Universität Rostock

Traditio et Innovatio

Distributed Algorithms

Flooding, Broadcast and Echo

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

Architecture of Application Systems (AVA)
Faculty for Informatics and Electrical Engineering
University of Rostock

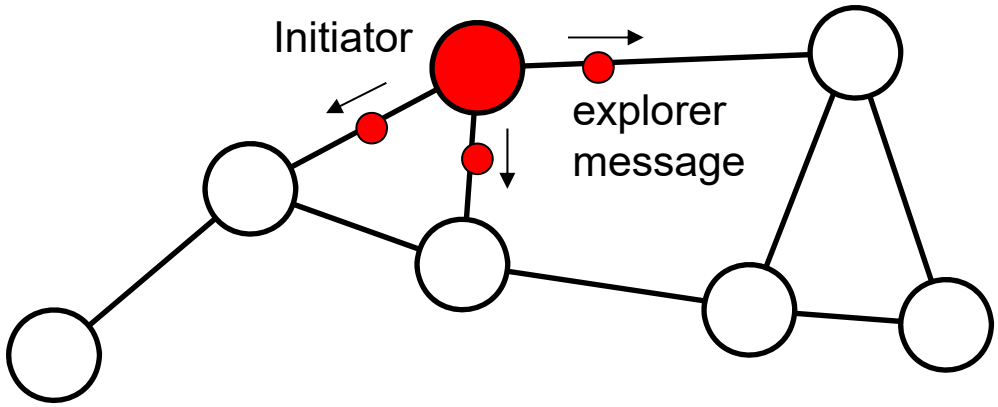
Overview

- Flooding
  - Distribution of information from a node to all nodes using *all edges* of the topology with or without confirmation
- Echo
  - Distribution of information from a node to all nodes using all edges and confirmation using only a *subset of edges*
  - Further applications
    - Collecting information
    - Construction of a spanning tree
- Broadcast
  - Distribution of information to all nodes on special topologies with or without confirmation

Flooding

Information Distribution with Flooding

- Precondition: connected topology
- Initiator starts spreading a *new* information by sending it to all its neighbor nodes
- A node that receives a *new* information from one of its neighbors, forwards it to all other neighbors
- Already known information is ignored
- Step by step all nodes will be informed



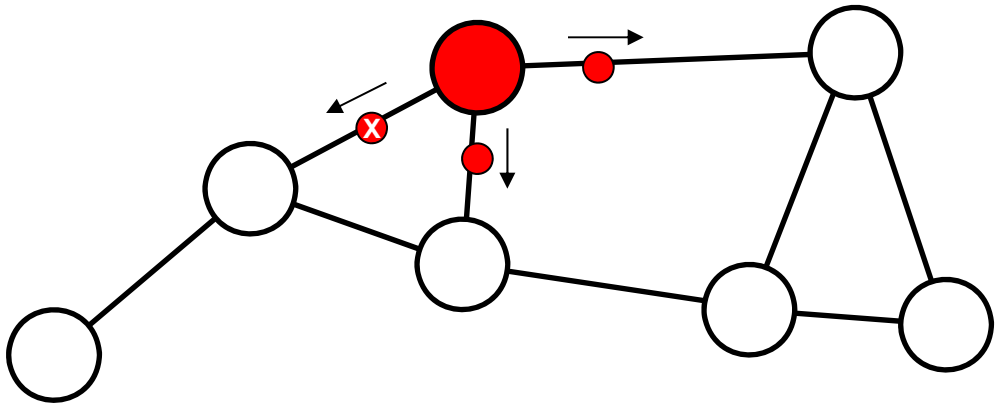
Flooding Algorithm

```
I:{NOT informed}
   SEND <info> TO all neighbors
   informed := TRUE;

R:{a message <info> is received}
   IF (NOT informed) THEN
       SEND <info> TO all other neighbors;
       informed := TRUE;
   FI
```

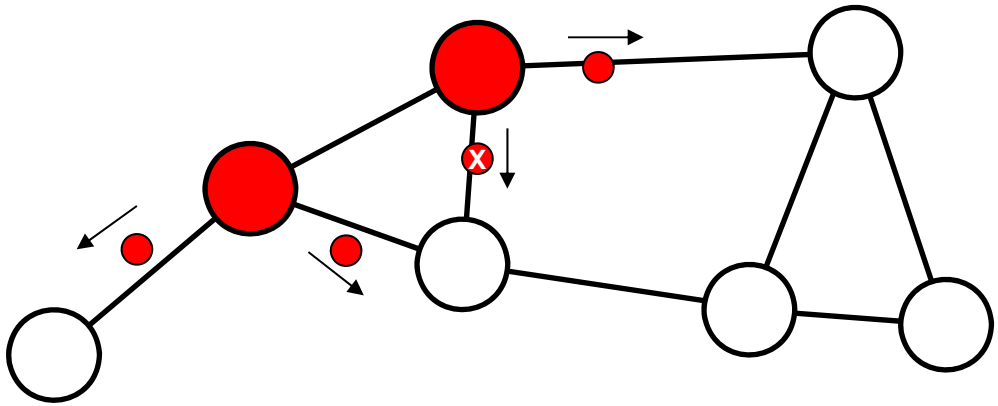
- Initially, **`informed == FALSE`** holds for all processes
- Action **`I`** is carried out by the initiator spontaneously

