An Adaptive Algorithm for Efficient Message Diffusion in Unreliable Environments

Benoît Garbinato
Fernando Pedone
Rodrigo Schmidt

Proceedings of the 2004 IEEE International Conference on Dependable Systems and Networks (DSN 2004)

Our goal

We want to solve the reliable broadcast problem in large-scale distributed systems, using as few messages as possible.

We want our solution to be able to adapt to changes in the system configuration.
Outline

- Distributed systems
- Large scale systems
- Reliable broadcast
- Probabilistic approach
- Adaptation & Optimality

Distributed systems

“A distributed system is one that stops you from getting any work done when a machine you’ve never even heard of crashes.”


izzazione as the slowest
- As weak as the weakest
  ⇒ In large-scale systems, this is even worse...
Large scale

- Many nodes (processes)
- and/or
- Long-distances (high-latency)

⇒ Changes occur frequently
⇒ Difficult to act globally

Reliability

- Try to solve typical problems, e.g., consensus, reliable broadcast, atomic commitment, etc., despite the occurrence of failures
- A solution depends on the actual failure model, e.g., crash-stop, crash-recovery, Byzantine, etc.
Gossiping algorithms

Principle: only talk to some of your neighbors

Probabilistic model

Model definition

The system is defined as a graph $G = (\Pi, \Lambda)$
$
\Pi = \{p_1, p_2, .. p_n\}$ is a set processes (vertices)
$
\Lambda = \{l_1, l_2, .. l_m\} \subseteq \Pi \times \Pi$ is a set of links (edges)

Processes communicate by message passing

A failure probability $P_i$ is associated with each process $p_i$ and a message-loss probability $L_j$ is associated with each link $l_j$. This probabilities set defines a configuration.
Probabilistic reliability

Problem statement

Validity. If a process broadcasts a message \( m \), then it eventually delivers \( m \).

Agreement. If a process delivers a message \( m \), then every process eventually delivers \( m \) with probability \( K \).

Integrity. For any message \( m \), every process delivers \( m \) at most once, and only if \( m \) was previously broadcast by some process.

Our goal revisited

- We want to provide a solution for the probabilistic reliable broadcast in large-scale systems.
- We want our solution to adapt to changes in the unreliability probabilities of processes & links.
- We want to measure the effectiveness of our adaptive strategy with respect to an algorithm that minimizes the number of messages.
Optimality & adaptation

Optimality. A probabilistic reliable broadcast algorithm O is optimal to some configuration C w.r.t. the number of messages if there is no algorithm X such that processes executing X in C exchange fewer messages than processes executing O in C.

Adaptation. A probabilistic reliable broadcast algorithm A is adaptive to some configuration C if and only if the number of messages exchanged by processes executing A in C in response to a reliable broadcast is eventually equal to the number of messages exchanged by processes executing O in C.

Optimal algorithm (1)

- Use adaptive gossiping, not random gossiping (traditional)
- Example:
  - Failure probability of path 1 = \( L \)
  - Failure probability of path 2 = \( \alpha L \)
    with \( \alpha > 1 \) (path 1 more reliable than 2)
- Probability \( K \) of reaching B from A

\[
K = 1 - L^{k_0/2}(\alpha L)^{k_0/2} = 1 - (\sqrt{\alpha} L)^{k_0}
\]

\[
K = 1 - L^{k_1}
\]
Optimal algorithm (2)

Maximum reliability tree

The maximum reliability tree is a spanning tree containing the most reliable path in G connecting all processes in \( \Pi \).
The reach function

Given a tree $T_i$ and a vector $\vec{m}_i$, the reach function computes the probability that all processes in $T_i$ are reached by at least one message.

$$reach(T_i, \vec{m}_i) = \begin{cases} 1 & \text{if } T_i = \bot \\ \prod_{T_j \in S_i} (1 - (1 - P_j) \times (1 - L_j) \times (1 - P_j))^{m_i[j]} \times reach(T_j, \vec{m}_j) & \text{otherwise} \end{cases}$$

$$reach(T_i, \vec{m}_i) = \prod_{j=1}^{n-1} \left(1 - (1 - P_{\text{pred}(j)}) \times (1 - L_j) \times (1 - P_j)\right)^{m_i[j]}$$

