Beyond Replicated Storage: Eventually-Consistent Distributed Data Structures

Konrad Iwanicki
University of Warsaw

PaPEC 2014, Amsterdam, the Netherlands, April 13th, 2014
What we do

- Extreme distributed systems
What we do

- Extreme distributed systems
  - Based on wireless sensor networks
What we do

• Extreme distributed systems
  • Based on wireless sensor networks:
    – Hundreds or even thousands of nodes in a network
    – A single node is severely constrained in resources
Extreme distributed systems

- More complex than sense-and-send:
  - Sensing
  - Analyzing and deciding
  - Actuating
Extreme distributed systems

• More complex than sense-and-send:
  • Sensing
  • Analyzing and deciding
  • Actuating

\{ \text{collaborative} \}
Extreme distributed systems

• More complex than sense-and-send:
  • Sensing
  • Analyzing and deciding \( \bigcup \) collaborative
  • Actuating

• Subject to various challenges:
  • Resource constraints
  • Unreliable communication
  • Interactions with the surrounding environment
Extreme distributed systems

- More complex than sense-and-send:
  - Sensing
  - Analyzing and deciding
  - Actuating

- Subject to various challenges:
  - Resource constraints
  - Unreliable communication
  - Interactions with the surrounding environment

- Distributed algorithms are increasingly complex
  - e.g., employ specific organizations.
Cluster hierarchy

Wireless connectivity between nodes.
Each node forms a level-0 cluster of which it becomes the head.
Cluster hierarchy

Proximate level-0 clusters are grouped into level-1 clusters.
Cluster hierarchy

And so on at higher levels, typically until a single cluster remains.
Cluster hierarchy

The membership of a node in the hierarchy is reflected in the node’s label.
Each node also maintains information for each sibling cluster in the hierarchy (e.g., a **routing entry**).
The problem

- Using the hierarchy is relatively easy:
  - Routing
  - Aggregation
  - In-network storage
- Maintaining it is a different story.
  - Connectivity changes
  - Node failures & arrivals
  - Nodes should be autonomous
General scheme – gossiping

- Each node maintains its state which it occasionally updates.
- For communication:
  - Each node operates in rounds.
  - In each round, it broadcasts its state to its neighbors.
  - It also receives the neighbors' states, which it merges with its own one.
Observation

- Gossiping can be efficient
Observation

- Gossiping can be efficient, but...
- … it makes it difficult to control when a bit of information reaches a particular node.
Observation

- Gossiping can be efficient, but...
- … it makes it difficult to control when a bit of information reaches a particular node.

- We have:
  - Updates done by nodes
  - Lazy update propagation
Observation

- Gossiping can be efficient, but...
- … it makes it difficult to control when a bit of information reaches a particular node.

- We have:
  - Updates done by nodes
  - Lazy update propagation

\{ Resemblance to eventually-consistent replicated storage systems. \}
Observation

• Gossiping can be efficient, but...
• … it makes it difficult to control when a bit of information reaches a particular node.

• We have:
  • Updates done by nodes
  • Lazy update propagation

Resemblance to eventually-consistent replicated storage systems.

Let's thus take a look at the problem of cluster hierarchy maintenance from the eventual consistency perspective.
We treat the cluster hierarchy as a distributed structure. The state of each node, which includes label and routing table, is a part of this structure. Each node can autonomously update its local state: locally altering the structure. The updates propagate through gossiping: eventually the structure becomes consistent globally.
EC perspective

Consider node labels.
They can be viewed as a distributed tree.
EC perspective

• What is different from the “traditional” model?
  • The state of each node is not a replica.
  • On the contrary:
    – Some of its parts are unique.
    – Some are replicated at other nodes (to a varying degree).
  • On the global scale the states of all nodes should form a coherent structure.
EC perspective

Logical view:
EC perspective

Physical view:
EC perspective

Physical view:
EC perspective

Physical view:
EC perspective

Physical view:

When this changes at one node, the other nodes must update their state accordingly.
EC perspective

Physical view:

When this changes at one node, the other nodes must update their state accordingly, but

- Updates can be concurrent
- They are often not independent
- Propagate lazily
- (Think of also about all the limitations of the nodes)
EC-related challenges

- How to decide that a given piece of the distributed structure should be updated?
- How such updates should be performed and which node(s) should do them?
- How can other nodes detect and merge the updates to their corresponding pieces of the distributed structure?
- (How to do this under constrained resources?)
Our solution

• Details, for instance, in:
  

  or

Our solution – overview

- Formalize the properties of a hierarchy as invariants, e.g.:
  1. Level-0 clusters correspond to individual nodes.
  2. There exists a single top-level cluster.
  3. Level-i+1 clusters are composed out of level-i clusters.
  4. Each level-i+1 cluster has a central subcluster that is adjacent to all other subclusters of the cluster.

- Maintaining a hierarchy = detecting and eliminating violations of the invariants.
Our solution – overview

• The invariants are global
  ➔ Have to maintained collaboratively by the nodes.

• A node's state is local
  ➔ Each node is concerned only with those invariants that are relevant to its part of the distributed structure.

• Eliminating a violation:
  = local update operation

• Propagating the update:
  = eventually-consistent gossiping.
Our solution - example

Local operations for maintaining labels.
Our solution - example

- Our label:
  - P.L.G
Our solution - example

• Our label:
  • P.L.G

• The label received in a gossip message:
  • A.L.D
Our solution - example

- Our label:
  - P.L.G

- The label received in a gossip message:
  - A.L.D

- What should we do?
Our solution - example

- Our label:
  - P.L.G
- The label received in a gossip message:
  - A.L.D
- What should we do:
  - leave our label as is, or
  - change it to P.L.D?
Take-home message

- Eventual consistency can offer lots of benefits to extreme distributed systems.
- Distributed data structures appear also in other fields.
- Eventually-consistent distributed data structures are poorly understood.
Thank You

Questions?

Supported by the (Polish) National Science Centre (NCN) within the SONATA programme under grant no. DEC-2012/05/D/ST6/03582.