Flooding – Example



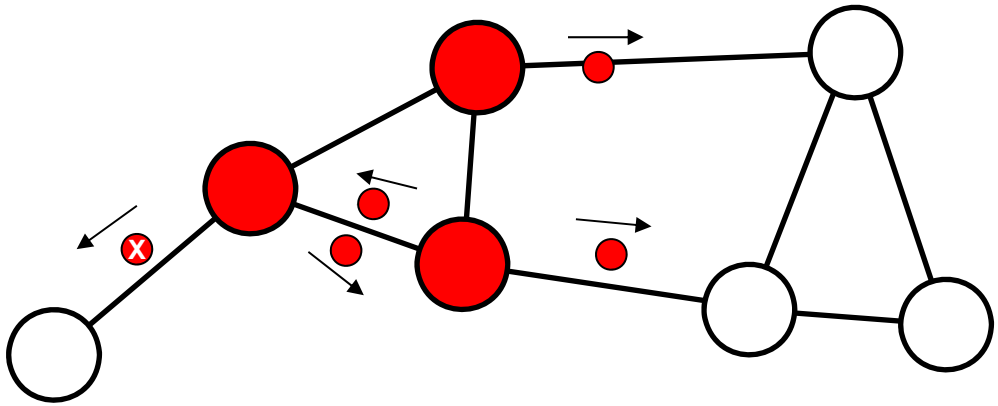
3 messages sent

Flooding – Example



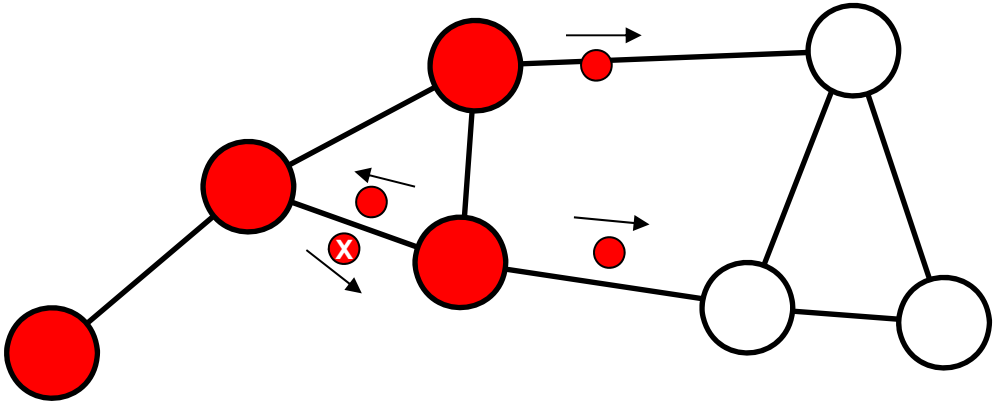
5 messages sent

Flooding – Example



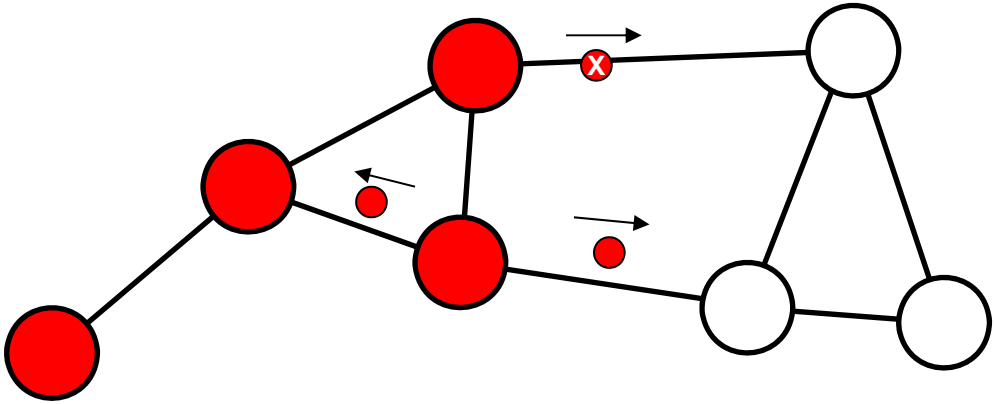
7 messages sent

Flooding – Example



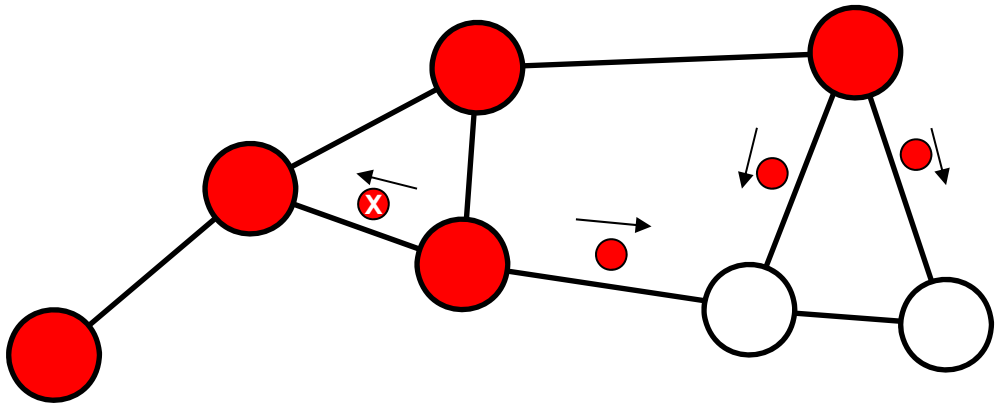
7 messages sent

Flooding – Example



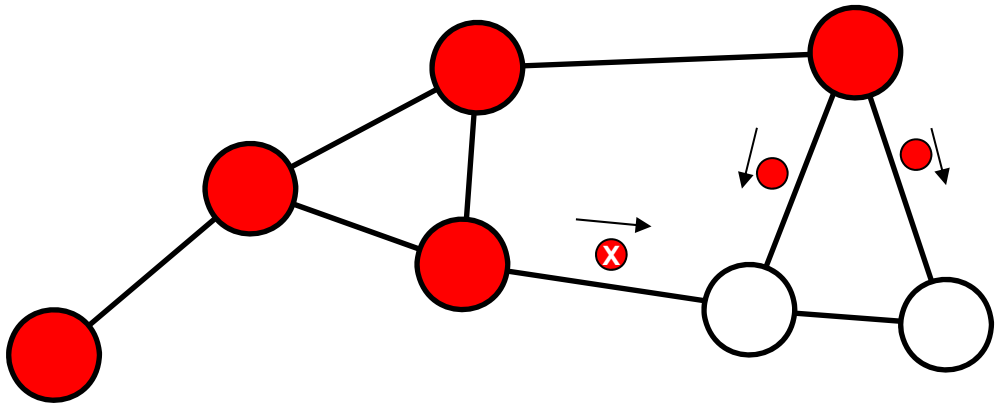
7 messages sent

Flooding – Example



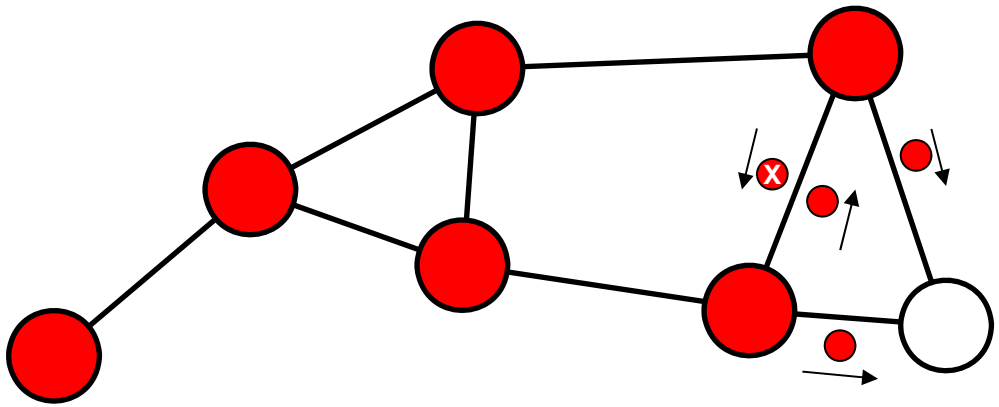
9 messages sent

Flooding – Example



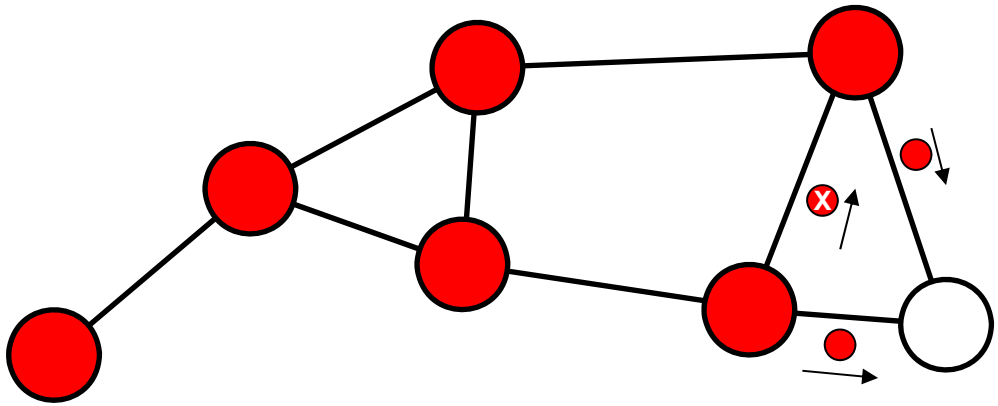
9 messages sent

Flooding – Example



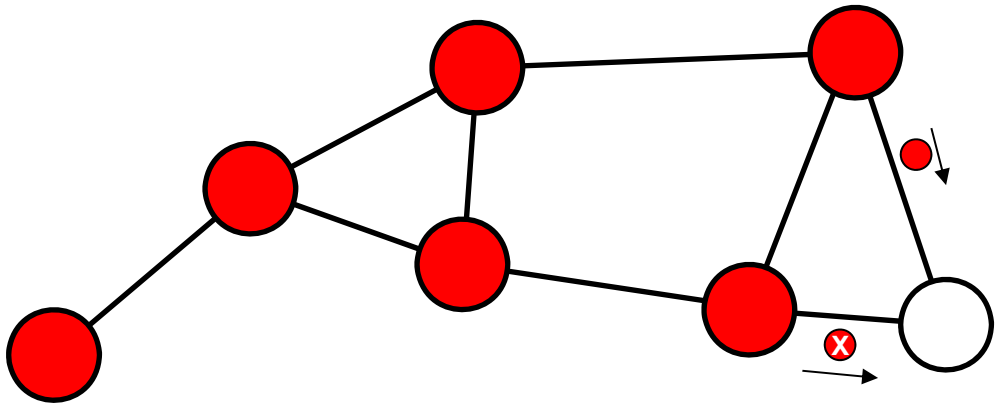
11 messages sent

Flooding – Example



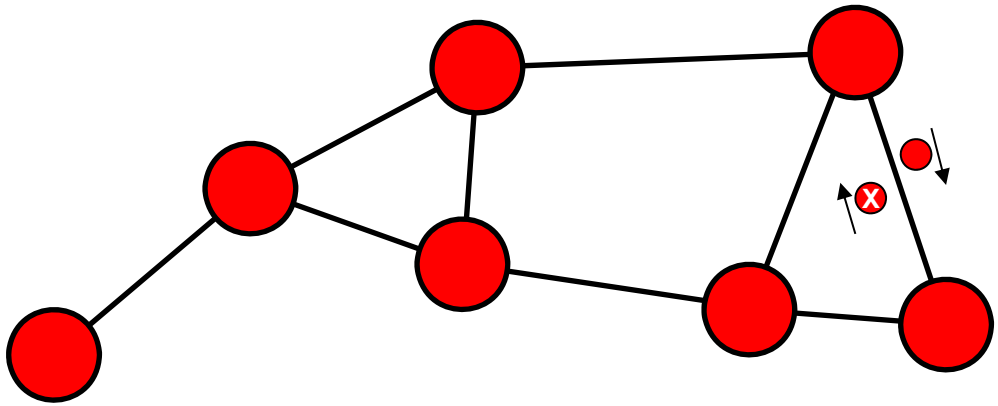
11 messages sent

Flooding – Example



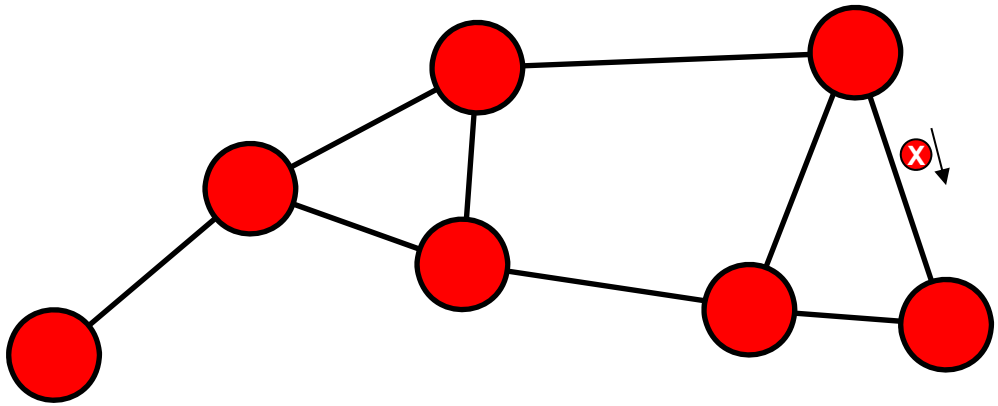
11 messages sent

Flooding – Example



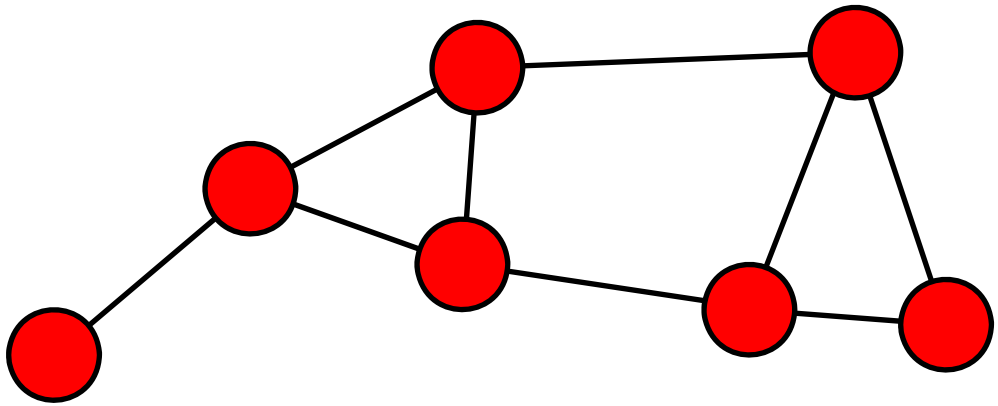
12 messages sent

Flooding – Example



12 messages sent

Flooding – Example



12 messages sent

Message Complexity of Flooding

- How many messages are sent altogether?
  - Let n be the number of nodes and e the number of edges
  - Each node sends over all his incident edges

    → $+2e$ messages
  - But not back over its activation edge

    → $-n$ messages
  - Exception: initiator (has no activation edge)

    → +1 message
  - ⇒ Altogether $2e - n + 1$ messages ($2 \cdot 9 - 7 + 1 = 12$)
- How does the initiator know that all nodes were reached?

Flooding with Confirmation

- Two message types explorers and confirmations
- Only the first explorer is acknowledged
  - A node that sends out *x* explorers when it receives the first explorer, acknowledges this explorer if it has received *x* confirmations from its other neighbors
  - Leaves acknowledge instantly as they have only one neighbor
- Algorithm terminates, when the initiator has received a confirmation from all its neighbors

Flooding with Confirmation

Initially, `informed == false` and `count == 0` for all nodes

```
I: {NOT informed} // executed by initiator
   SEND explorer TO all neighbors;
   informed := TRUE;

{explorer from neighbor N is received}
   IF (NOT informed) THEN
      informed := TRUE;
      SEND explorer TO all neighbors except N;
      A := N; // remember activation edge
      IF (#neighbors == 1) THEN // node is leaf
         SEND confirmation to neighbor A;
      FI
   ELSE
      SEND confirmation TO N;
   FI
```

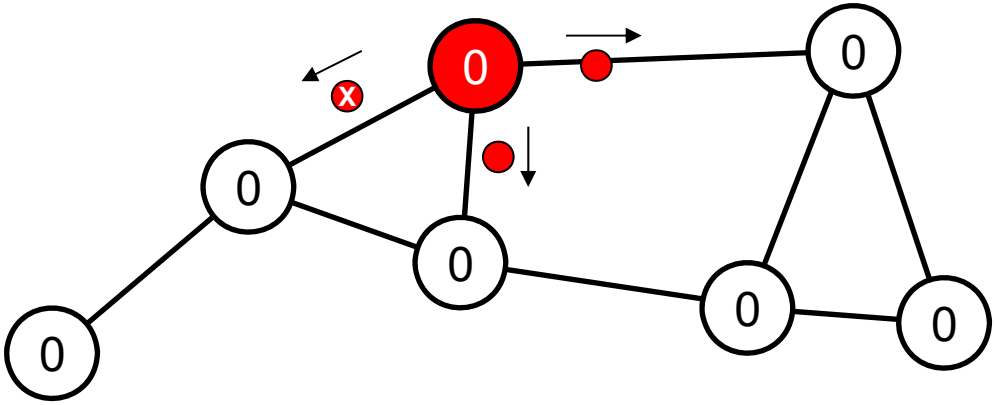
Flooding with Confirmation

Initially, `informed == false` and `count == 0` for all nodes

```
{confirmation is received}
  count++; // count confirmation
  IF (NOT initiator)
    IF (count == #neighbors - 1) THEN
      SEND confirmation TO neighbor A;
    FI
  ELSE IF (count == #neighbors) THEN
    EXIT; // algorithm terminated
  FI
```

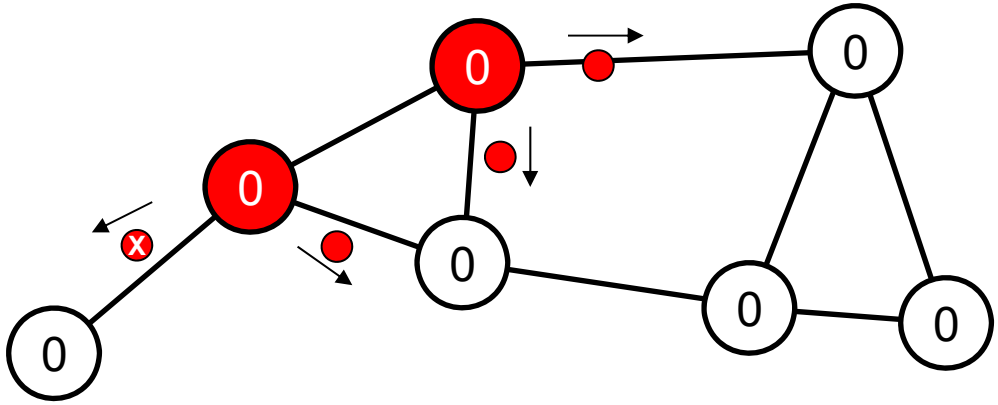
Flooding with Confirmation – Example

Here, the number of received confirmations is counted by each node.



