Multicast Communication

February 16, 2021

Roadmap

Multicast

Application-Level Multicast

Epidemic Algorithms

Roadmap

Multicast

Application-Level Multicast

Epidemic Algorithms

Multicast Communication

- Is communication via a channel with n receivers
- On broadcast networks ((W)LANs), can be done efficiently by using broadcast communication provided by the MAC layer
- On point-to-point networks, can be implemented by:

Multicast Communication

- Is communication via a channel with n receivers
- On broadcast networks ((W)LANs), can be done efficiently by using broadcast communication provided by the MAC layer
- On point-to-point networks, can be implemented by:
  - n single-ended channels, if only one sender
  - $n \cdot m$ single-ended channels, if m senders
- But:

Multicast Communication

- Is communication via a channel with n receivers
- On broadcast networks ((W)LANs), can be done efficiently by using broadcast communication provided by the MAC layer
- On point-to-point networks, can be implemented by:
  - n single-ended channels, if only one sender
  - $n \cdot m$ single-ended channels, if m senders
- But:
  - The sender has to know each of the receivers
  - The sender has to send n separate messages
    - n system calls
    - Several links in the underlying communication network will be traversed by the same message
- More efficient implementations can be done at:

  Network Layer *IP Multicast*
  Application *Application Level Multicasting*
    - By means of **overlay networks**, networks built on top of the Internet

Multicast On Point-to-Point Networks

Question How can you **broadcast** efficiently on a point-to-point network?

Multicast On Point-to-Point Networks

Question How can you **broadcast** efficiently on a point-to-point network?

Answer Use a spanning tree

- This is a tree that includes all nodes of the network

Question How can you **multicast** efficiently on a point-to-point network?

Multicast On Point-to-Point Networks

Question How can you **broadcast** efficiently on a point-to-point network?

Answer Use a spanning tree

- This is a tree that includes all nodes of the network

Question How can you **multicast** efficiently on a point-to-point network?

Answer Use a spanning tree that includes:

- The sender
- The *n* receivers
- The nodes between them (to ensure we have one tree)

IP Multicast: Lab 2

- Applications can use IP multicast via UDP
  - Specifying reliable multicast is not easy, let alone implement it
- For the sender, it is just like unicast with UDP
  - Except that it must use an IP multicast address, rather than the IP address of a host
  - The routers forward the packets along the spanning tree, so that all group members receive the datagram
- A receiver:
  1. Must **subscribe** the multicast group before receiving
     - This is needed so that it is added to the spanning tree
  2. Should **unsubscribe** the multicast group when it does not wish to receive more messages to that group
     - This allows pruning unused branches of the spanning tree

IP Multicast

- The IP multicast interface is nice (if you can live with "UDP guarantees")
- Unfortunately IP multicast management does not scale across multiple administrative domains (take it on faith ...)
  - It appears it is costly and ISPs are unable to monetize it
- This is somewhat ironic:
  - IPv4 did not support multicast
    - But nevertheless, the first implementation of multicast on the Internet, was on IPv4 (end of 1980s)
    - It relied on the MBone, an overlay/virtual network, that used IP tunnelling
  - IPv6 supports multicast
    - Supposedly multicast should be widely available ...

Some Alternatives

All of them use an **overlay network** whose nodes are the multicast group members, and whose edges are (virtual) links between the nodes

- Need not be a complete graph

Some Alternatives

All of them
: use an **overlay network** whose nodes are the multicast group members, and whose edges are (virtual) links between the nodes

- Need not be a complete graph

Application-level Multicast

- Build a **spanning tree** on the overlay network
- A node that wants to send to the multicast group sends the message to the root of the tree
- The message is then multicasted, by using unicast communication along the tree branches, from the root towards the leaves

Some Alternatives

All of them use an **overlay network** whose nodes are the multicast group members, and whose edges are (virtual) links between the nodes

- Need not be a complete graph

Application-level Multicast

- Build a **spanning tree** on the overlay network
- A node that wants to send to the multicast group sends the message to the root of the tree
- The message is then multicasted, by using unicast communication along the tree branches, from the root towards the leaves

