LAB EXERCISES
Distributed Algorithms (IN4150)

Distributed Systems (DS)
Department Software Technology
Faculty EEMCS

Assignment 1

Reliable communication

2023-2024

Reliable Communication: Dolev's algorithm

In this lab assignment, you will implement Dolev's Reliable Communication (RC) algorithm [1] and its five modifications as described by Bonomi et al. [2]. Your implementation enforces RC in a non-fully connected network. Reliable communication is a fundamental requirement in distributed systems that ensures that a message sent by one process is delivered to all other processes in the system. The algorithm is designed to work in a network where up to f of the N processes can be Byzantine. RC requires the following properties:

RC-Validity If a correct process p broadcasts a message m, then every correct process eventually RC-delivers m.

RC-No duplication No correct process RC-delivers a message m more than once.

RC-Integrity If a correct process RC-delivers a message m with sender p_i, then m was previously broadcast by p_i.

Deliverables

Please ensure that the solution adheres to the following requirements:

1. The algorithm is implemented using the Python template and the IPv8 networking library (available on Brightspace).
2. The implementation runs across multiple Docker containers.
3. The network structure and the node behaviour are adjustable (e.g. network size and messages sent).
4. Each node incorporates random delays before sending a message in order to emulate network conditions (e.g. through traffic control).
5. Each node logs events in a **human-readable** fashion (e.g. to a log file or terminal).
6. The code is submitted to the GitLab repository.

The assignment can be split up into three parts:

Part A

We will focus on Dolev's algorithm, which provides reliable communication despite the presence of f Byzantine processes if processes are interconnected by reliable and authenticated communication channels, and if the communication network is at least $(2f + 1)$-connected [1]. Figure 1 illustrates such a network where $f = 1$.

Algorithm 1 *Reliable communication in $(2f+1)$-connected networks (Dolev's protocol) at process p_i*

Parameters:
    f : max. number of Byzantine processes in the system.
Uses: Auth. async. perfect point-to-point links, instance al.

upon event $\langle Dolev, \texttt{Init} \rangle$ **do**
    $delivered =$ **False**
    $paths = \emptyset$

upon event $\langle Dolev, \texttt{Broadcast} \mid m \rangle$ **do**
    forall $p_j \in \texttt{neighbors}(p_i)$ **do**
        trigger $\langle al, \texttt{Send} \mid p_j, [m, [\ ]] \rangle$
    $delivered =$ **True**
    trigger $\langle Dolev, \texttt{Deliver} \mid m \rangle$

upon event $\langle al, \texttt{Deliver} \mid p_j, [m, path] \rangle$ **do**
    $paths.\texttt{insert}(path + [p_j])$
    forall $p_k \in \texttt{neighbors}(p_i) \setminus (path \cup \{p_j\})$ **do**
        trigger $\langle al, \texttt{Send} \mid p_k, [m, path + [p_j]] \rangle$

upon event (p_i is connected to the source through $f+1$ node-disjoint paths contained in $paths$) **and** $delivered = \texttt{False}$ **do**
        trigger $\langle Dolev, \texttt{Deliver} \mid m \rangle$
        $delivered =$ **True**

The idea behind this protocol is to leverage the authenticated channels to collect the labels of processes traversed by a message. Processes use those labels to determine whether they received a message through at least $f+1$ node-disjoint paths and if so deliver it.

Algorithm 1 recalls the pseudocode of Dolev's algorithm. Implement this basic version of Dolev's algorithm. The procedure you employ to verify whether paths are disjoint should be robust against Byzantine behaviours.

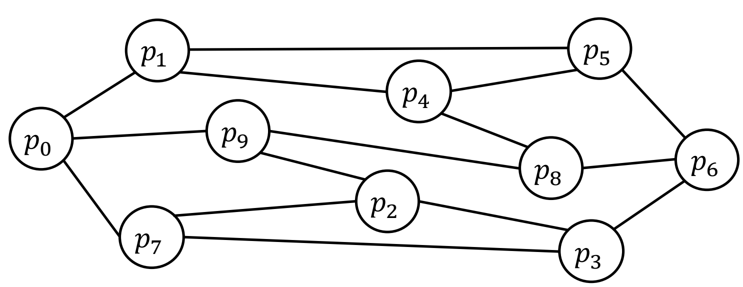

Figure 1: A communication graph with $N = 10$ and node connectivity $k = 3$.

Part B

Bonomi et al. presented several modifications that reduce the number of messages transmitted along with their size in practical executions [2]:

MD.1 If a process p receives a content directly from the source s, then p directly delivers it.

MD.2 If a process p has delivered a message, then it can discard all the related paths and relay the content only with an empty path to all of its neighbours.

MD.3 A process p relays path related to a content only to the neighbors that have not delivered it.

MD.4 If a process p receives a message with an empty path from a neighbour q, then p can abstain from relaying and analyzing any further path related to the content that contains the label of q.

MD.5 A process p stops relaying further paths related to a message after it has been delivered and the empty path has been forwarded.

Implement these modifications and observe their effect on the total number of messages generated.

Part C

Write tests and prepare a demonstration that shows that:

- Your code prevents a malicious process from replaying a message sent by a correct node (RC-Integrity).
- A malicious node can lead only a subset of the correct nodes to deliver a message.

Design three additional similar tests that you will discuss during your final demonstration. Identify the experimental parameters that impact the most the number of messages exchanged.

References

[1] D. Dolev, "Unanimity in an unknown and unreliable environment," in *FOCS*, IEEE, 1981.

[2] S. Bonomi, G. Farina, and S. Tixeuil, "Multi-hop byzantine reliable broadcast with honest dealer made practical," *Journal of the Brazilian Computer Society*, vol. 25, pp. 1–23, 2019.