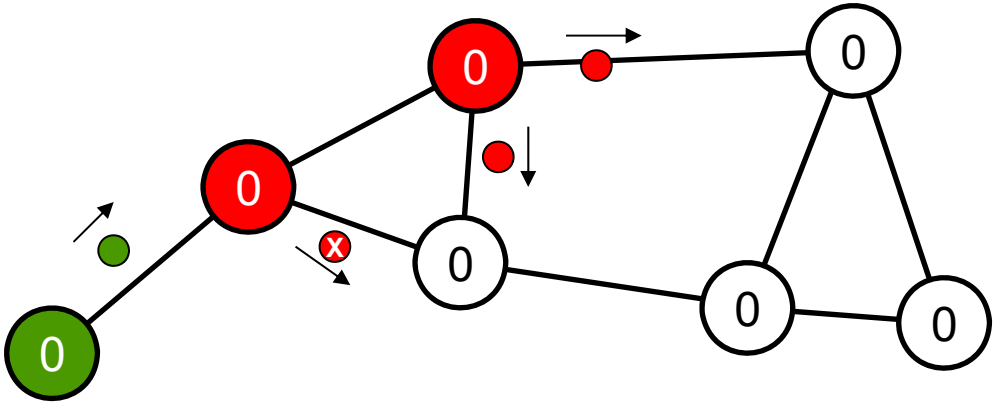
3 explorer sent
0 confirmations sent

Flooding with Confirmation – Example



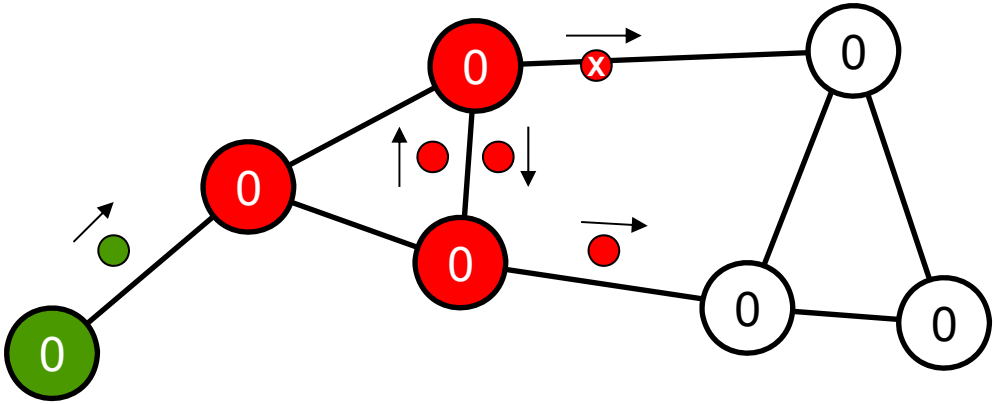
5 explorer sent
0 confirmations sent

Flooding with Confirmation – Example



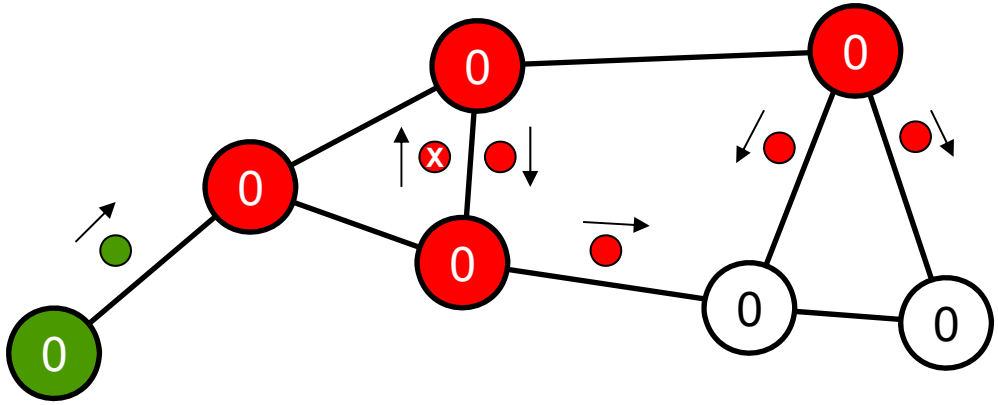
5 explorer sent
1 confirmations sent

Flooding with Confirmation – Example



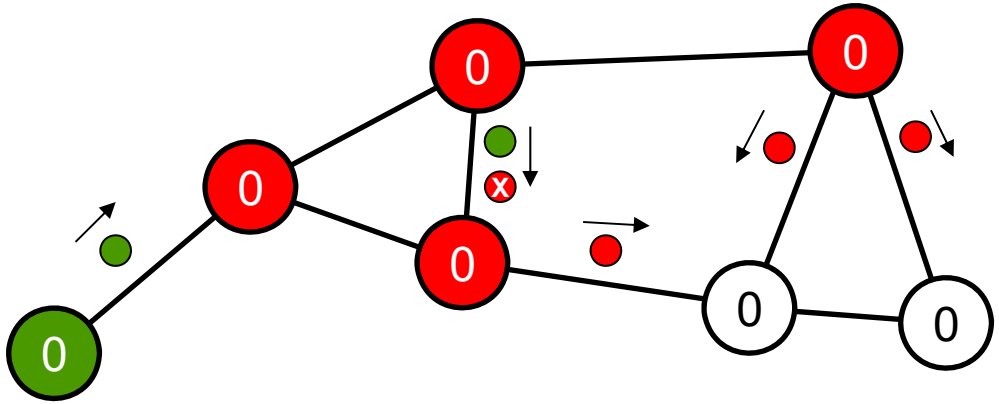
7 explorer sent
1 confirmations sent

Flooding with Confirmation – Example



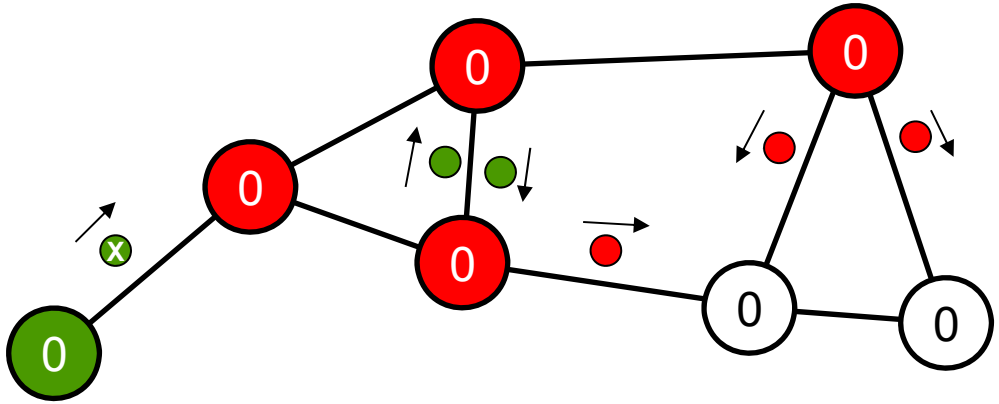
9 explorer sent
1 confirmations sent

Flooding with Confirmation – Example



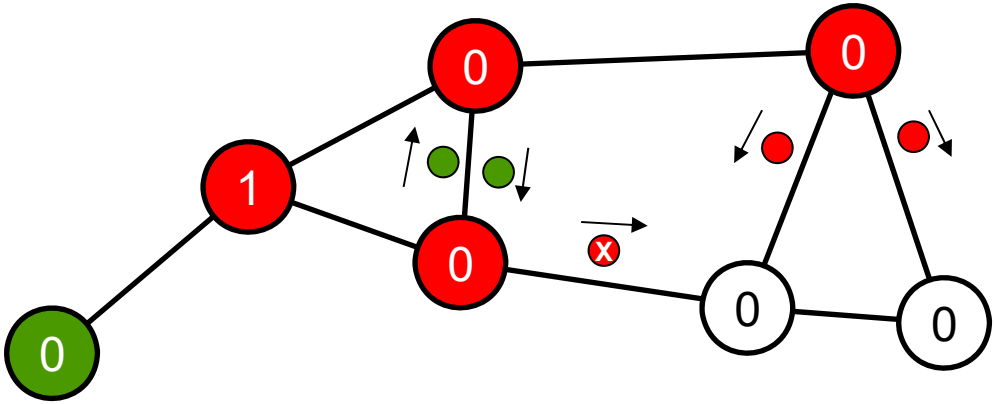
9 explorer sent
2 confirmations sent

Flooding with Confirmation – Example



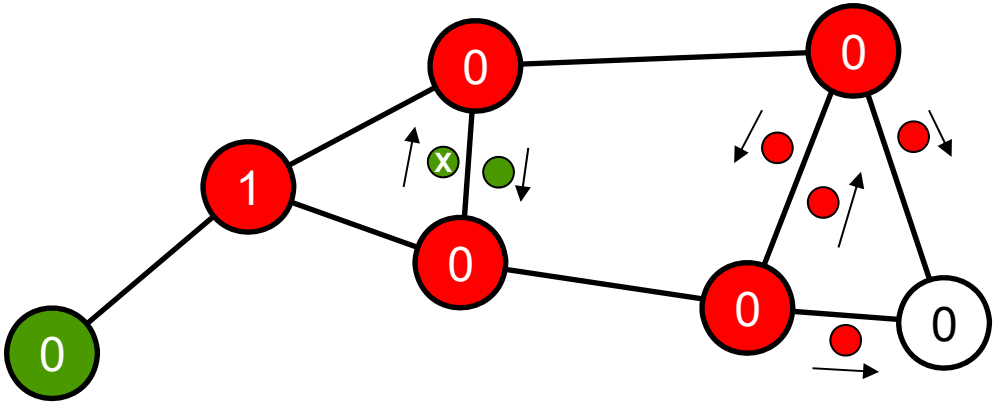
9 explorer sent
3 confirmations sent

Flooding with Confirmation – Example



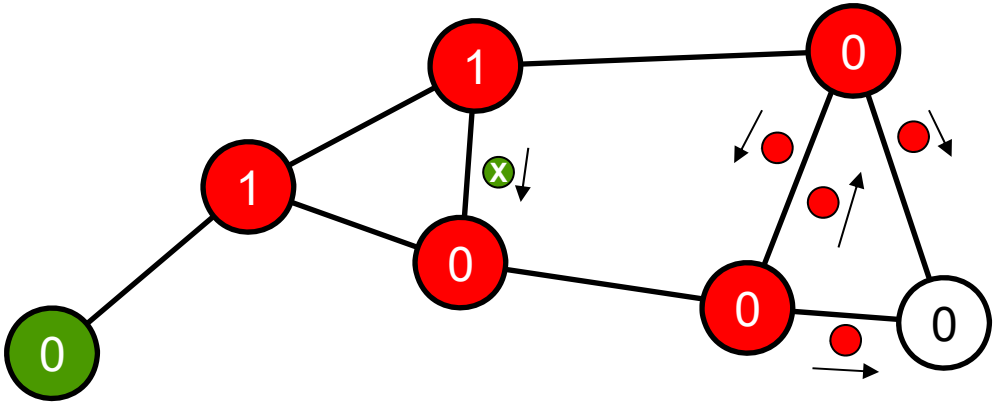
9 explorer sent
3 confirmations sent

Flooding with Confirmation – Example



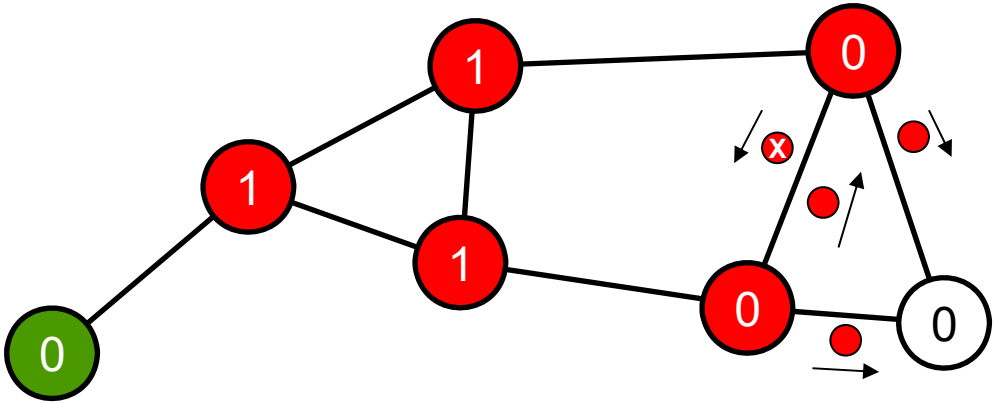
11 explorer sent
3 confirmations sent

Flooding with Confirmation – Example



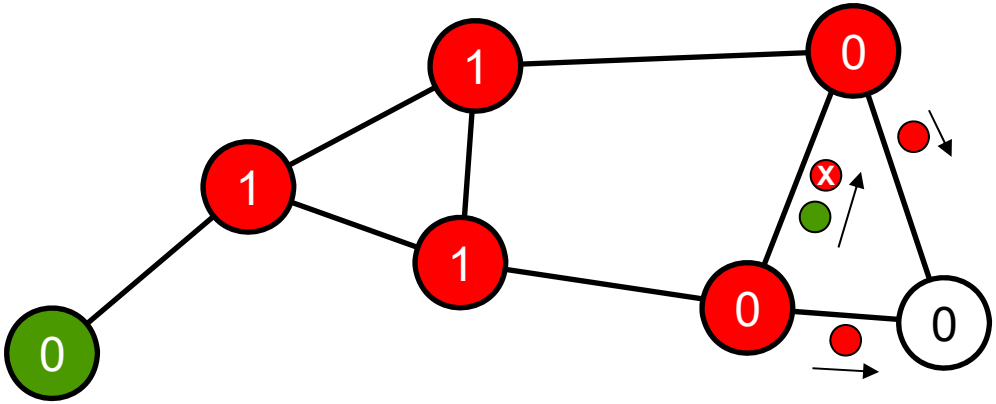
11 explorer sent
3 confirmations sent

Flooding with Confirmation – Example



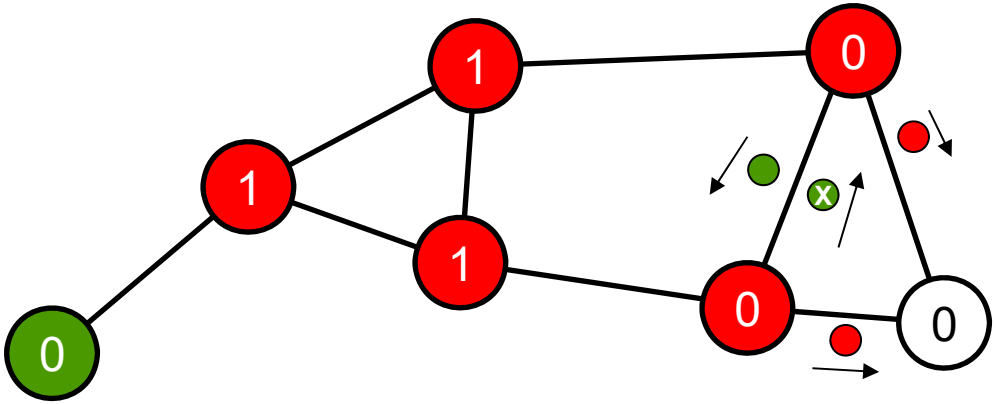
11 explorer sent
3 confirmations sent

Flooding with Confirmation – Example



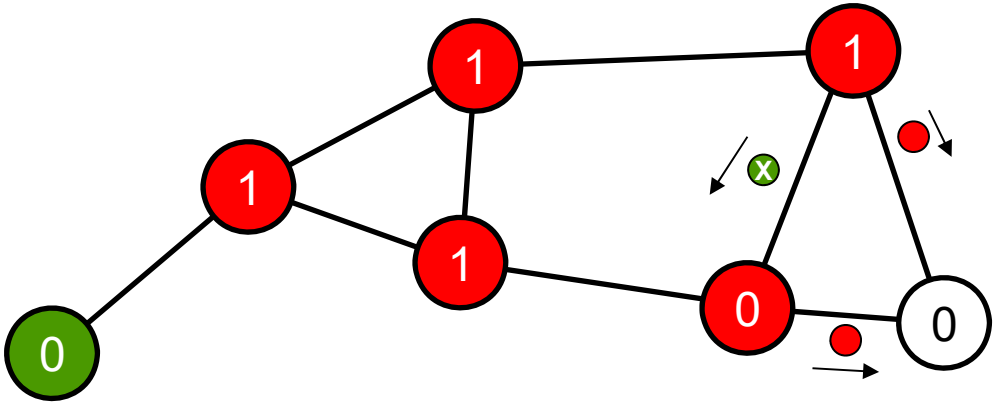
11 explorer sent
4 confirmations sent

Flooding with Confirmation – Example



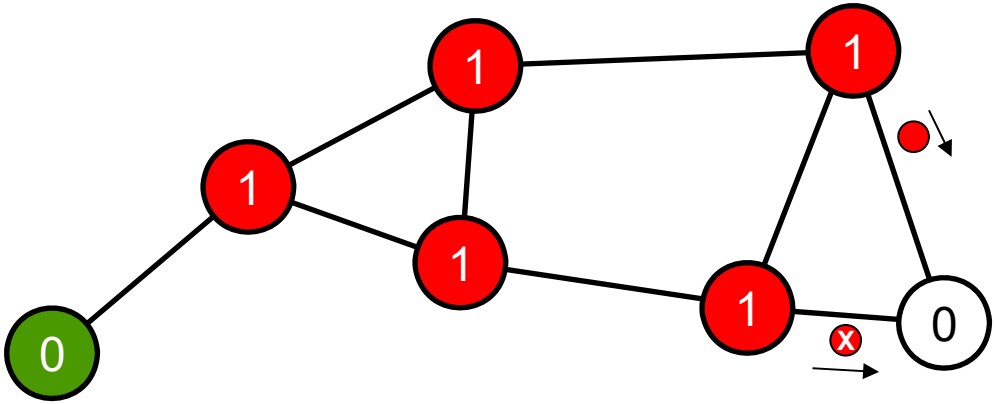
11 explorer sent
5 confirmations sent

Flooding with Confirmation – Example