Epidemic Protocols use **limited flooding** , rather than a spanning tree, on the overlay network

- By exchanging messages with some neighbors, a message eventually spreads to all nodes in the network

Roadmap

Multicast

Application-Level Multicast

Epidemic Algorithms

Example: *Switch Trees* (1/4)

Problem The implementation of optimal algorithms such as *shortest-path tree (SPT)* or *minimum-spanning trees (MST)* is not practical

- The maximum degree of the tree nodes may easily exceed the capacity of the nodes
- The MST algorithm is complex

Idea Incrementally change the topology of the multicast tree:

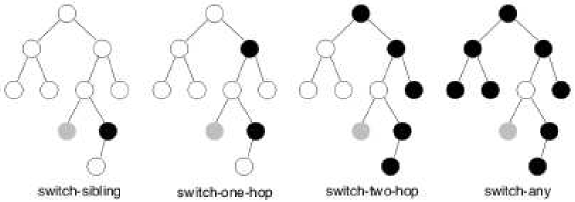
- Taking into account resource constraints
- But improving some performance metric, e.g. the cost

Limitation Assumes that the multicast tree has been previously created

- For example, when a node joins the tree it becomes a child of the root
  - By performing small changes, the topology becomes more balanced

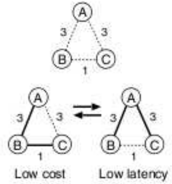
Example: *Switch Trees* (2/4)

- In principle, a node may switch its parent to any node that is not in the subtree of which it is the root. **Why?**
- By imposing restrictions on the **candidates** for new parent, we can obtain different protocols:



source: Helder and Jamin 2002

- The selection of a node among the candidates can use one of several metrics:
  - "Cost" of the tree (approx. MST)
  - Delay to the root (source) (approx. SPT)



source: Helder and Jamin 2002

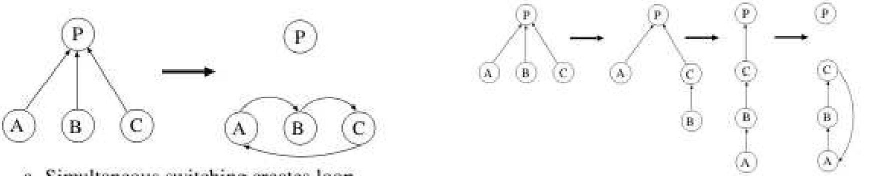
Example: *Switch Trees* (3/4)

Banana Tree Protocol (BTP)

- *One-hop switch* protocol
- When a node joins the multicast group, it becomes a child of the root
  - If the tree does not exist, it becomes the root
- To switch its parent, a node has to ask for permission from its new parent, which may reject the request
- If a node fails, the tree of which it is root partitions, and its children will become children of the root
  - Alternatively, to avoid overloading the root, they can become children of their grandparent, i.e. the parent of the faulty node

Example: *Switch Trees* (4/4)

BTP: Cycles



a. Simultaneous switching creates loop

b. Switching with outdated information creates loop

source: Helder and Jamin 2002

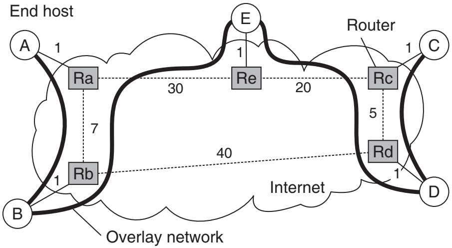
a Concurrent attempts by different nodes to switch their parents may lead to cycles in the "spanning tree"
- This can be avoided, if one node that is in the process to switch its parent, rejects all requests to become parent of another node
  - What's the issue with this approach?

b Outdated topological information may also lead to cycles
- May be prevented by including topological information in the authorization request
  - E.g., in the switch-one-hop algorithm, the parent of the requesting node is enough

Application-layer Multicast (ALM) Overheads

Problem Neighbors of the overlay network may be many hops away on the underlying physical network

- This may lead to a less efficient use of the network



Link stress How many times is a physical link crossed by a message on its multicast?

