Network Computing and Efficient Algorithms

Tree Algorithms

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Definitions

Definition 2.1(Broadcast).

A broadcast operation is **initiated by a single node**, the source. The source wants to **send a message to all other nodes** in the system.

Definitions

Definition 2.1(Broadcast).

A broadcast operation is **initiated by a single node**, the source. The source wants to **send a message to all other nodes** in the system.

Definition 2.2(Distance, Radius, Diameter).

The **distance** between two nodes u and v in an undirected graph G is the number of hops of a minimum path between u and v.
The **radius of a node** u is the maximum distance between u and any other node in the graph.
The **radius of a graph** is the minimum radius of any node in the graph.
The **diameter of a graph** is the maximum distance between two arbitrary nodes.

Definitions

Definition 2.1(Broadcast).

A broadcast operation is **initiated by a single node**, the source. The source wants to **send a message to all other nodes** in the system.

Definition 2.2(Distance, Radius, Diameter).

The **distance** between two nodes u and v in an undirected graph G is the number of hops of a minimum path between u and v.
The **radius of a node** u is the maximum distance between u and any other node in the graph.
The **radius of a graph** is the minimum radius of any node in the graph.
The **diameter of a graph** is the maximum distance between two arbitrary nodes.

$$R \leq D \leq 2R$$

Message Complexity

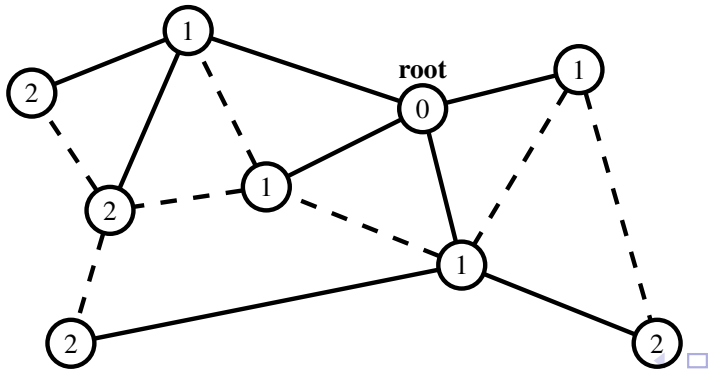
Definition 2.3 (Message Complexity).

The message complexity of an algorithm is determined by the total number of messages exchanged

Theorem 2.4 (Broadcast Lower Bound).

The **message complexity** of broadcast is **at least** $n-1$. The **sources radius** is a lower bound for the **time complexity**.

- You can use a pre-computed **spanning tree** to do broadcast with tight message complexity. If the spanning tree is a breadth-first search spanning tree (for a given source), then the time complexity is tight as well.



Clean Broadcast

Definition 2.5 (Clean).

A graph (network) is clean if the nodes do not know the topology of the graph.

Clean Broadcast

Definition 2.5 (Clean).

A graph (network) is clean if the nodes do not know the topology of the graph.

Definition 2.6 (Clean Broadcast Lower Bound).

For a clean network, the number of edges m is a lower bound for the broadcast message complexity.

Clean Broadcast

Definition 2.5 (Clean).

A graph (network) is clean if the nodes do not know the topology of the graph.

Definition 2.6 (Clean Broadcast Lower Bound).

For a clean network, the number of edges m is a lower bound for the broadcast message complexity.

Proof: If you do not try every edge, you might miss a whole part of the graph behind it.

Asynchronous Distributed Algorithms

Definition 2.7 (Asynchronous Distributed Algorithm).

In the asynchronous model, algorithms are event driven (upon receiving message . . . , do . . .). Nodes cannot access a global clock. A message sent from one node to another will arrive in finite but unbounded time.

Asynchronous Distributed Algorithms

Definition 2.7 (Asynchronous Distributed Algorithm).

In the asynchronous model, algorithms are event driven (upon receiving message . . . , do . . .). Nodes cannot access a global clock. A message sent from one node to another will arrive in finite but unbounded time.

- The asynchronous model and the synchronous model (Definition 1.8) are the cornerstone models in distributed computing.
- Note that in the asynchronous model, messages that take a longer path may arrive earlier.

Asynchronous Time Complexity

Definition 2.8 (Asynchronous Time Complexity).

For asynchronous algorithms (as defined in 2.7) the time complexity is **the number of time units** from the start of the execution to its completion **in the worst case** (every legal input, every execution scenario), assuming that each message has a delay of at most one time unit.

- The clean broadcast lower bound (Theorem 2.6) directly brings us to the well known flooding algorithm.

Flooding

ALGORITHM 2.9 FLOODING()

1: The source (root) sends the message to all neighbors.
2: **Each other node** v upon receiving the message the first time forwards the message to all (other) neighbors.
3: Upon later receiving the message agian (over other edges), a node can discard the message.

Flooding

ALGORITHM 2.9 FLOODING()

1: The source (root) sends the message to all neighbors.
2: **Each other node** v upon receiving the message the first time forwards the message to all (other) neighbors.
3: Upon later receiving the message agian (over other edges), a node can discard the message.

- If node v receives the message first from node u, then node v calls node u parent. **This parent relation defines a spanning tree** T. If the flooding algorithm is executed in a **synchronous system**, then T is a breadth-first search spanning tree (with respect to the root).