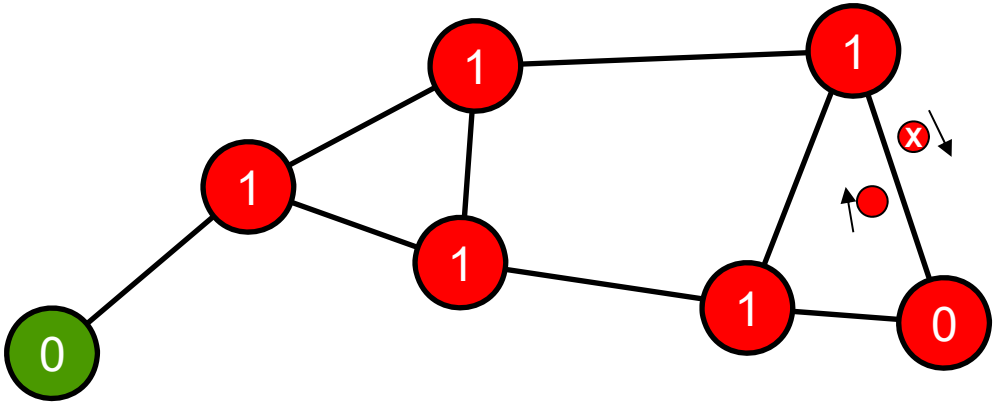
11 explorer sent
5 confirmations sent

Flooding with Confirmation – Example



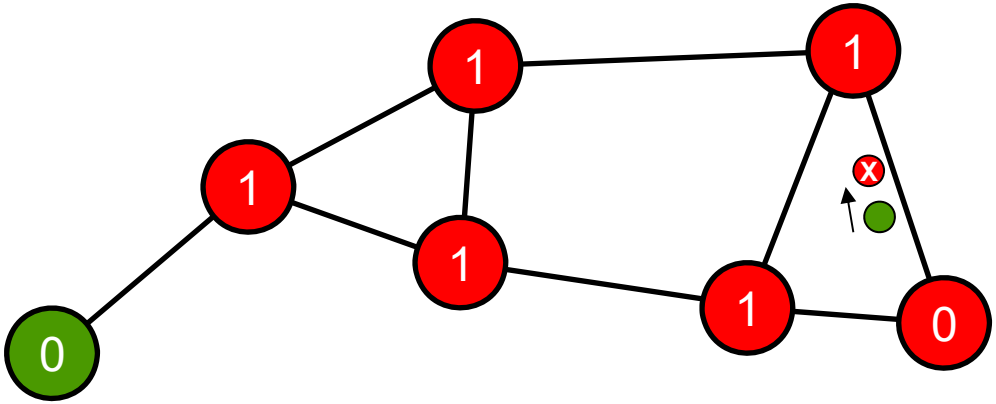
11 explorer sent
5 confirmations sent

Flooding with Confirmation – Example



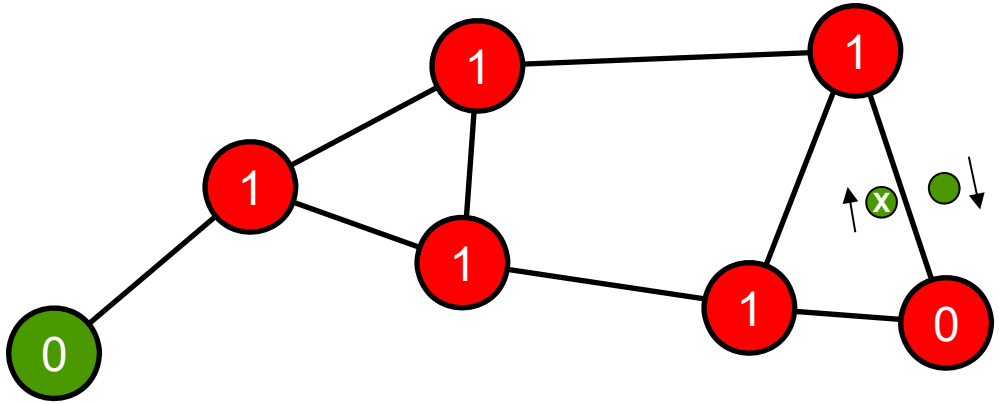
12 explorer sent
5 confirmations sent

Flooding with Confirmation – Example



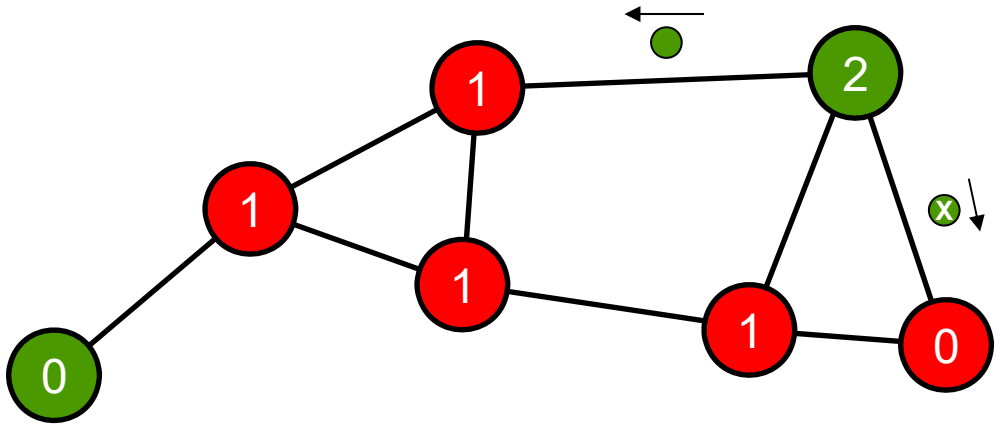
12 explorer sent
6 confirmations sent

Flooding with Confirmation – Example



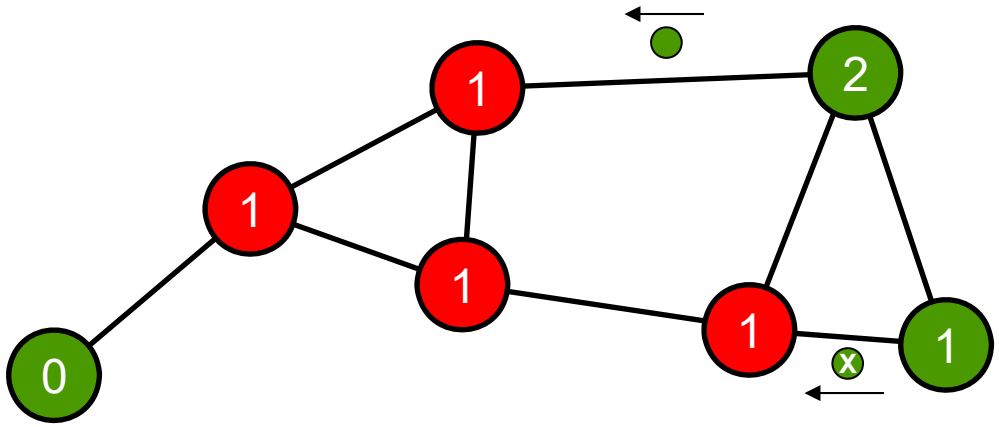
12 explorer sent
7 confirmations sent

Flooding with Confirmation – Example



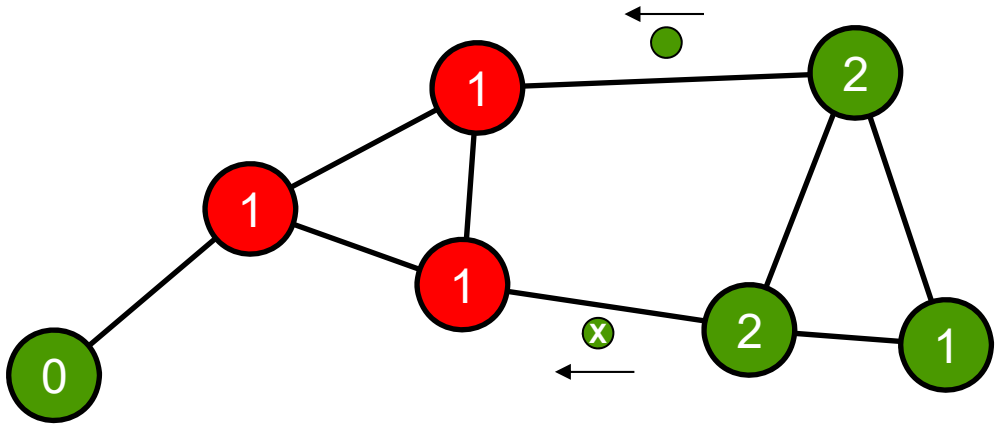
12 explorer sent
8 confirmations sent

Flooding with Confirmation – Example



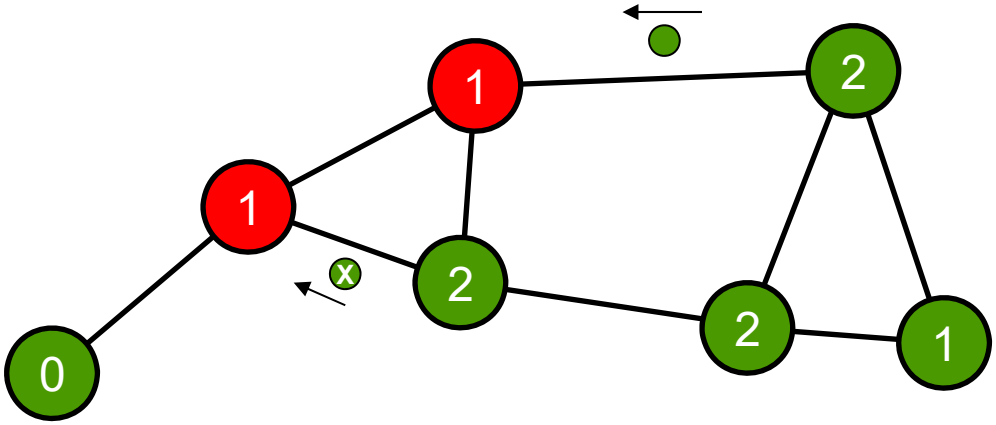
12 explorer sent
9 confirmations sent

Flooding with Confirmation – Example



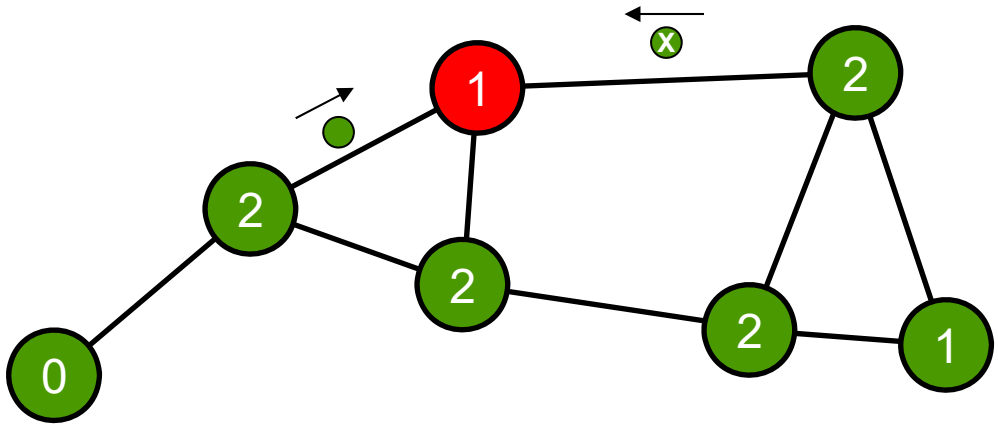
12 explorer sent
10 confirmations sent

Flooding with Confirmation – Example



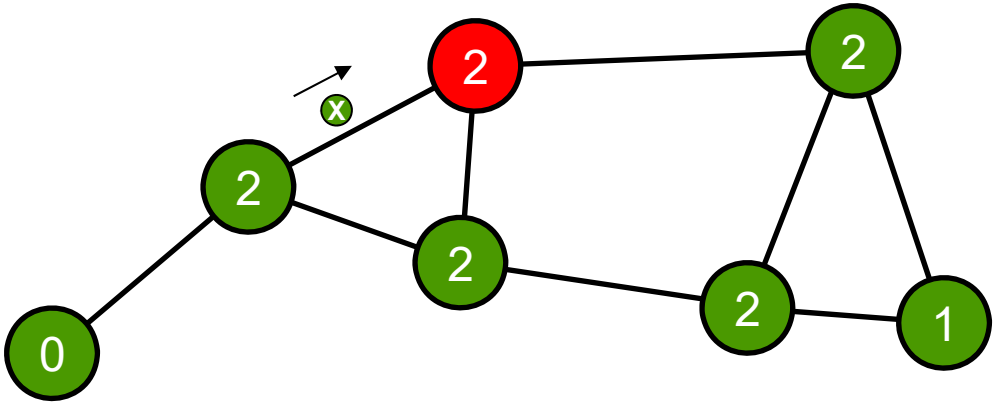
12 explorer sent
11 confirmations sent

Flooding with Confirmation – Example



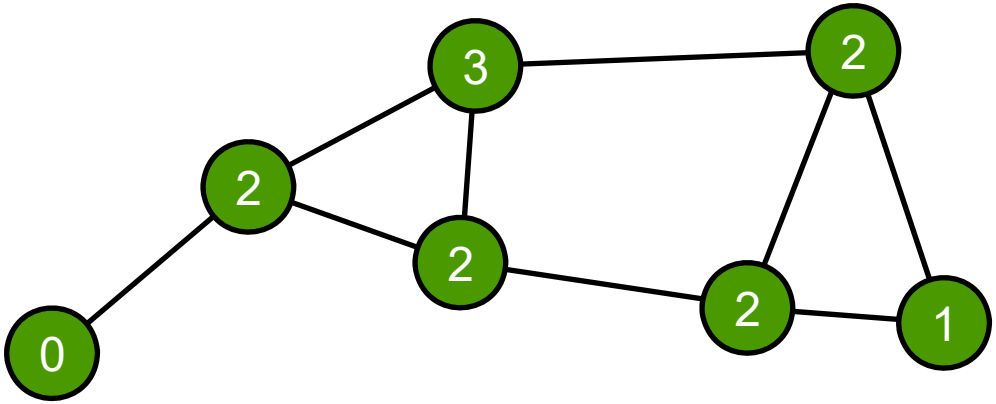
12 explorer sent
12 confirmations sent

Flooding with Confirmation – Example



12 explorer sent
12 confirmations sent

Flooding with Confirmation – Example



12 explorer sent
12 confirmations sent

Msg. Complexity of Flooding w/ Confirmation

- How many explorers altogether?
  - Every node sends an explorer on all its edges → +2*e* explorer
  - But not on its activation edge → -*n* explorer
  - Exception initiator → +1 explorer
  - $2e - n + 1$ explorer