- For example, a message traverses *link* (Ra, Rb) twice

Link stretch Ratio of the distances between two nodes on the overlay network and on the physical network

- The cost of the path between B and C in the multicast tree is 73(pg *vs* 47 in the underlying physical network

Roadmap

Multicast

Application-Level Multicast

Epidemic Algorithms

Background

Objective Disseminate information by the nodes (replicas) of a distributed system

Idea Update the information by passing it to some neighbor nodes

- These will pass it on to their neighbors in a "lazy" way, i.e. not immediately.
- Eventually, all nodes with copies of that piece of information will update it.

Note The name (**epidemic**) stems from the fact that these protocols spread information/messages in a way analogous to the spread of a contagious disease

Alternatives

Anti-entropy Each node periodically chooses a random node with which it exchanges messages.

Rumor spreading A node **N** that has a "new" message passes it on to other nodes

- But if node **N** picks a node that has already received that message, it may stop disseminating that message.
- Actually, this is a variant of anti-entropy

Anti-Entropy

Idea

Periodically node P randomly chooses node Q for exchanging messages

Results from the theory of epidemic propagation

- Eventually, all nodes will receive all messages. I.e. the probability of a node missing a message tends to 0

Alternatives for Message Exchange

Push P only pushes its messages to Q

Pull P pulls in new messages from Q

Push-Pull P e Q exchange messages

- After this exchange P and Q have the same messages

Strategy Analysis

Let a **round** be the time interval required for each node to pick another node and exchange messages with it

Let p_i be the probability of a node missing a message after i rounds

A node that has not received the message after i rounds, does not receive it after $i+1$ rounds, if:

Push: none of the nodes that received the message after i rounds pick it

$$p_{i+1} = p_i \cdot \left(1 - \frac{1}{N-1}\right)^{(1-p_i)N} \approx p_i e^{-1}$$

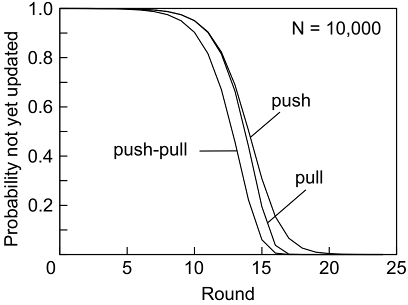
Pull: it picks a node that has not received the message after i rounds

$$p_{i+1} = (p_i)^2$$

Push-Pull: both

- none of the nodes that received the message after i rounds pick it, **and**
- it picks a node that has not received the message after i rounds

Strategy Comparison



Src.: van Steen & Tanenbaum

Push Propagation of a message in the "final phase" slightly slower

- As the message is disseminated, the probability of choosing a node that does **not** have the message **decreases**

Pull Propagation in the final phase slightly faster

- As the message is disseminated, the probability of choosing a node **with** the message **increases**

Push-Pull Combines the advantages of both

Gossiping

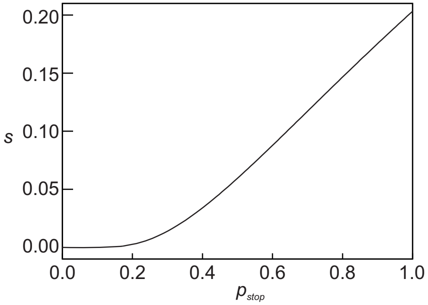
Idea

Variation of epidemic algorithms, in which node P looses the motivation to disseminate a message, if it tries to disseminate it to another node Q that already knows it

- Disseminates messages rather efficiently
- Does not ensure that all nodes will receive the message
  - Let p_{stop} be the probability of P stopping disseminating a message, if Q already received it
  - Then, the fraction, s, of nodes that will not receive the message is:

$$s = e^{-(1+\frac{1}{p_{stop}})(1-s)}$$

(for $p_{stop} = 0.2$, $s \approx 0.0025$)



Src.: van Steen & Tanenbaum

Epidemic Algorithms: Discusssion

- Robust
  - Can easily tolerate crashes on nodes
  - Even if each node has only a partial view of the system, if this vision is continuously updated, the result is a random graph
- Highly scalable
  - Sincronization between nodes is local
- Yet, the analysis above assumes that any node can randomly select any other node
  - It would require every node to know every other node: not very scalable