RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Distributed Systems
Course Code: CSE751/CSE20
Credits: 3:0:0/3:0:0:1

Term: Oct 2021– Feb 2022

Faculty:
Sini Anna Alex

RAMAIAH
Institute of Technology

Topology abstraction and overlays

The topology of a distributed system can be typically **viewed as an undirected graph** in which the **nodes represent the processors** and the **edges represent the links connecting the processors**.

Physical - arrangement of devices on a computer network through the actual cables that transmit data.

Logical- routers and switches, communication mechanism for all nodes in a network.

Superimposed- peer-to-peer networks and client-server applications are overlay networks because their nodes run on top of the Internet

RAMAIAH
Institute of Technology

Elementary graph algorithms

The distributed algorithms here introduce the reader to the difficulty of designing distributed algorithms wherein **each node has only a partial view of the graph** (system), which is confined to its immediate neighbors.

Further, **a node can communicate with only its immediate neighbors along the incident edges**. Unless otherwise specified, we assume unweighted undirected edges, and asynchronous execution by the processors.

Communication is by message-passing on the edges

RAMAIAH
Institute of Technology

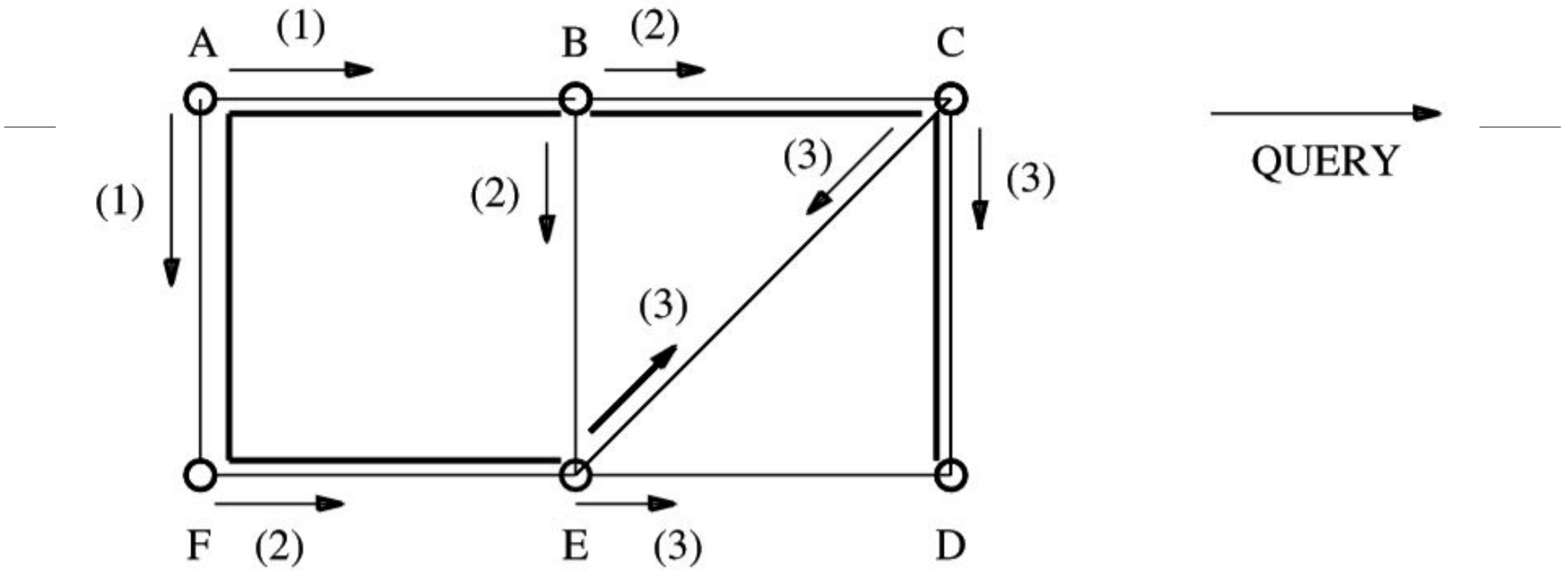
Synchronous single-initiator spanning tree algorithm using flooding

The code for all processes is not only symmetrical, but also proceeds in rounds. This algorithm assumes a designated root node, root, which initiates the algorithm.