- How many confirmations altogether?
  - Every node gets a confirmation on every edge → +2e messages
  - But not on its activation edge → -*n* messages
  - Exception initiator → +1 message
  - $2e - n + 1$ confirmations
- Altogether: $4e - 2n + 2$ messages, which is twice the number for flooding without confirmation

Echo

Echo Algorithm

- Initially all nodes are *white*
- The unique initiator becomes *red* and sends explorers to all its neighbors
- A white node, receiving an explorer
  - memorizes that edge as activation edge and
  - becomes red itself and sends explorers to all other neighbors
- A red node that has received an explorer or an echo over all its edges
  - becomes green and sends a echo over its activation edge that also becomes green
  - leaves immediately send an echo when receiving an explorer

Echo Algorithm

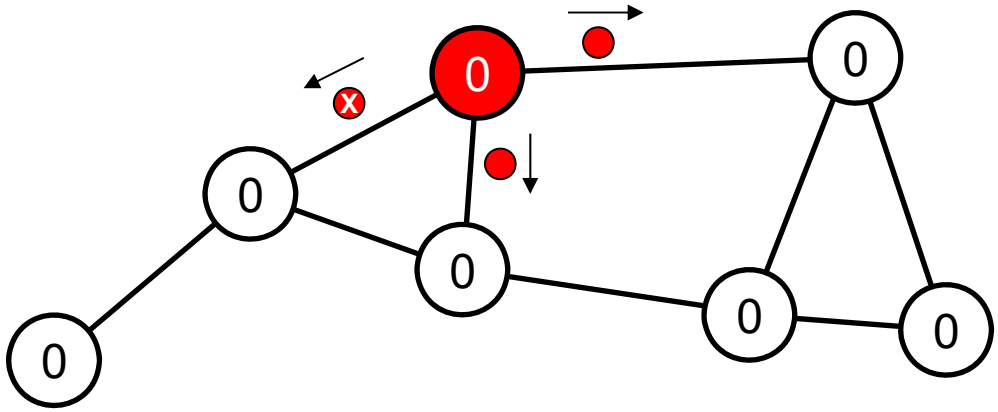
```
I:{NOT informed} // executed by initiator
   SEND <explorer> TO all neighbors;
   informed := TRUE;

R:{a message from neighbor N is received}
   IF (NOT informed) THEN // first explorer
      SEND <explorer> TO all neighbors except N;
      informed := TRUE;
      A := N; // save activation edge
   FI
   count++; // count message
   IF (count == #neighbors) THEN
      IF (NOT initiator) THEN
         SEND <echo> TO neighbor A;
      ELSE
         EXIT; // termination
      FI
   FI
```

Initially, `informed == false` and `Count == 0` for all nodes

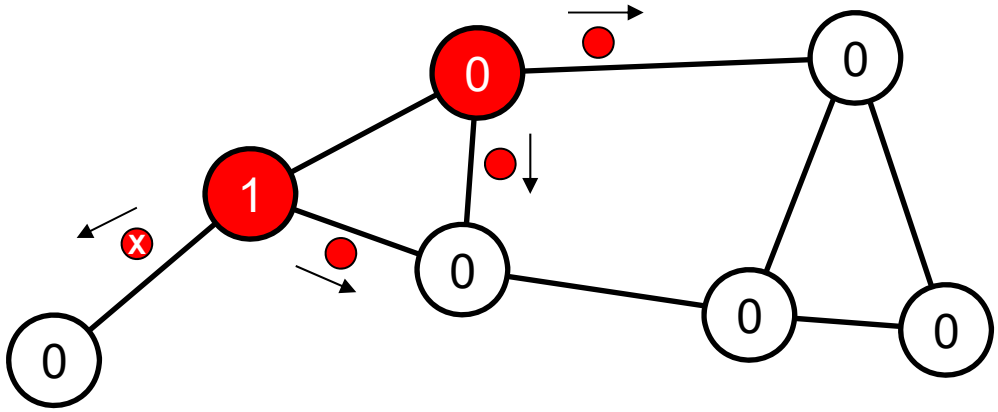
Echo Algorithm – Example

Here, all received messages are counted.