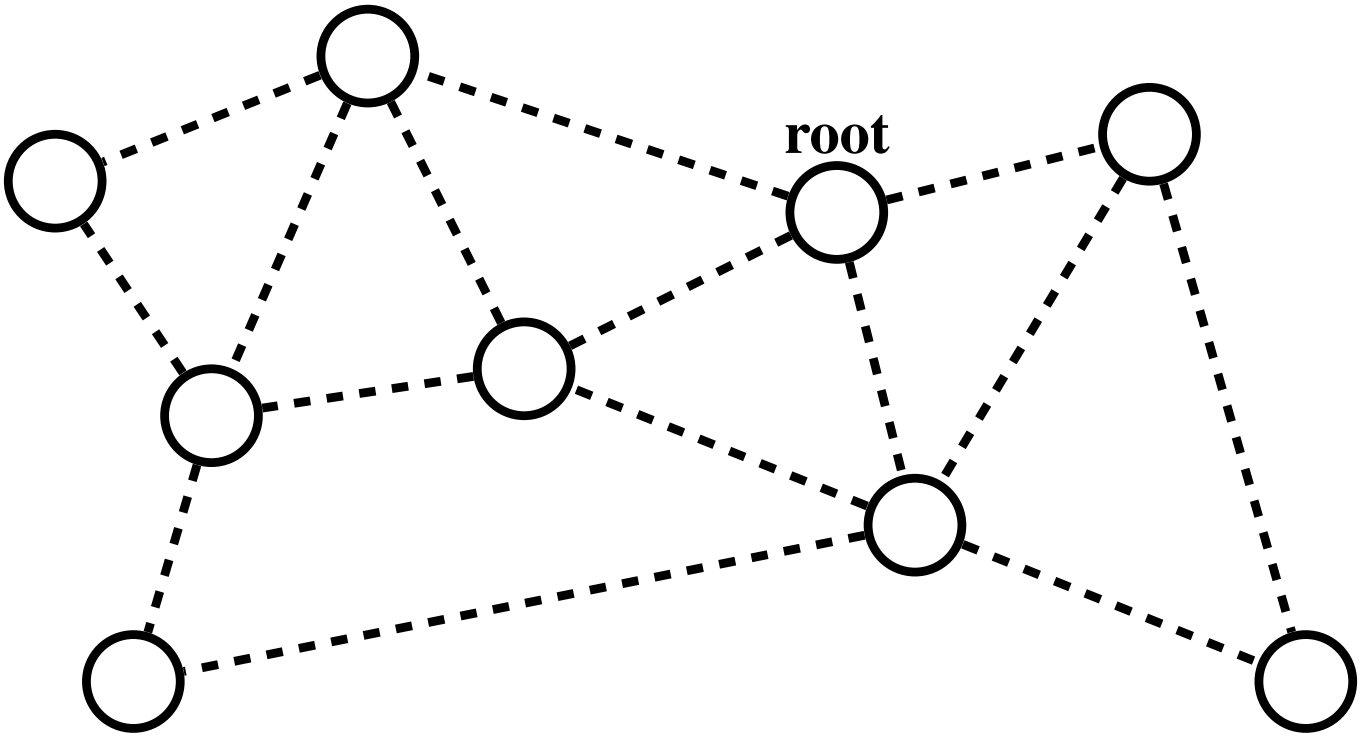
Flooding

ALGORITHM 2.9 FLOODING()

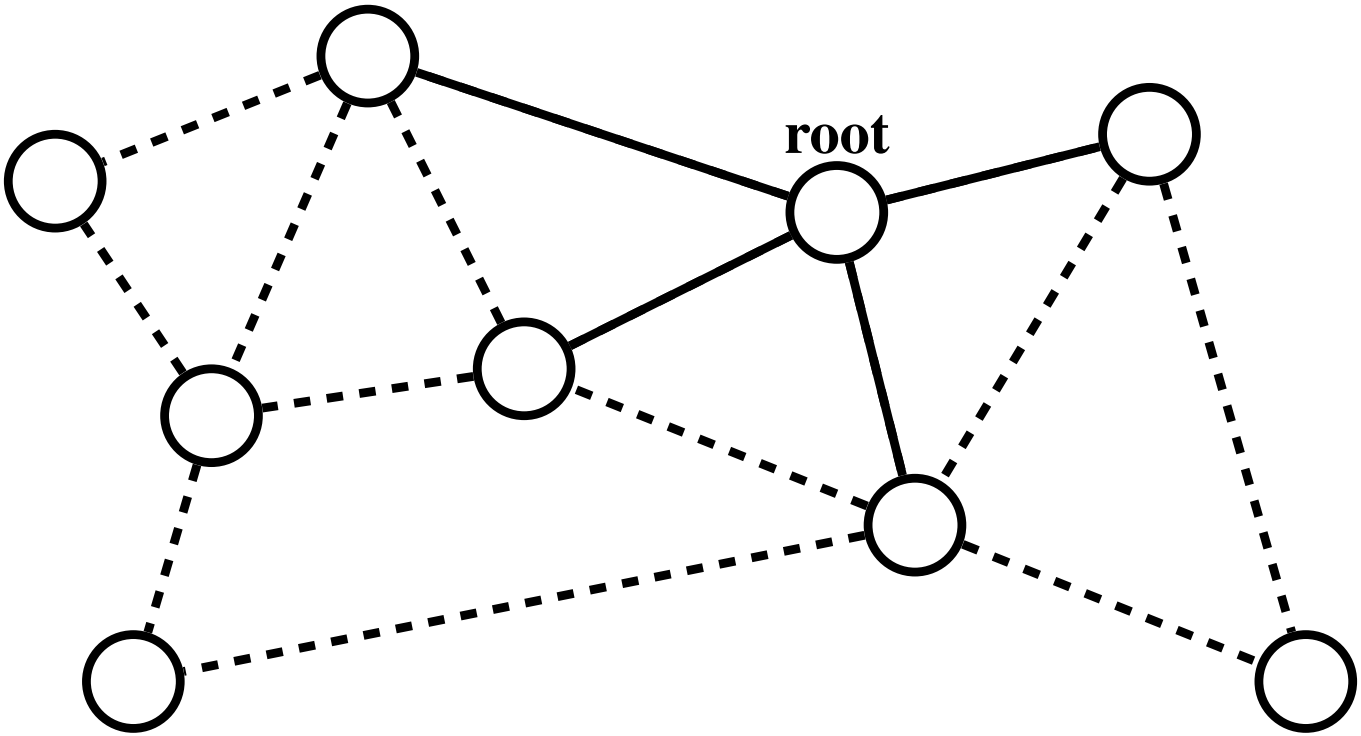
1: The source (root) sends the message to all neighbors.
2: **Each other node** v upon receiving the message the first time forwards the message to all (other) neighbors.
3: Upon later receiving the message agian (over other edges), a node can discard the message.