The root initiates a flooding of QUERY messages in the graph to identify tree edges. The parent of a node is that node from which a QUERY is first received; if multiple QUERYs are received in the same round, one of the senders is randomly chosen as the parent.

```
(local variables)
int visited, depth ⟵ 0
int parent ⟵ ⊥
set of int Neighbors ⟵ set of neighbors
(message types)
QUERY

(1)  if i = root then
(2)        visited ⟵ 1;
(3)        depth ⟵ 0;
(4)        send QUERY to Neighbors;
(5)  for round = 1 to diameter do
(6)        if visited = 0 then
(7)              if any QUERY messages arrive then
(8)                    parent ⟵ randomly select a node from which
                         QUERY was received;
(9)                    visited ⟵ 1;
(10)                   depth ⟵ round;
(11)                   send QUERY to Neighbors \ {senders of
                         QUERYs received in this round};
(12)       delete any QUERY messages that arrived in this round.
```



Termination

The algorithm terminates after all the rounds are executed.

RAMAIAH
Institute of Technology

Asynchronous single-initiator spanning tree algorithm using flooding

The root initiates a flooding of QUERY messages in the graph to identify tree edges.

The parent of a node is that node from which a QUERY is first received; an ACCEPT message is sent in response to such a QUERY. Other QUERY messages received are replied to by a REJECT message.

Each node terminates its algorithm when it has received from all its non-parent neighbors a response to the QUERY sent to them.

RAMAIAH
Institute of Technology

Asynchronous single-initiator spanning tree algorithm using flooding

In this asynchronous system, there is no bound on the time it takes to propagate a message.

Unlike in the synchronous algorithm, each node here needs to track its neighbors to determine which nodes are its children and which nodes are not. This tracking is necessary in order to know when to terminate.

After sending QUERY messages on the outgoing links, the sender needs to know how long to keep waiting. This is accomplished by requiring each node to return an “acknowledgement” for each QUERY it receives.

The acknowledgement message has to be of a different type than the QUERY type.

(1) When the predesignated root node wants to initiate the algorithm:
(1a) **if** ($i = root$ **and** $parent = \perp$) **then**
(1b) **send** QUERY to all neighbors;
(1c) $parent \longleftarrow i$.

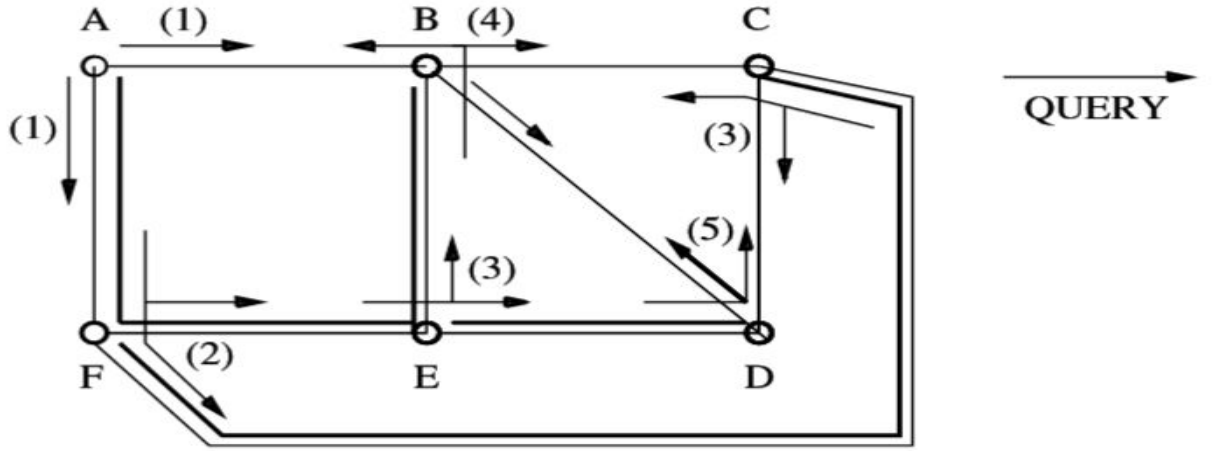
(2) When QUERY arrives from j:
(2a) **if** $parent = \perp$ **then**
(2b) $parent \longleftarrow j$;
(2c) **send** ACCEPT to j;
(2d) **send** QUERY to all neighbors except j;
(2e) **if** $(Children \cup Unrelated) = (Neighbors/\{parent\})$ **then**
(2f) **terminate.**
(2g) **else send** REJECT to j.

