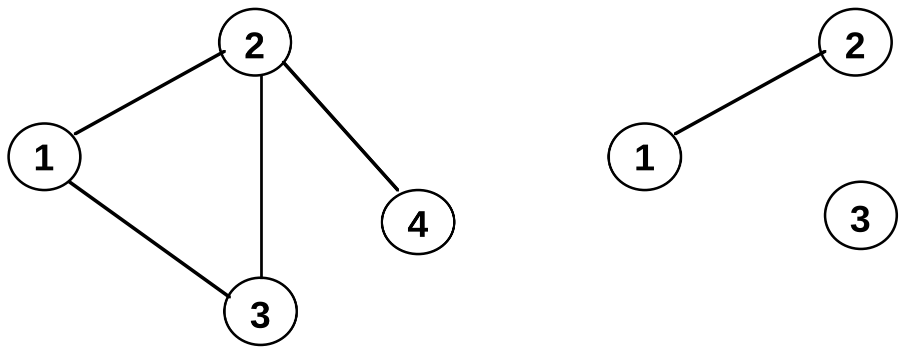
- **A cycle is a walk (n1, n2,...,nk) with n1 = nk, k>3, and with no repeated nodes except n1 = nk**



Cycle (1,2,4,3,1)

Connected graph

- A graph is connected if a path exists between each pair of nodes.

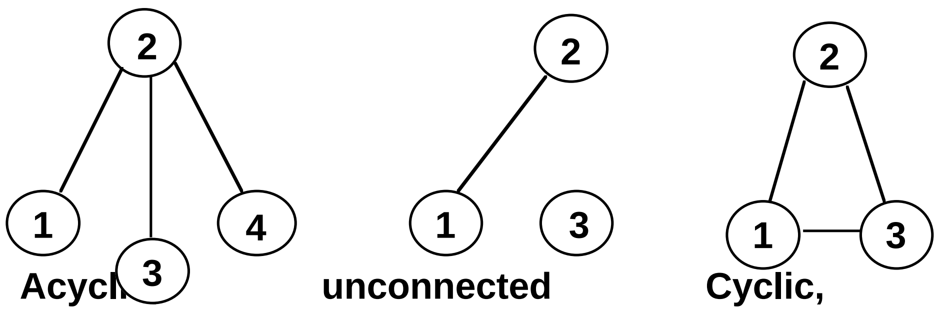


Connected Unconnected

- An unconnected graph can be separated into two or more connected components.

Acyclic graphs and trees

- An acyclic graph is a graph with no cycles.
- A tree is an acyclic connected graph.



Acycl
connected

unconnected
not tree

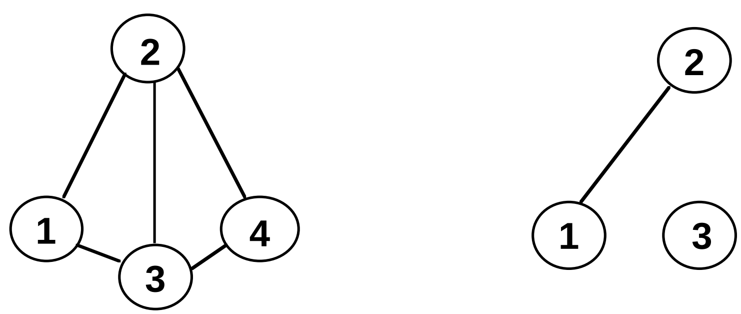
Cyclic,
not tree

- The number of arcs in a tree is always one less than the number of nodes
  - Proof: start with arbitrary node and each time you add an arc you add a node => N nodes and N-1 links. If you add an arc without adding a node, the arc must go to a node already in the tree and hence form a cycle

Subgraphs

- G' = (N',A') is a subgraph of G = (N,A) if
  - 1) G' is a graph
  - 2) N' is a subset of N
  - 3) A' is a subset of A

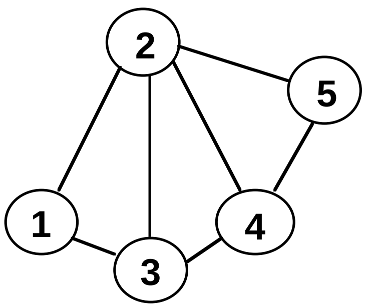
- One obtains a subgraph by deleting nodes and arcs from a graph
  - Note: arcs adjacent to a deleted node must also be deleted



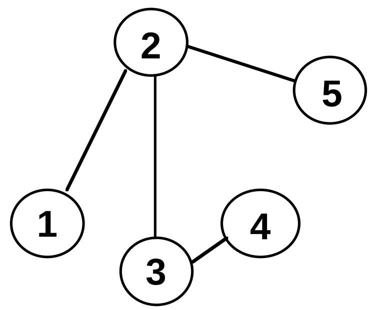
– Graph G Subgraph G' of G

Spanning trees

- T = (N',A') is a spanning tree of G = (N,A) if
  - T is a subgraph of G with N' = N and T is a tree



Graph G



Spanning tree of G

Spanning trees

- Spanning trees are useful for disseminating and collecting control information in networks; they are sometimes useful for routing