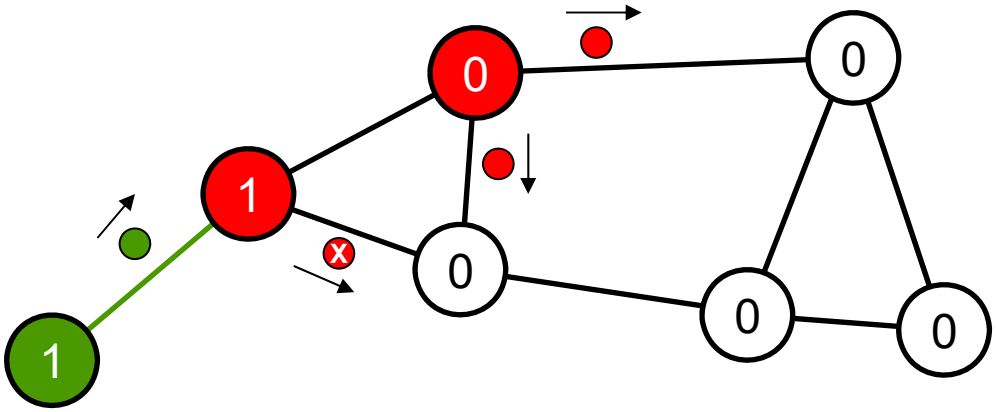
3 explorer sent
0 echos sent

Echo Algorithm – Example



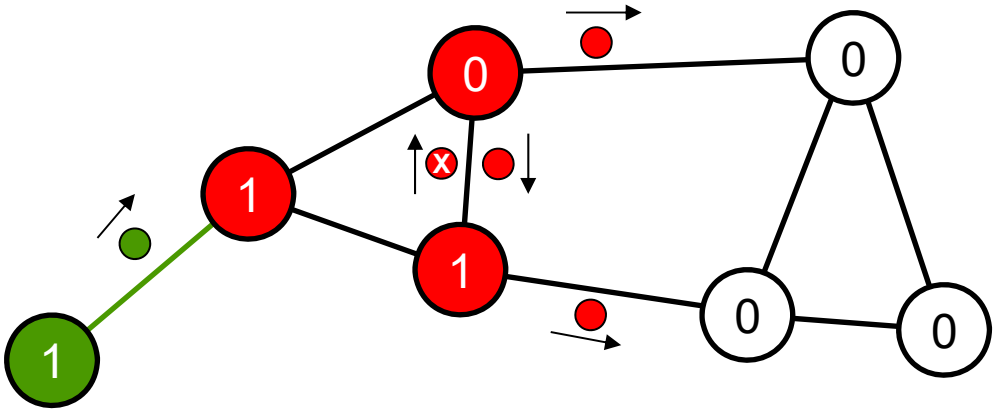
5 explorer sent
0 echos sent

Echo Algorithm – Example



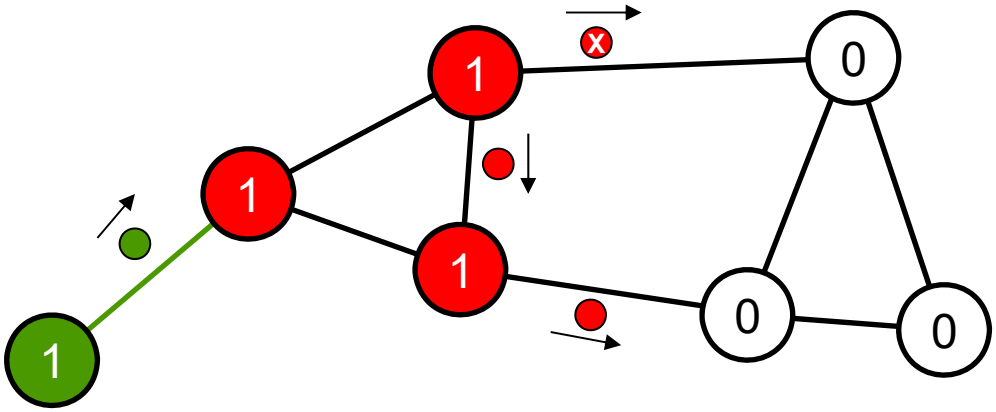
5 explorer sent
1 echos sent

Echo Algorithm – Example



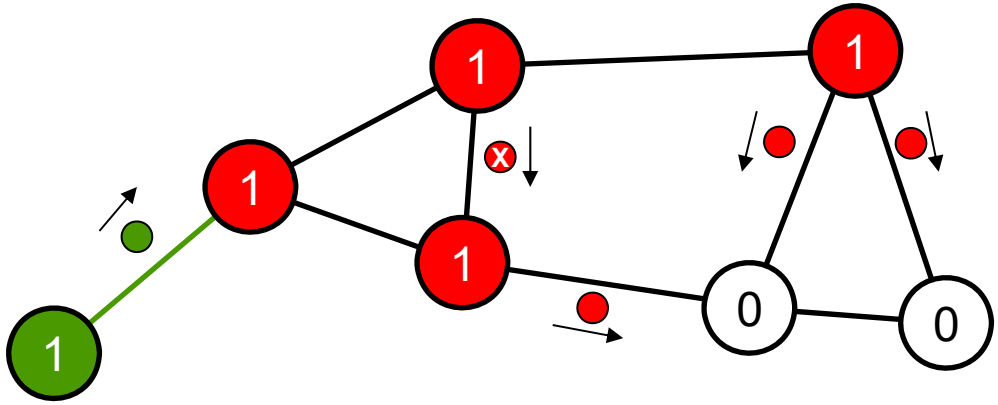
7 explorer sent
1 echos sent

Echo Algorithm – Example



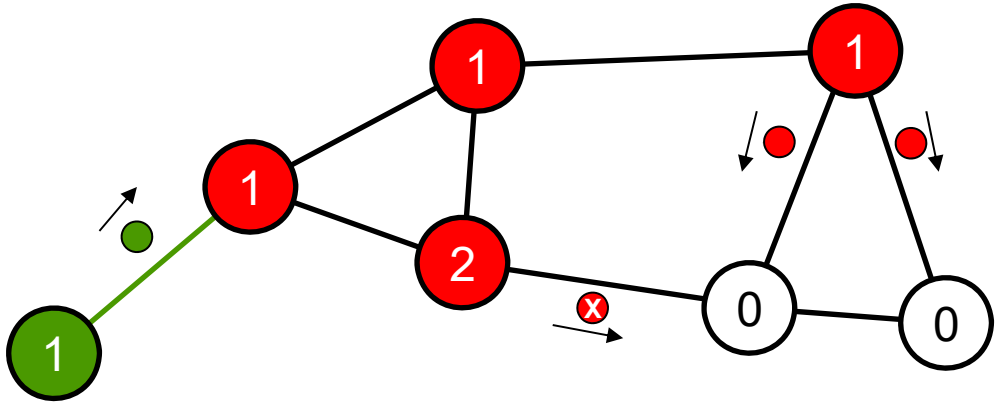
7 explorer sent
1 echos sent

Echo Algorithm – Example



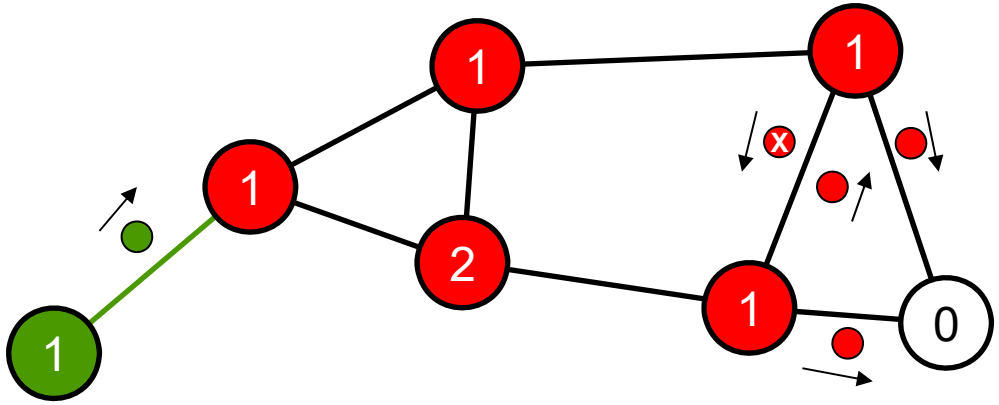
9 explorer sent
1 echos sent

Echo Algorithm – Example



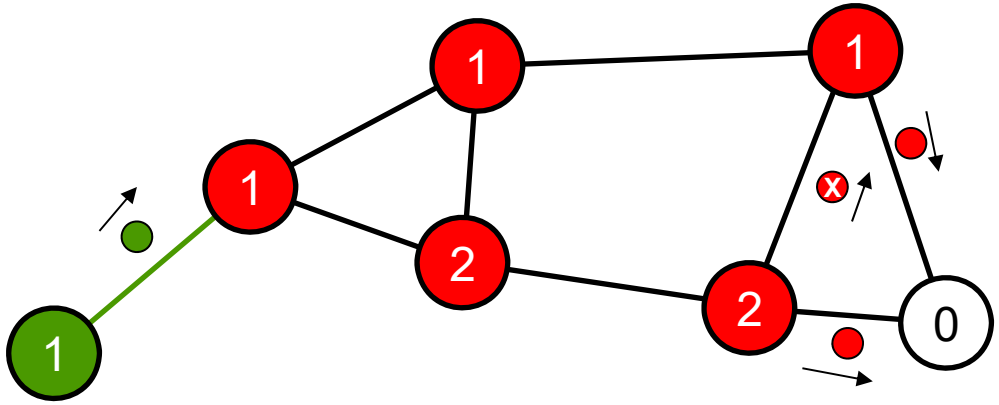
9 explorer sent
1 echos sent

Echo Algorithm – Example



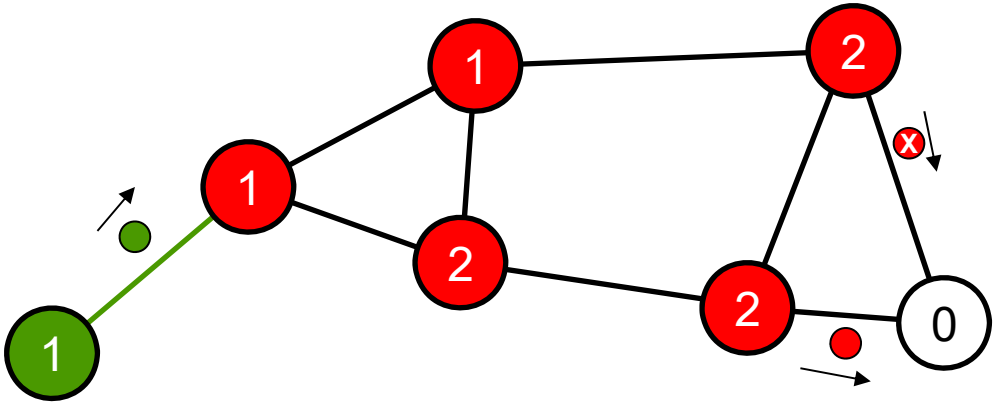
11 explorer sent
1 echos sent

Echo Algorithm – Example



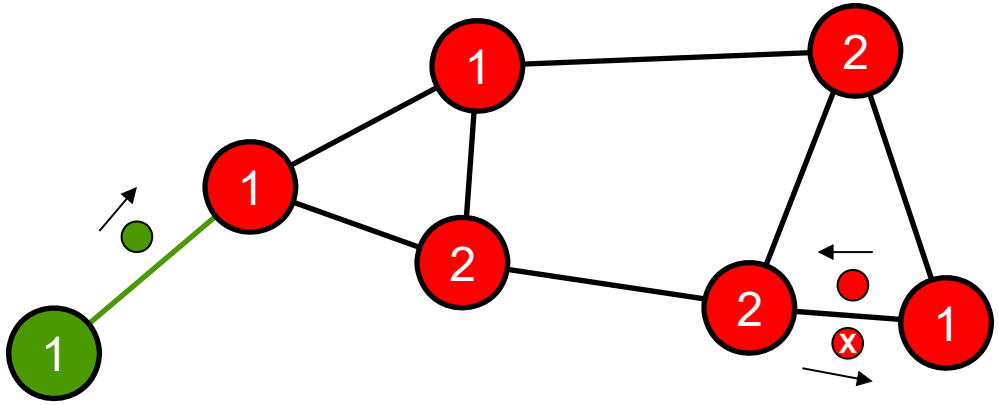
11 explorer sent
1 echos sent

Echo Algorithm – Example



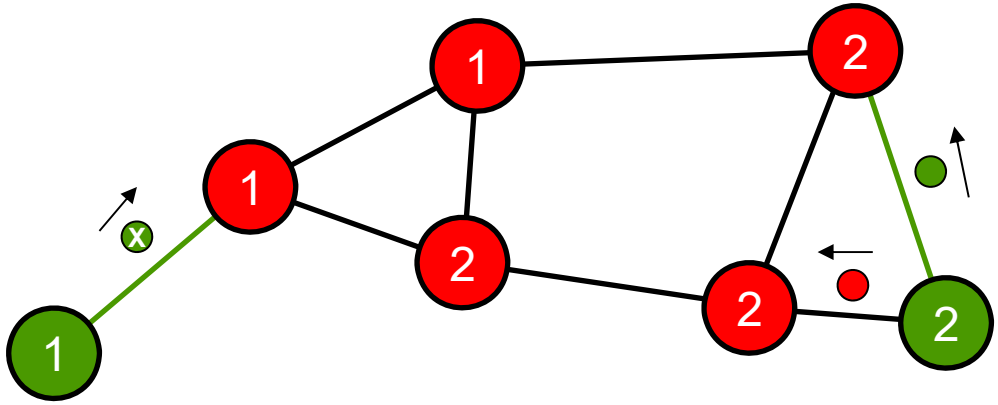
11 explorer sent
1 echos sent

Echo Algorithm – Example



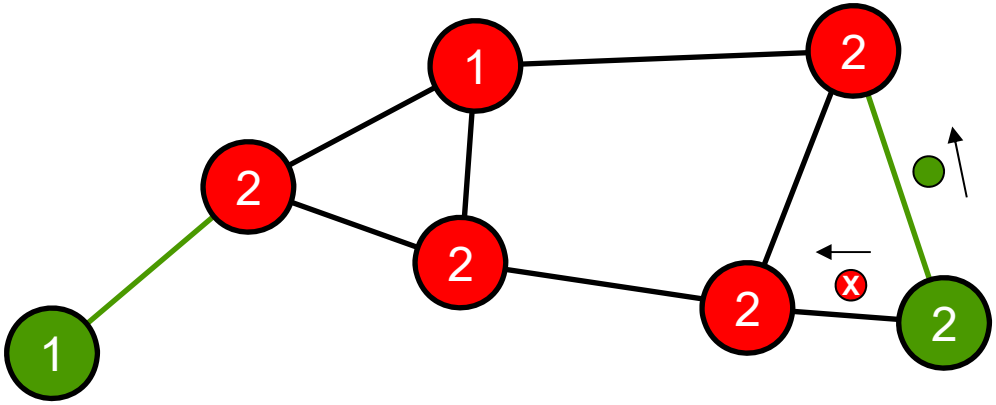
12 explorer sent
1 echos sent

Echo Algorithm – Example



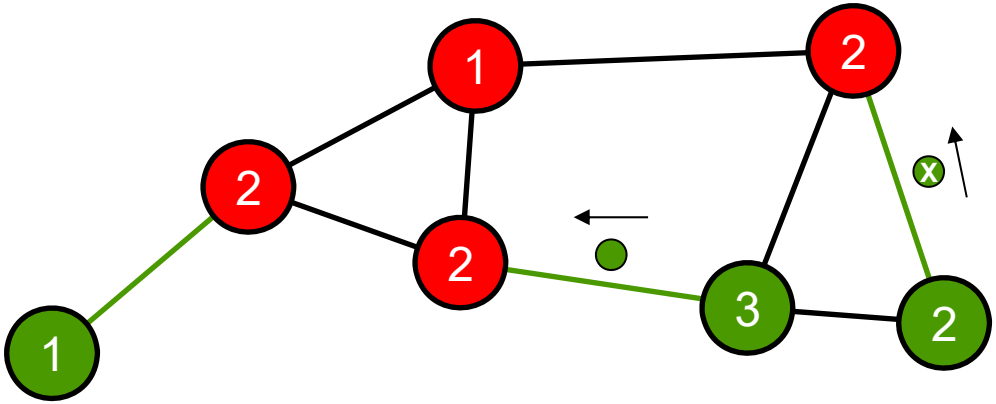
12 explorer sent
2 echos sent