- If node v receives the message first from node u, then node v calls node u parent. **This parent relation defines a spanning tree** T. If the flooding algorithm is executed in a **synchronous system**, then T is a breadth-first search spanning tree (with respect to the root).
- More interestingly, also in **asynchronous systems** the flooding algorithm terminates after R time units, R being the radius of the source. However, the constructed spanning tree **may not** be a breadth-first search spanning tree.

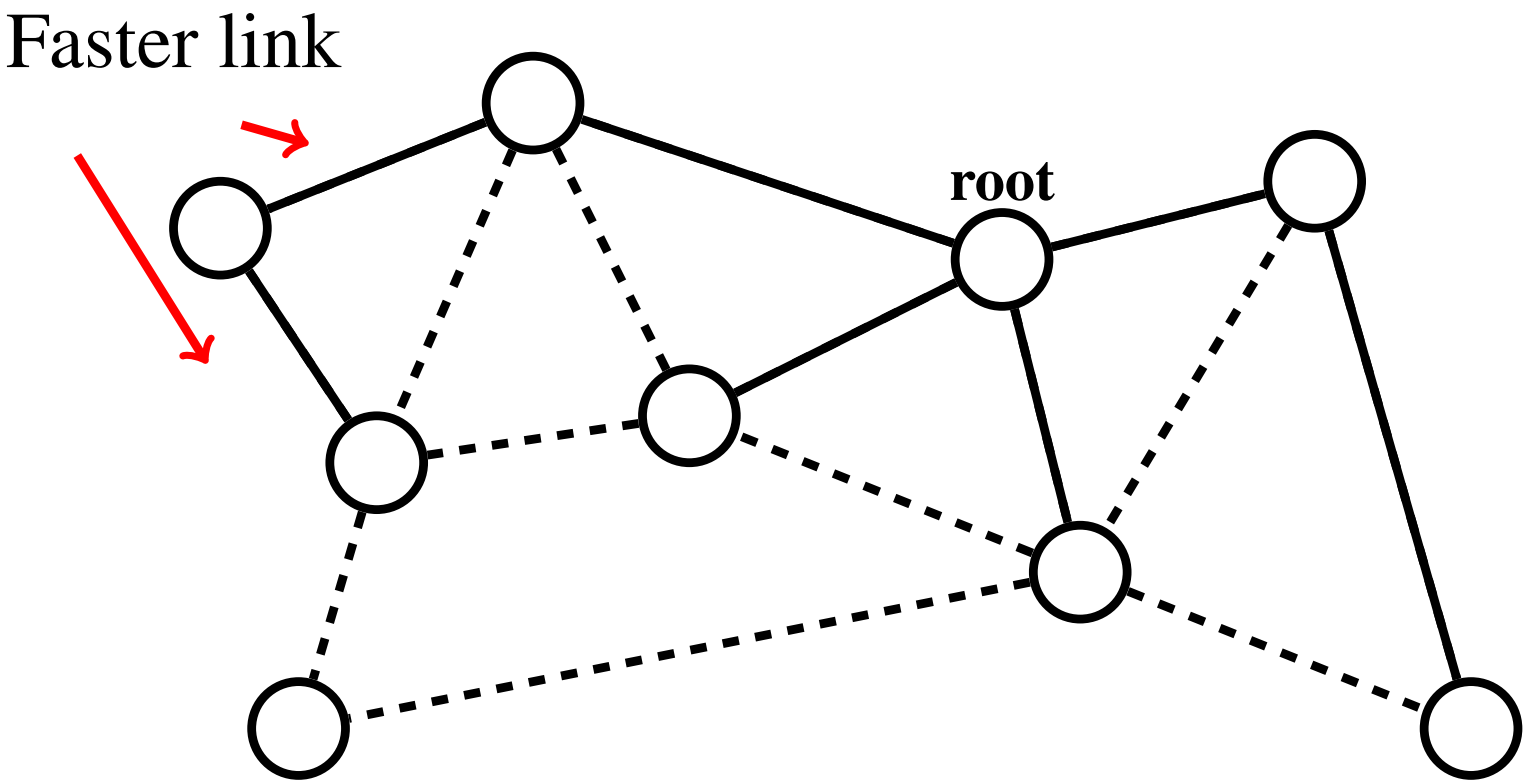
Example



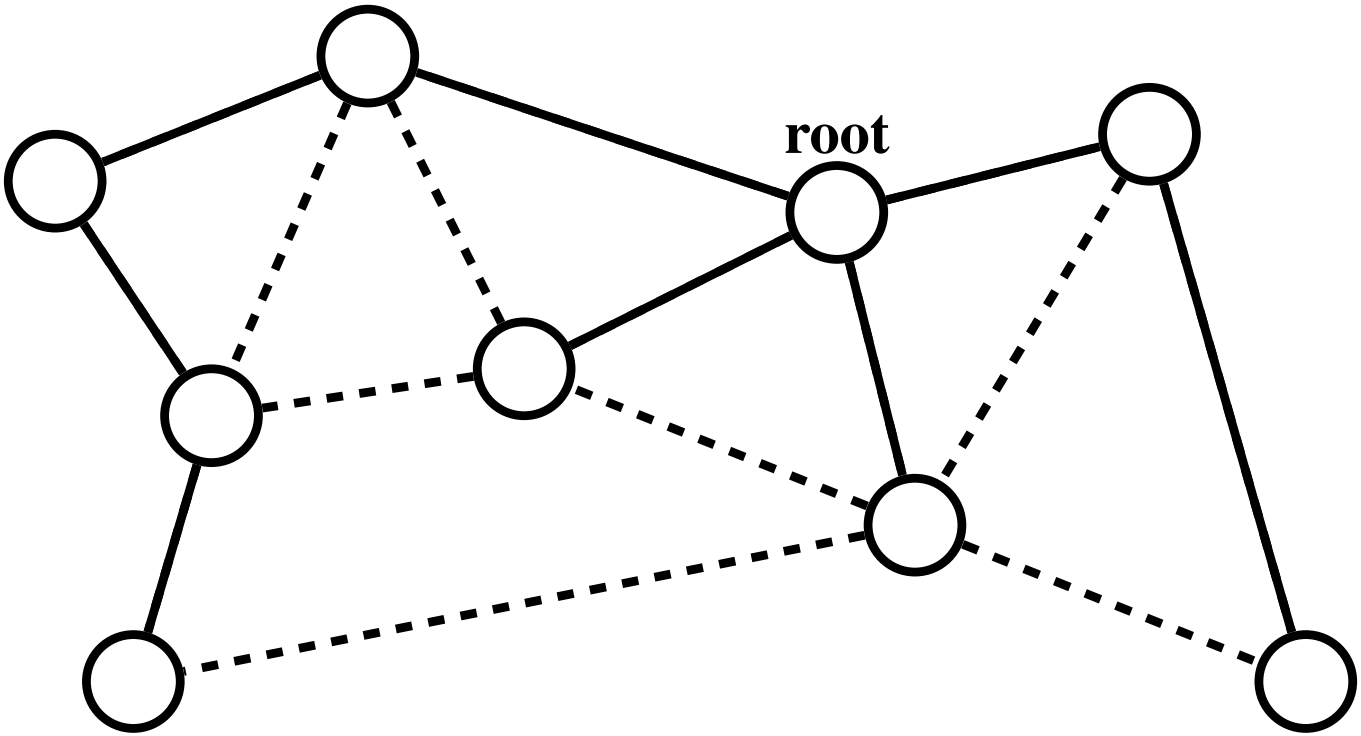
Example



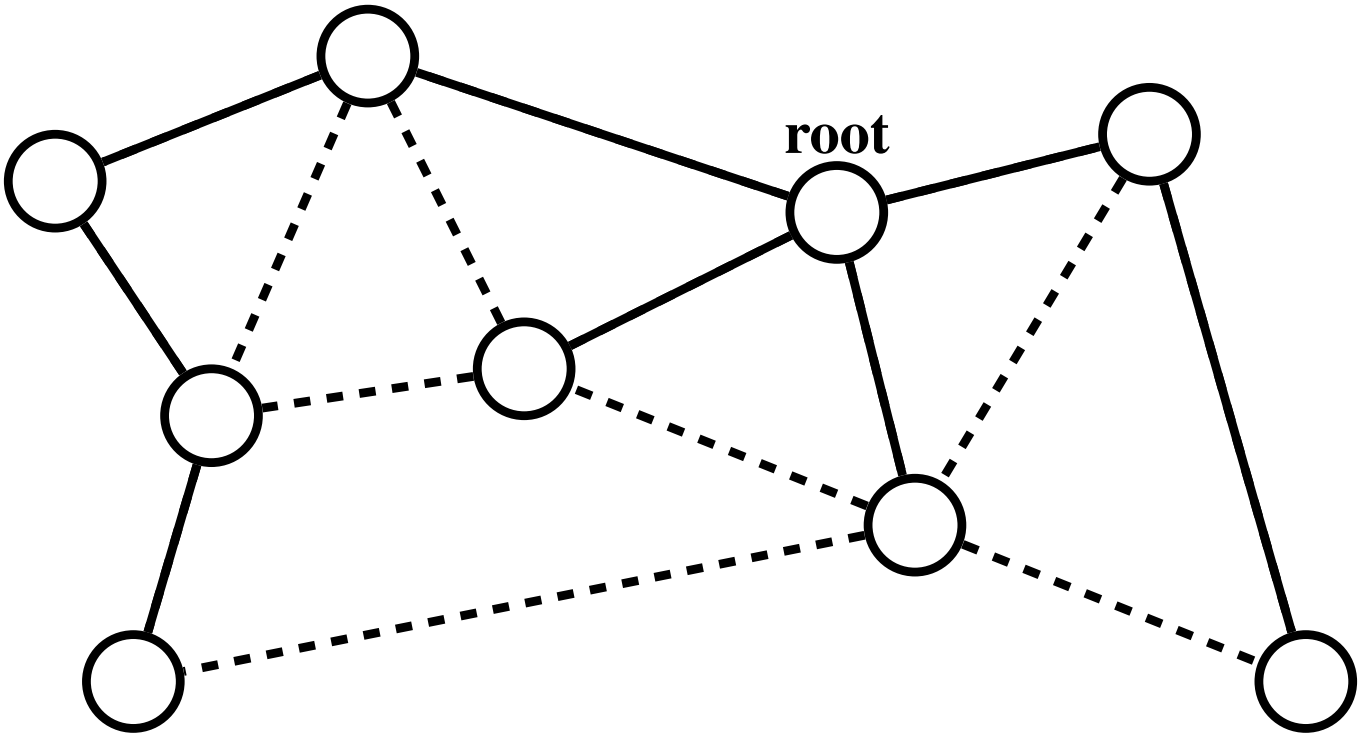
Example



Example



Example



Not a BFS spaning tree!

Convergecast

Convergecast is the same as broadcast, just reversed: Instead of a root sending a message to all other nodes, all other nodes send information to a root (starting from the leaves, *i*.*e*., the tree T is known). The simplest convergecast algorithm is the echo algorithm:

Convergecast

Convergecast is the same as broadcast, just reversed: Instead of a root sending a message to all other nodes, all other nodes send information to a root (starting from the leaves, *i.e.*, the tree T is known). The simplest convergecast algorithm is the echo algorithm:

ALGORITHM 2.10 ECHO()

1: A leave sends a message to its parent.
2: If an inner node has received a message from each child, it sends a message to the parent

Convergecast

Convergecast is the same as broadcast, just reversed: Instead of a root sending a message to all other nodes, all other nodes send information to a root (starting from the leaves, *i.e.*, the tree T is known). The simplest convergecast algorithm is the echo algorithm:

