Probabilistic Multicast

Patrick Th. Eugster
Rachid Guerraoui

Distributed Programming Laboratory
Swiss Federal Institute of Technology
in Lausanne
Context

- **Reliable multicast**
  - Ensure that a precise subset of processes/nodes in a group delivers a message (ideally none of the other processes receives the message)

- **In an environment characterized by**
  - Large number of processes
    - Not every process can/will know every other
    - No process is interested in every message
    - No global and static classification of interests feasible
  - No global network-level multicast protocol
Approaches

- **Deterministic schemes**
  - With **strong** reliability guarantees [HT93] do not scale sufficiently (e.g., $O(n^2)$ msgs)

- **Probabilistic, gossip-based, schemes**
  - Every process periodically (every $P$ ms) „talks” to a subset of (Fanout, $F$) processes about some messages
  - Good trade-off between reliability and scalability
    - Very resilient to arbitrary crash failures
    - Usually the „time“ necessary to reach all processes in a group is $O(\log n)$, $n$ being the size of the group
  - Several broadcast algorithms, e.g., pbcast [Birman et al.’99], lpbcast [Eugster et al.’01]
Simple Solutions?

- „Plain broadcast“: send everything to everybody
  - Non-interested processes receive too; ↓ network resources
  - Everybody has to know everybody; ↓ memory resources

- „True multicast“: send only to interested processes
  - Have to know all, and their interests; ↓↓ memory resources

- „Subgroup broadcast“: divide process group according to interests
  - Send data to those groups of processes manifesting corresponding interests
  - Need to know interests to create groups, groups might change as soon as interests change
Probabilistic Multicast

**Specification**

- **Validity**: If a correct process multicasts a message $m$, then some correct process in $\text{Dest}(m)$ eventually delivers $m$.

- **Integrity**: For any message $m$, every correct process $p$ delivers $m$ at most once, and only if $m$ was previously multicast by $\text{Sender}(m)$.

- **Probabilistic Agreement**: If a correct process in $\text{Dest}(m)$ delivers message $m$, then every correct process in $\text{Dest}(m)$ eventually delivers $m$ with known, high, probability $\mathbb{P}$.

Note: $\text{Dest}(m)$ given implicitly by „interests“ of processes.

**Implicit requirements**

- Scalability w.r.t. message & time complexity, membership knowledge.
Implementing pmcast: An Intuition

- **Spanning tree**
  - Little to no redundant sends
  - No process has to know all the other processes in the tree

- **Gossips**
  - Good probability of success with unreliable links
    - The children of a node gossip among them, a.s.o.
    - With respect to a particular message only those interested/representing interested children

- **Fault-tolerance**
  - Every „node“ in the tree is represented by \( r > 1 \) processes
  - A process which appears as „member“ of a given node also appears in the child nodes
Building the "Tree"

- **Bottom-up**
  - Processes have precise knowledge about "close" neighbors
  - Knowledge decreases with the "distance"
  - Processes are orchestrated in subgroups, according to distance
  - $r$ delegates are chosen for each subgroup, manifest interests of all represented processes
  - Each process knows the $r$ delegates of each (known) subgroup

- **Recursively**
  - Put several subgroups of level 1 together: form a group of level 2
  - Elect $r$ delegates for every such supergroup
  - etc.
pmcast Algorithm Overview

- Membership „tree” maintenance
  - Every process maintains table for its subgroup of each level
  - Periodically processes exchange membership info with others
  - Lowest level processes are „monitored“

- Level-wise gossiping
  - Number of nec. gossip rounds is computed through approx.
  - Gossips merged with membership info

- Every process has a buffer for each level
  - Periodically, every message of every buffer
    - Forwarded to the interested processes within a random subset (maximum number of rounds)
    - Update rounds and possibly level
Illustration
Performance

- **Number of processes that a given process knows**
  - $O(\log n)$

- „Worst“ case: broadcast
  - Same „time“ to complete as when every process knows every other process: $O(\log n)$
  - Same message complexity: $O(n \log n)$

- **Probability of delivery for interested processes**
  - Comes very close to 1

- **Probability of receiving for non-interested processes**
  - Very small for cases where „enough“ processes are interested in a given message
Deliver and Unvoluntary Delivery

10000 processes, 3 levels, redundancy and fanout of 3
Scalability and Tuning

3 levels, redundancy 4, fanout 3

?10000 processes, 3 levels, redundancy and fanout of 3
Conclusions and Ongoing Efforts

“Natural“ trade-offs

- Increasing the number of levels
  - Reduces the membership knowledge each process has
  - Increases the average filtering load, etc.
- Approximations for the number of rounds at each level
  - Small vs large fractions of interested processes

Root processes

- Require more computing power
  - Use approximate matching to reduce filtering load
- Require more memory
  - Hash functions to compact space for „interests“
Questions