Echo Algorithm – Example



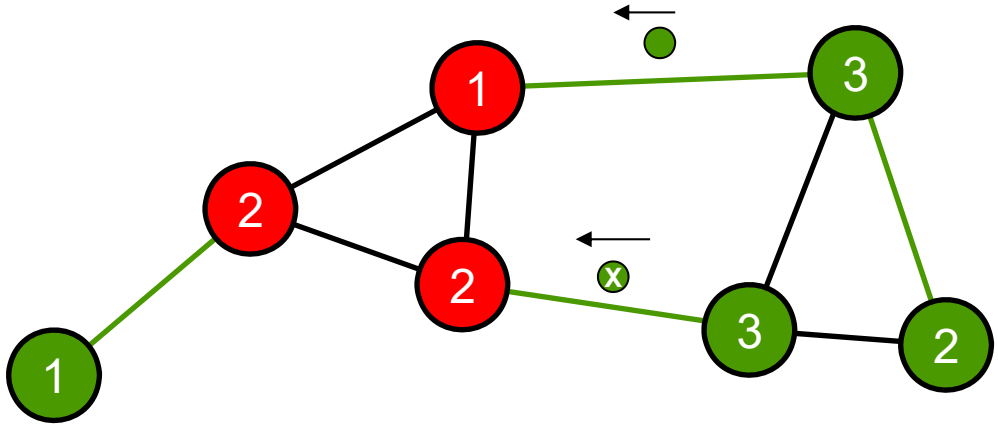
12 explorer sent
2 echos sent

Echo Algorithm – Example



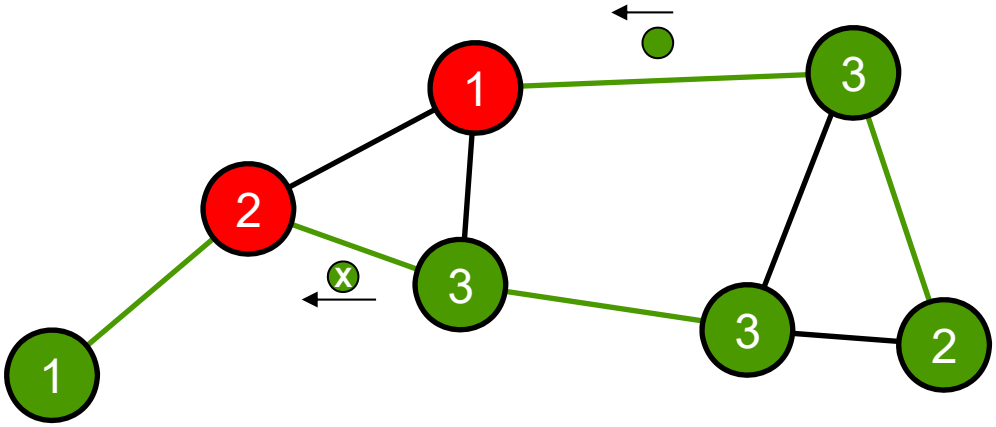
12 explorer sent
3 echos sent

Echo Algorithm – Example



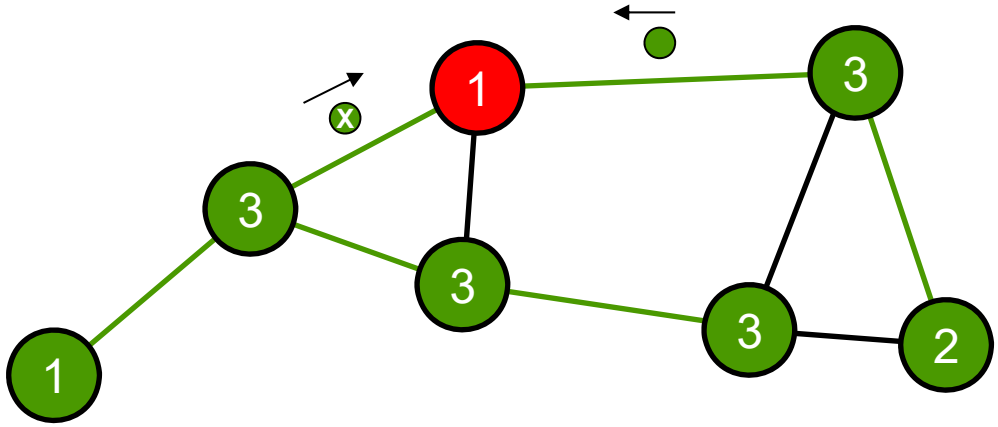
12 explorer sent
4 echos sent

Echo Algorithm – Example



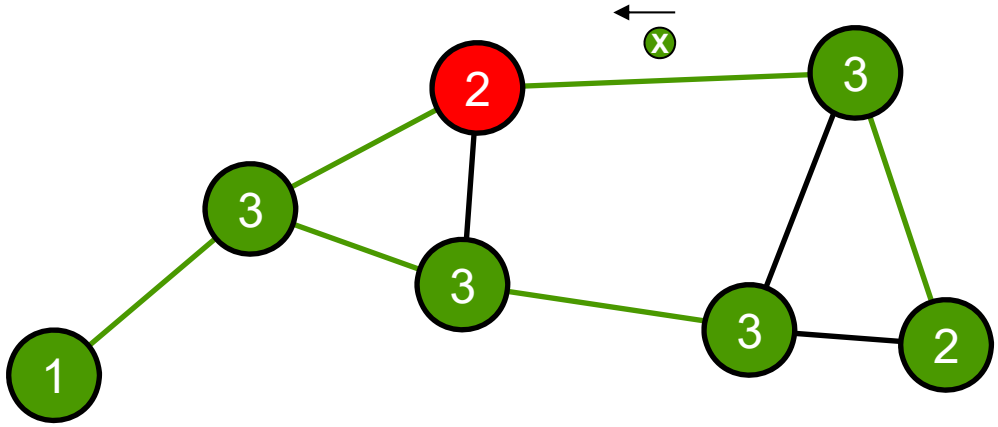
12 explorer sent
5 echos sent

Echo Algorithm – Example



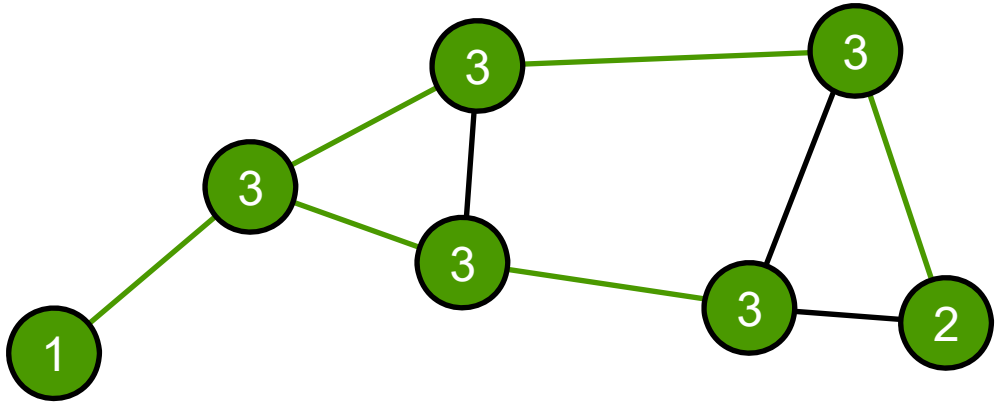
12 explorer sent
6 echos sent

Echo Algorithm – Example



12 explorer sent
6 echos sent

Echo Algorithm – Example



12 explorer sent
6 echos sent

Echo Algorithm – Message Complexity

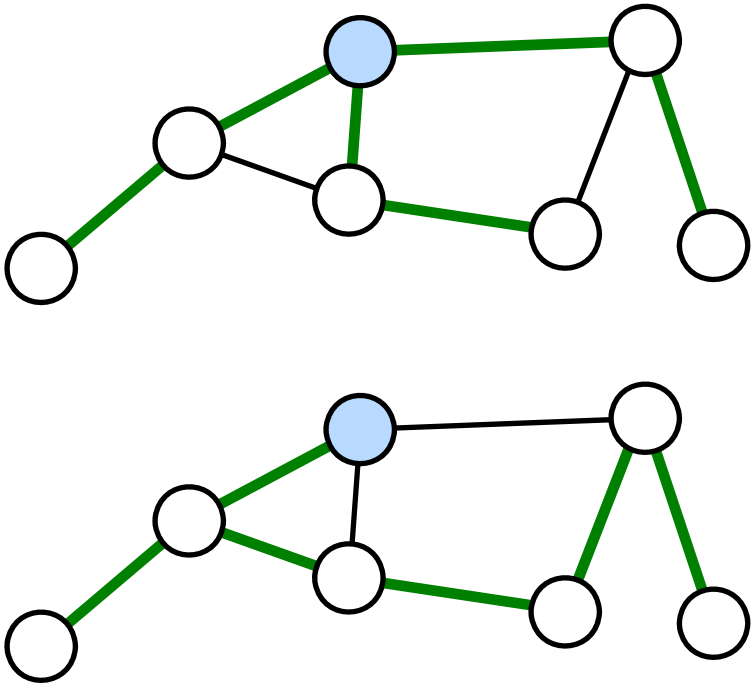
- How many messages altogether?
  - Every node sends an explorer on all edges → +2*e* explorer
  - Exception: activation edge → -*n* explorer
  - Exception from exception: initiator → +1 explorer
  - Every node sends an echo on its activation edge → +*n* echos
  - Exception: initiator → -1 echo
- Parallel traversing a graph with 2*e* messages
- Exactly two messages traverse every edge in opposite directions; two cases can appear
  1. An explorer and an echo
  2. Two explorers