ALGORITHM 2.10 ECHO()

1: A leave sends a message to its parent.
2: If an inner node has received a message from each child, it sends a message to the parent

- Usually the echo algorithm is paired with the flooding algorithm, which is used to let the leaves know that they should start the echo process; this is known as flooding/echo.

Convergecast

Convergecast is the same as broadcast, just reversed: Instead of a root sending a message to all other nodes, all other nodes send information to a root (starting from the leaves, *i.e.*, the tree T is known). The simplest convergecast algorithm is the echo algorithm:

ALGORITHM 2.10 ECHO()

1: A leave sends a message to its parent.
2: If an inner node has received a message from each child, it sends a message to the parent

- Usually the echo algorithm is paired with the flooding algorithm, which is used to let the leaves know that they should start the echo process; this is known as flooding/echo.
- One can use convergecast for termination detection, for example. If a root wants to know whether all nodes in the system have finished some task, it initiates a flooding/echo; the message in the echo algorithm then means This subtree has finished the task.

Convergecast

- Message complexity of the echo algorithm is $n-1$, but together with flooding it is $O(m)$, where $m = |E|$ is the number of edges in the graph.

Convergecast

- Message complexity of the echo algorithm is $n-1$, but together with flooding it is $O(m)$, where $m = |E|$ is the number of edges in the graph.