Optimization problem

With $\lambda_j = 1 - (1 - P_{\text{pred}(j)}) \times (1 - L_j) \times (1 - P_j)$, we end up with the following optimization problem:

$$\begin{align*}
\text{minimize} & \quad c(\vec{m}) = \sum_{j=1}^{\lvert \vec{m} \rvert} m[j] \\
\text{subject to} & \quad r(\vec{m}) = \prod_{j=1}^{\lvert \vec{m} \rvert} 1 - \lambda_j^{m[j]} \geq K
\end{align*}$$
The optimize function

We use a greedy algorithm, since our optimization problem is itself greedy

\begin{verbatim}
function optimize(mrt, K)
    \( \bar{m} \leftarrow (1, 1, 1, \cdots, 1) \)
    while \( r(\bar{m}) < K \) do
        let \( \bar{u}_j \) be such that \( \frac{r(\bar{m} + \bar{u}_j)}{r(\bar{m})} \) is maximum
        \( \bar{m} \leftarrow \bar{m} + \bar{u}_j \)
    return \( \bar{m} \)
\end{verbatim}

Optimal algorithm

1: To execute broadcast(\( m \)) do
2: \( mrt_k \leftarrow mrt_k(G, C) \)
3: propagate(\( m, mrt_k, p_k \))
4: deliver(\( m \))

5: when receive \( (m, mrt_j) \) for the first time
6: propagate(\( m, mrt_j, p_k \))
7: deliver(\( m \))

8: function propagate(\( m, mrt_j, p_k \))
9: \( \bar{m}_j \leftarrow \text{optimize}(mrt_j, K) \)
10: for all subtree \( T_i \in S_{j,k} \) do
11:    repeat \( \bar{m}_j[i] \) times
12:        send \( (m, mrt_j) \) to \( p_i \)
Performance of the optimal algorithm

From optimal to adaptive

Our algorithm is proven to be optimal when it has exact knowledge of the system configuration.

By replacing such (unrealistic) exact knowledge with a module that approximates the changing configurations, we get an adaptive algorithm.

As soon as the approximation module converged towards the actual system configuration, we are again optimal.

Optimal algorithm

Adaptive module
Bayesian inference (1)

$$P_B|F[u] = \frac{P_F|B[u] \times P_B[u]}{\sum_{v=1}^{U} P_F|B[v] \times P_B[v]}$$
Bayesian inference (2)

Algorithm 5 Reliability beliefs management
1: Initialization
2: $U \leftarrow 100$ {precision of information}
3: $C_k[p_l].P_{F|B}[u] \leftarrow 0.2$
4: $C_k[p_l].P_F[u] \leftarrow 0.2$

(a) Initial configuration

| $u$ | $C_k[p_l].P_{F|B}[u]$ | $C_k[p_l].P_F[u]$ |
|-----|------------------------|------------------|
| 1   | [0.0 , 0.2]            | 0.2              |
| 2   | [0.2 , 0.4]            | 0.2              |
| 3   | [0.4 , 0.6]            | 0.2              |
| 4   | [0.6 , 0.8]            | 0.2              |
| 5   | [0.8 , 1.0]            | 0.2              |

(b) After a failure suspicion

| $u$ | $C_k[p_l].P_{F|B}[u]$ | $C_k[p_l].P_F[u]$ |
|-----|------------------------|------------------|
| 1   | [0.0 , 0.2]            | 0.04             |
| 2   | [0.2 , 0.4]            | 0.12             |
| 3   | [0.4 , 0.6]            | 0.20             |
| 4   | [0.6 , 0.8]            | 0.28             |
| 5   | [0.8 , 1.0]            | 0.36             |

Adaptation convergence

![Graph showing adaptation convergence](image-url)
Question?