- To disseminate data from Node n:
  - Node n broadcasts data on all adjacent tree arcs
  - Other nodes relay data on other adjacent tree arcs

- To collect data at node n:
  - All leaves of tree (other than n) send data
  - Other nodes (other than n) wait to receive data on all but one adjacent arc, and then send received plus local data on remaining arc

General construction of a spanning tree

- **Algorithm to construct a spanning tree for a connected graph G = (N,A):**

  1) Select any node n in N; N' = {n}; A' = { }

  2) If N' = N, then stop (T=(N',A') is a spanning tree)

  3) Choose $(i,j) \in A$, $i \in N'$, $j \notin N'$

  $N' := N' \cup \{j\}$; $A' := A' \cup \{(i,j)\}$; go to step 2

- **Connectedness of G assures that an arc can be chosen in step 3 as long as $N' \neq N$**

- **Is spanning tree unique?**

- **What makes for a good spanning tree?**

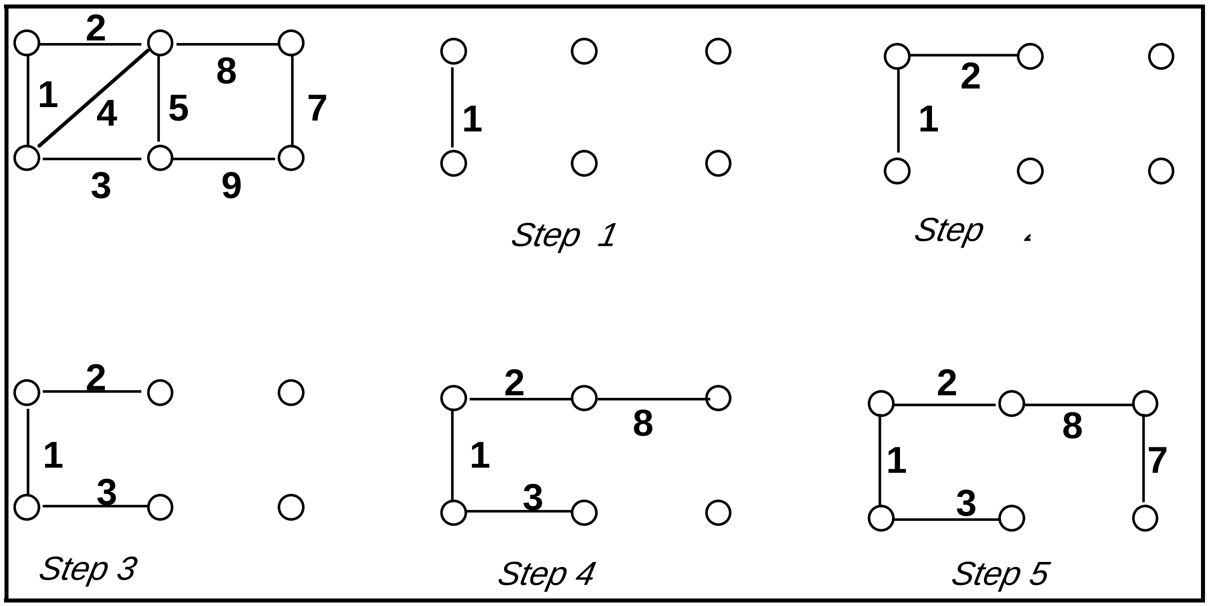
Minimum Weight Spanning Tree (MST)

- **Generic MST algorithm steps:**
  - **Given a collection of subtrees of an MST (called fragments) add a minimum weight outgoing edge to some fragment**

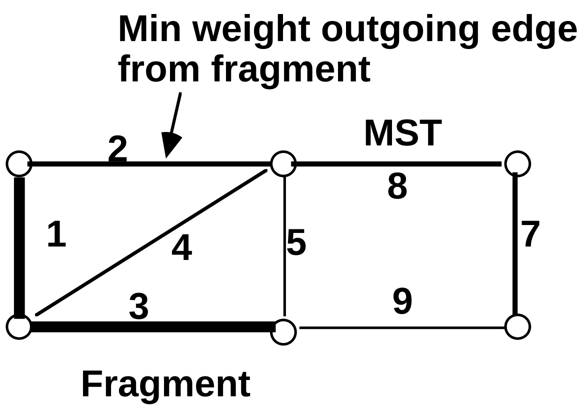
- **Prim-Dijkstra: Start with an arbitrary single node as a fragment**
  - **Add minimum weight outgoing edge**

- **Kruskal: Start with each node as a fragment;**
  - **Add the minimum weight outgoing edge, minimized over all fragments**

Prim-Dijkstra Algorithm



Kruskal's Algorithm Example



- **Suppose the arcs of weight 1 and 3 are a fragment**
  - **Consider any spanning tree using those arcs and the arc of weight 4, say, which is an outgoing arc from the fragment.**
  - **Suppose that spanning tree does not use the arc of weight 2.**
  - **Removing the arc of weight 4 and adding the arc of weight 2 yields another tree of smaller weight.**
  - **Thus an outgoing arc of min weight from fragment must be in MST.**