- The time complexity of the echo algorithm is determined by the depth of the spanning tree (*i.e.*, the radius of the root within the tree) generated by the flooding algorithm.

Convergecast

- Message complexity of the echo algorithm is $n-1$, but together with flooding it is $O(m)$, where $m = |E|$ is the number of edges in the graph.
- The time complexity of the echo algorithm is determined by the depth of the spanning tree (*i.e.*, the radius of the root within the tree) generated by the flooding algorithm.
- The flooding/echo algorithm can do much more than collecting acknowledgements from subtrees. One can for instance use it to compute the number of nodes in the system, or the maximum ID, or the sum of all values stored in the system, or a route-disjoint matching.

Convergecast

- Message complexity of the echo algorithm is $n-1$, but together with flooding it is $O(m)$, where $m = |E|$ is the number of edges in the graph.
- The time complexity of the echo algorithm is determined by the depth of the spanning tree (*i.e.*, the radius of the root within the tree) generated by the flooding algorithm.
- The flooding/echo algorithm can do much more than collecting acknowledgements from subtrees. One can for instance use it to compute the number of nodes in the system, or the maximum ID, or the sum of all values stored in the system, or a route-disjoint matching.
- Moreover, by combining results one can compute even fancier aggregations,*e.g.*, with the number of nodes and the sum one can compute the average. With the average one can compute the standard deviation.And so on ...