Echo Algorithm – Characteristics

- Echo algorithm is a wave algorithm
- Forth wave: nodes become red
  - Distribution of information (to all nodes over all edges)
- Back wave: nodes become green
  - Collecting of information
    (from potentially all nodes over the activation edges)
  - Construction of a spanning tree

Echo Algorithm – Characteristics

- Echo edges form a spanning tree
- Depending on the message delays, the spanning tree looks differently because fast edges are preferred



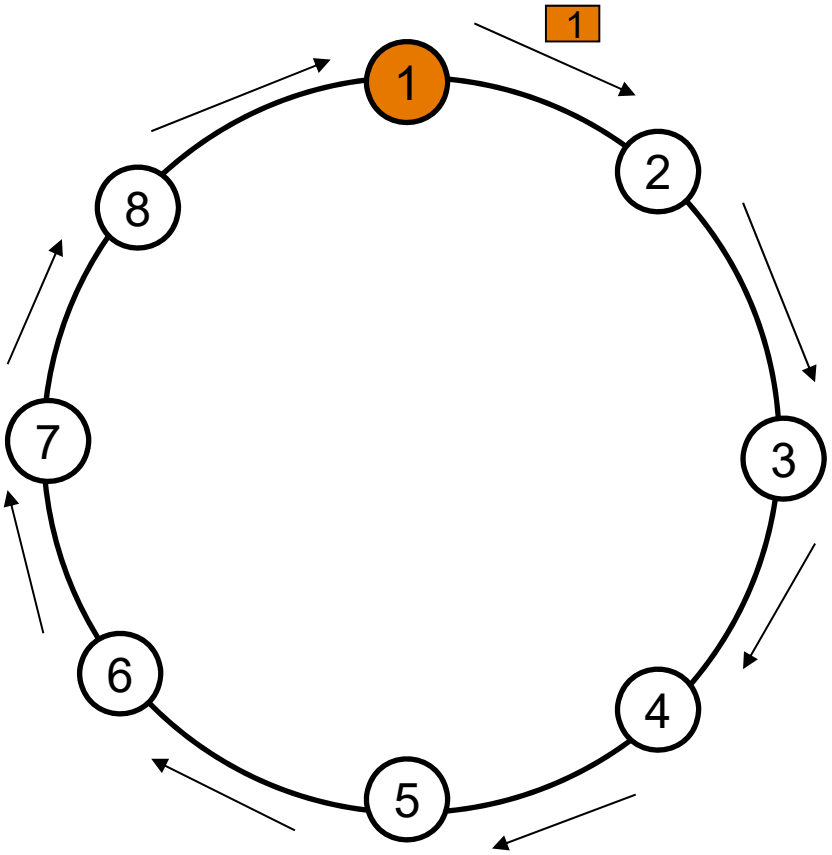
Broadcast

Broadcast on Special Topologies

- Sending a message to all nodes, optionally with confirmation
- Flooding realizes a broadcast on arbitrary connected undirected topologies
  - Fault-tolerant, because all edges are used for distributing the information
- For special topologies, a broadcast with less messages is possible, provided the algorithm is tailored to the topology
  - Less or even not fault-tolerant, because each node may only be reached over a single edge → $n - 1$ messages
- Exemplary topologies: rings, trees, hypercubes

Broadcast on Unidirectional Rings

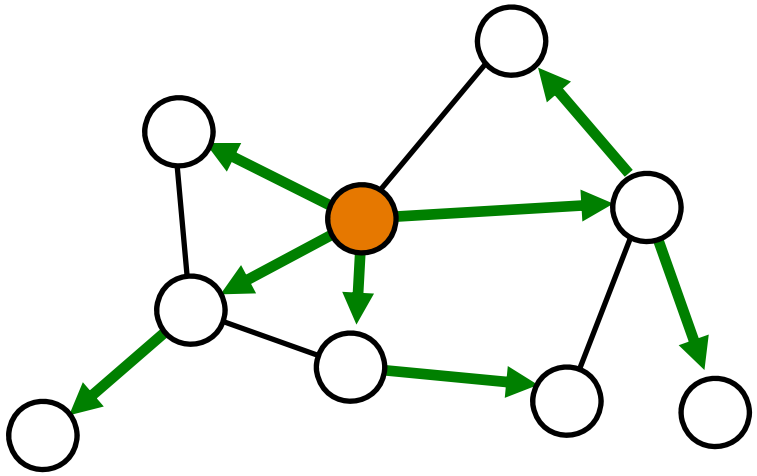
- Token circulates with message
- All nodes are informed, if the token reaches the initiator again
- *n* messages with or without confirmation
- A ring can also be defined on another topology by numbering the nodes from 1 to *n* and allowing only communication between nodes (*i*, *i* + 1) and (*n*, 1)
  → logical ring



Broadcast on Trees

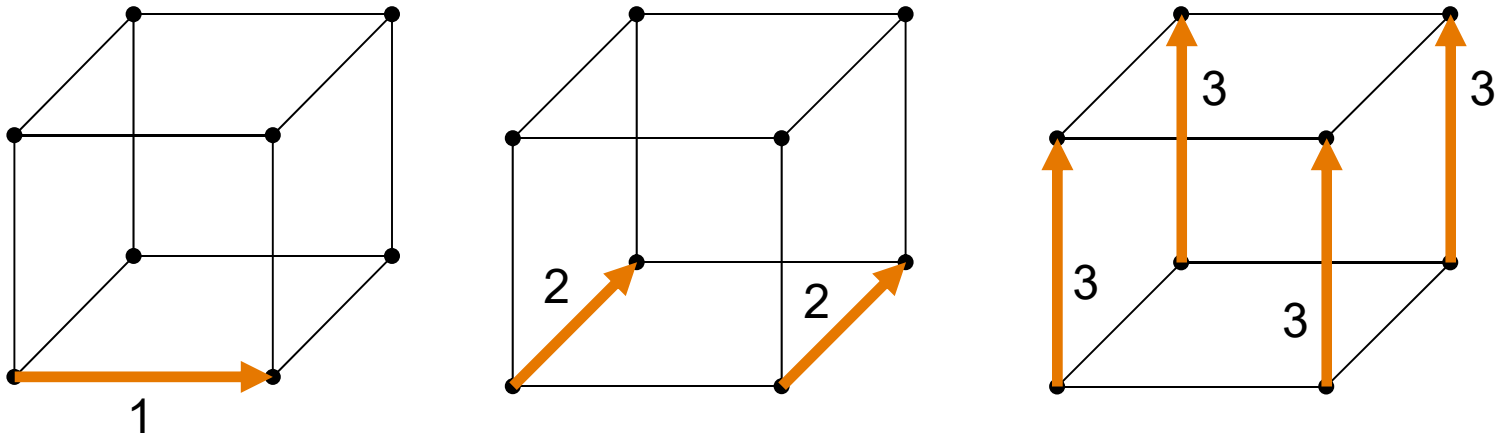
- Tree has $n - 1$ edges
- One message traverses each edge without confirmation
- With confirmation, one additional message traverses every edge

- A tree can be defined on another topology
  → spanning tree



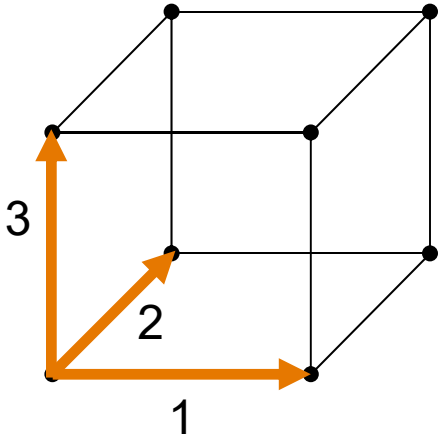
Broadcast on Hypercubes

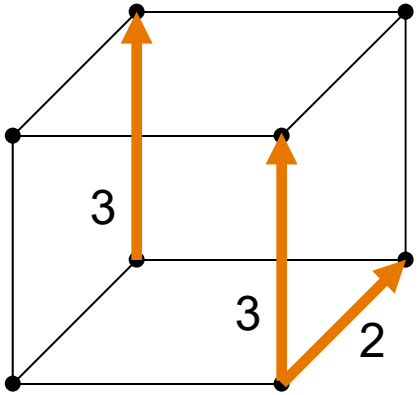
- Analogous to recursive construction of a hypercube
  1. Initiator sends in dimension 1
  2. All nodes of dimension 1 send in dimension 2
  3. All nodes of dimension 2 send in dimension 3
  - ...

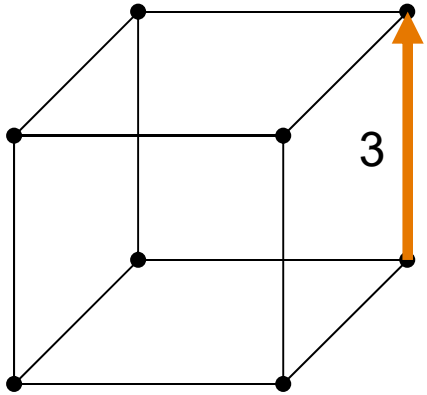


Broadcast on Hypercubes

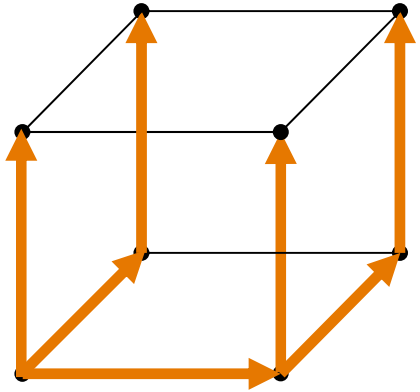
- Three nodes notified after one time unit
- Three nodes notified after two time units
- One node notified after three time units







Resulting spanning tree



Broadcast on Hypercubes

- Unit time complexity
  - After d cycles all nodes are informed
  - Is that optimal?
  - Yes, because the diameter of a hypercube is d
- Message complexity
  - $1 + 2 + 4 + \ldots + 2^{d-1} = 2^d - 1 = n - 1$
  - Is that optimal?
  - Yes, for reaching each of the $n - 1$ other nodes at least a single message is necessary

Exemplary Exam Questions

1. Explain the functioning of Flooding, Flooding with confirmation and the Echo algorithm!
2. What are the message complexities of these algorithms?
3. What determines which edges are part of the Echo algorithm generated spanning tree?
4. What message complexity does a broadcast have at least?
5. Explain broadcast on hypercubes!
6. What is the unit time complexity and the message complexity of broadcast on hypercubes?

Literature

- E. Chang. *Echo algorithms: Depth parallel operations on graphs*. IEEE Transactions on Software Engineering, 8(4):391--400, 1982.

Thank you for your kind attention!

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

gero.muehl@uni-rostock.de
http://wwwava.informatik.uni-rostock.de