(3) When ACCEPT arrives from j:
(3a) $Children \longleftarrow Children \cup \{j\}$;
(3b) **if** $(Children \cup Unrelated) = (Neighbors/\{parent\})$ **then**
(3c) **terminate.**

(4) When REJECT arrives from j:
(4a) $Unrelated \longleftarrow Unrelated \cup \{j\}$;
(4b) **if** $(Children \cup Unrelated) = (Neighbors/\{parent\})$ **then**
(4c) **terminate.**

RAMAIAH
Institute of Technology

Working



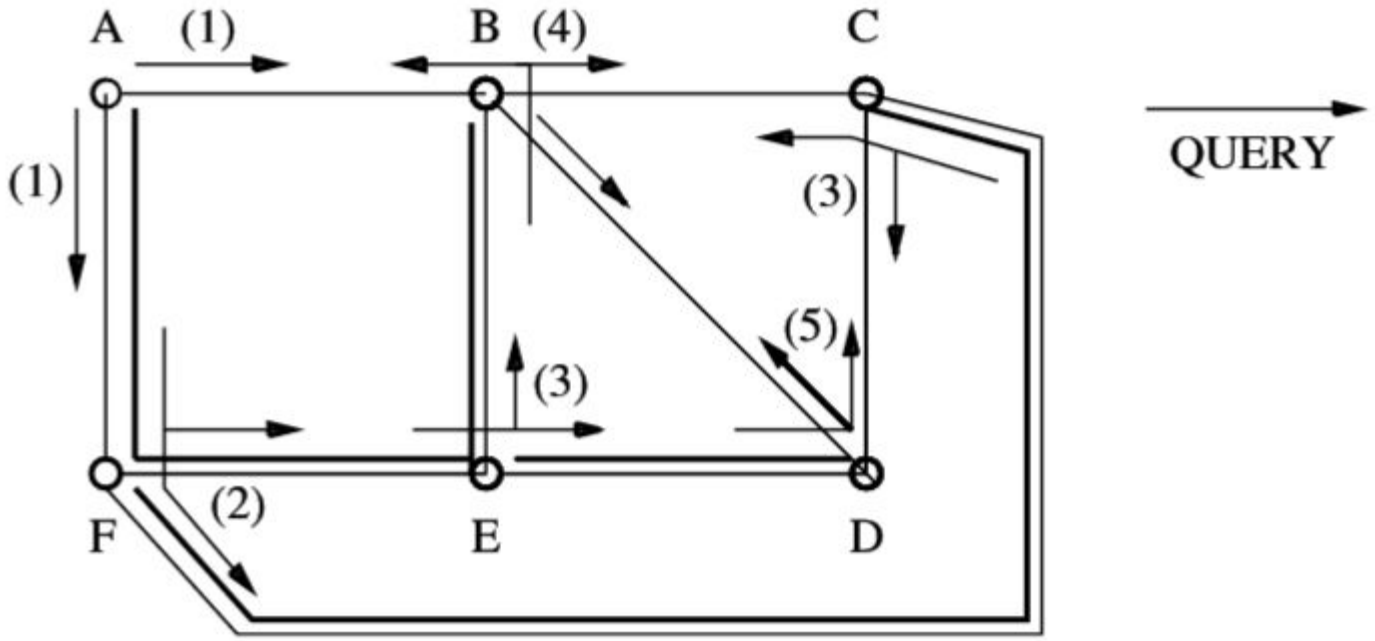
1. A sends a QUERY to B and F.

2. F receives QUERY from A and determines that AF is a tree edge. F forwards the QUERY to E and C.

3. E receives a QUERY from F and determines that FE is a tree edge. E forwards the QUERY to B and D. C receives a QUERY from F and determines that FC is a tree edge. C forwards the QUERY to B and D.

4. B receives a QUERY from E and determines that EB is a tree edge. B forwards the QUERY to A, C, and D.

5. D receives a QUERY from E and determines that ED is a tree edge. D forwards the QUERY to B and C.



The resulting spanning tree rooted at A is shown in boldface. The numbers next to the QUERY messages indicate the approximate chronological order in which messages get sent. The same numbering used for messages sent by different nodes implies that those actions occur concurrently and independently.

Thank you