BFS Tree Construction

- In synchronous systems the flooding algorithm is a simple yet efficient method to construct a breadth-first search (BFS) spanning tree.

BFS Tree Construction

- In synchronous systems the flooding algorithm is a simple yet efficient method to construct a breadth-first search (BFS) spanning tree.
- How do we construct a BFS tree in asynchronous systems?

BFS Tree Construction

- In synchronous systems the flooding algorithm is a simple yet efficient method to construct a breadth-first search (BFS) spanning tree.
- How do we construct a BFS tree in asynchronous systems?
- Two classic BFS constructions:

BFS Tree Construction

- In synchronous systems the flooding algorithm is a simple yet efficient method to construct a breadth-first search (BFS) spanning tree.
- How do we construct a BFS tree in asynchronous systems?
- Two classic BFS constructions:
  - Dijkstra and Bellman-Ford

Dijkstra BFS

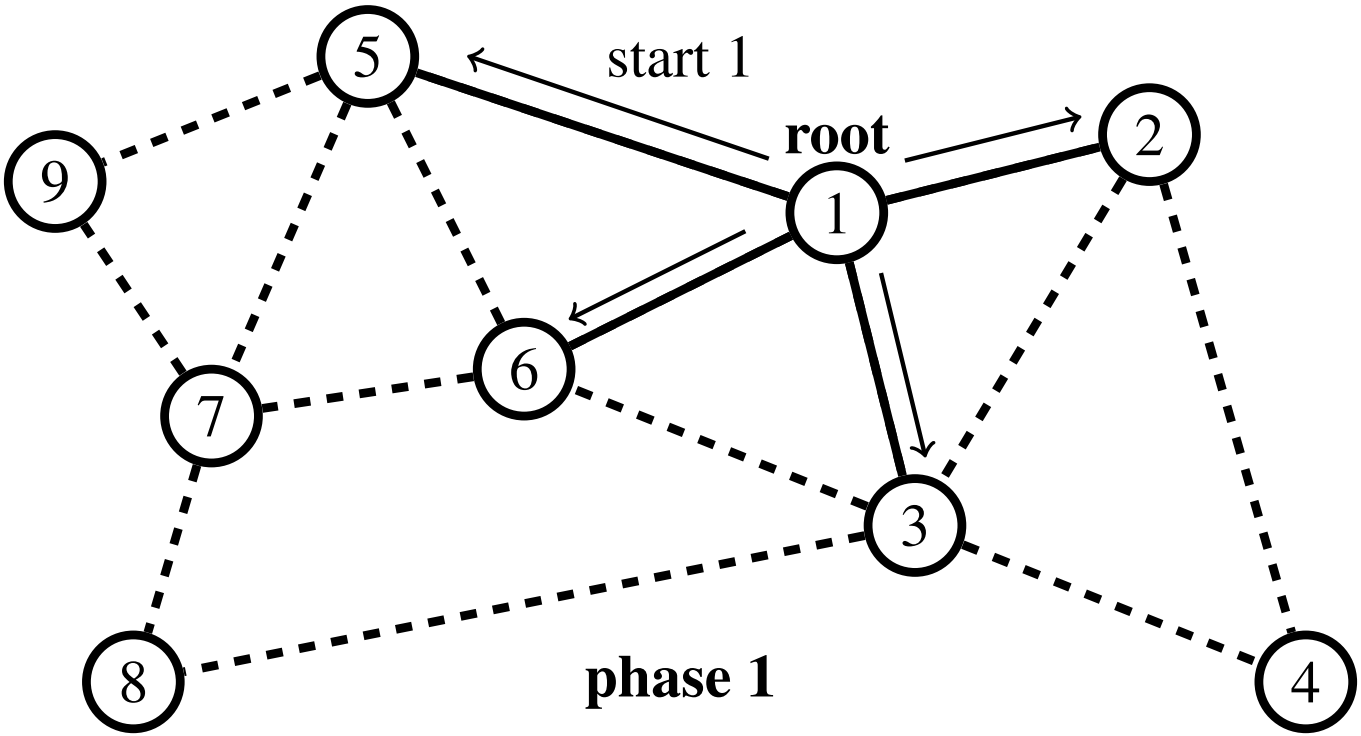
Basic idea:

- always adding the closest node to the existing part of the BFS tree.
- developing the BFS tree layer by layer.
- The algorithm proceeds in phases. In phase p the nodes with distance p to the root are detected. Let T_p be the tree in phase p.

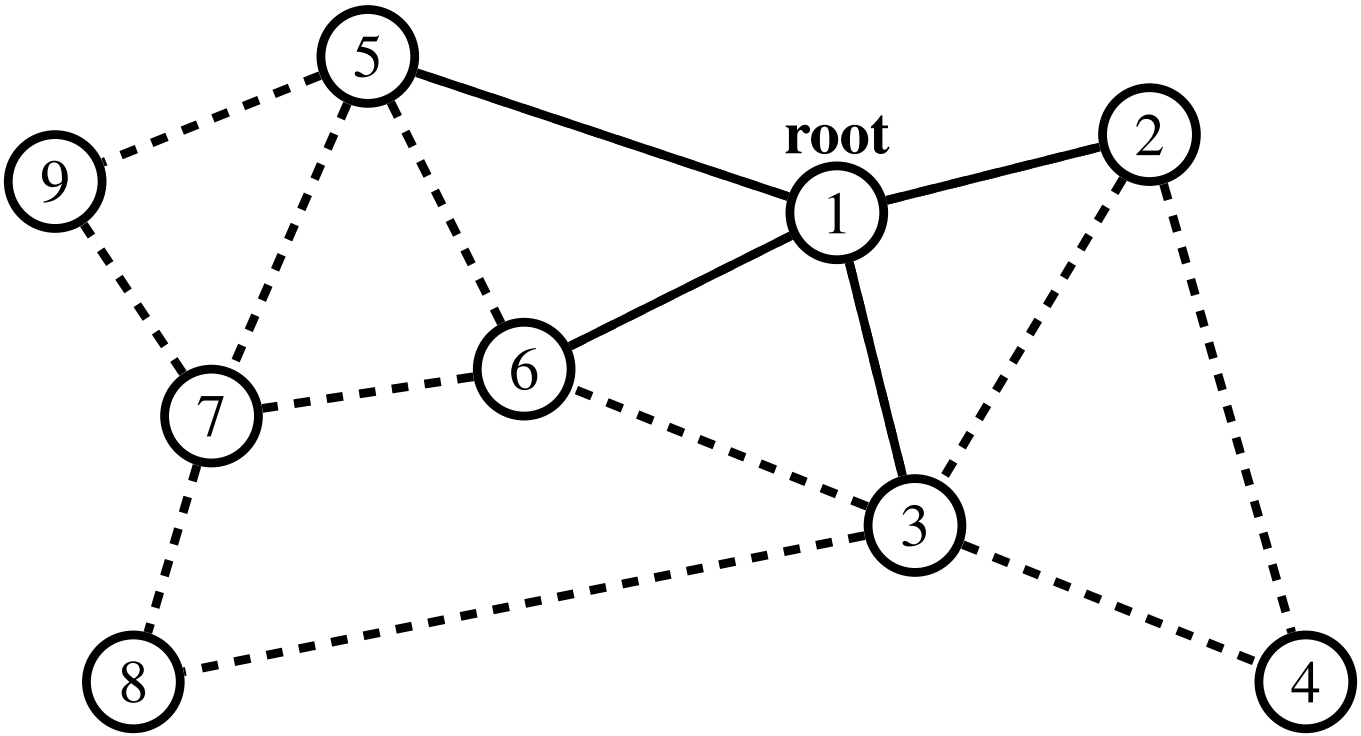
ALGORITHM 2.11 DIJKSTRA BFS()

We start with T_1 which is the root plus all direct neighbors of the root. We start with phase $p = 1$:
repeat
    The root starts phase p by broadcasting start p within T_p.
    When receiving start p a leaf node u of T_p (that is, a node that was newly discovered in the last phase) sends a join $p+1$ message to all quiet neighbors. (A neighbor v is quiet if u has not yet talked to v.)
    A node v receiving the first join $p+1$ message replies with ACK and becomes a leaf of the tree T_{p+1}.
    A node v receiving any further join message replies with NACK.
    The leaves of T_p collect all the answers of their neighbors; then the leaves start an echo algorithm back to the root.
    When the echo process terminates at the root, the root increments the phase
until there was no new node detected

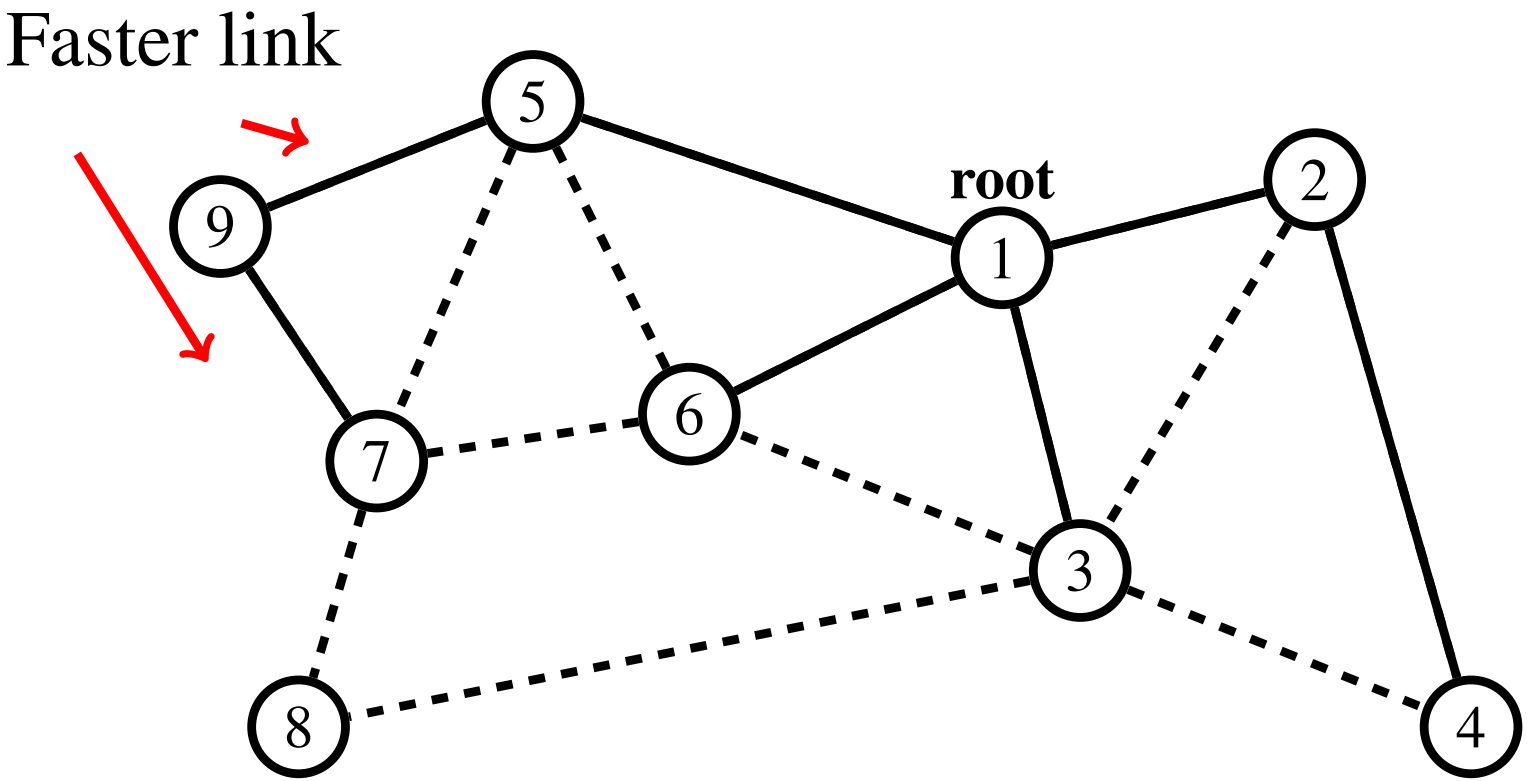
Example



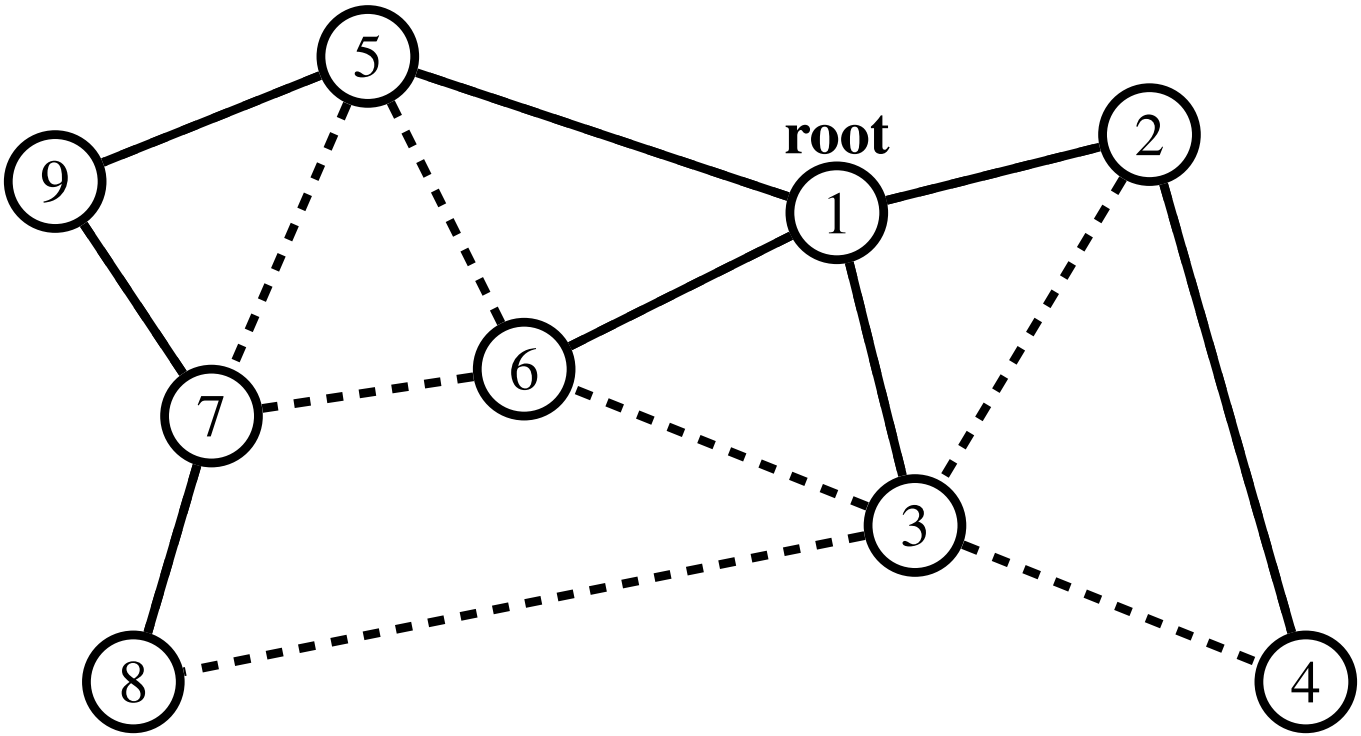
Example



Example



Example



Example