A Decentralised Self-Healing Approach for Network Topology Maintenance*

JAAMAS Track

Arles Rodríguez
Fundación Universitaria Konrad Lorenz
Bogotá, Colombia
arlese.rodriguezp@konradlorenz.edu.co

Jonatan Gómez
ALIFE Research Group - Universidad Nacional de Colombia
Bogotá, Colombia
jgomezpe@unal.edu.co

Ada Diaconescu
Telecom Paris, LTCI, Institut Polytechnique de Paris
Paris, France
ada.diaconescu@telecom-paris.fr

ABSTRACT

In many distributed systems, from cloud to sensor networks, different configurations impact system performance, while strongly depending on the network topology. Hence, topological changes may entail costly reconfiguration and optimisation processes. This paper proposes a multi-agent solution for recovering a network's topology in case of node failures. The proposed approach relies on local information about the network's topology, collected and disseminated at runtime. Two strategies for distributing topological data are studied: one based on Mobile Agents (our proposal) and the other based on Trickle (a reference gossiping protocol from the literature). These two strategies were adapted for our self-healing approach, to collect topological information for network recovery; and were evaluated in terms of resource overheads. Experimental results show that both variants can recover the network topology, up to a certain node failure rate, which depends on the network topology. At the same time, Mobile Agents collect less information, focusing on local dissemination, which suffices for network recovery. This entails less bandwidth overheads than when Trickle is used. Still, Mobile Agents utilise more memory and exchange more messages during data-collection than Trickle does. These results validate the viability of the proposed self-healing solution, offering two variant implementations with diverse performance characteristics, which may suit different application domains.

KEYWORDS

Complex Networks, Topology Self-healing, Mobile Agents exploration, Trickle gossiping, Decentralised Data Collection

ACM Reference Format:
Arles Rodríguez, Jonatan Gómez, and Ada Diaconescu. 2021. A Decentralised Self-Healing Approach for Network Topology Maintenance: JAAMAS Track. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

In many distributed applications, such as server farms and cloud management, maintaining and recovering servers from failures is essential for meeting stringent requirements of availability and reliability. Since these systems serve millions of users in parallel, downtimes can generate losses of thousands of dollars per minute [11]. System administration expertise is expensive and in short supply, considering distributed system pervasiveness. In addition, experts may be unable to react fast enough for dealing with multiple failures within short periods, e.g. minutes [11]. E.g., in 2017, a disruption in the Amazon s3 service occurred due to a command entered incorrectly that removed a large set of servers causing failures in other important subsystems [1]. Hence, fast automated solutions are needed to enable system *self-healing*.

Moreover, many distributed graph algorithms – e.g. spanning tree calculation [14], vertex colouring [10] or leader election [6] – depend on the underlying network topology and incur important overloads to re-execute if the topology changes. Network topology is also essential for information dissemination, hence impacting traffic dynamics and performance [17]. Finally, topologies can be optimised for maximising network transmission capacity [4], e.g., in clouds, data farms, sensor networks and the IoT.

This paper proposes a generic network self-healing approach based on two decentralised processes: i) a data-collection and dissemination algorithm for gathering knowledge about the network topology; and, ii) a local node recreation and re-connection mechanism for recovering the network topology when nodes fail.

To describe our contribution, we employ the ODD (Overview, Design Concept and Details) Protocol [7], and associated documentation suggestions [8]. ODD facilitates the understanding and clarifies the documentation of agent-based model (ABM) simulations. It starts by defining the purpose of the model, then highlights the main theoretical concepts and their transfer to simulation design, and finally provides the implementation and simulation details.

2 RELATED WORK

The proposed approach mainly differs from existing work in that it aims to recreate and maintain a network's topology, rather than merely its connectivity or diameter. This is feasible in application contexts where nodes can be recreated and reconnected (e.g. spare servers in Data Centres; dormant devices in sensor networks; redundant robots or handhelds in ad-hoc mobile networks).

For instance, [9] proposed a design method for growing both robust and efficient onion-like topological structures. [2] aimed to build a reliable topology based on a fractal cell-structure and compared it with scale-free networks, as defined in [3]. This is complementary to our approach in that we aim to recover the network topology in case of failure. Other approaches aimed to deal with node failures by reconnecting remaining nodes; hence

*This paper is an extended abstract of our full paper in [16]

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

avoiding network splitting into isolated components, which would render the system dysfunctional. E.g., the proposals in [5] and [18] trigger node reconnection to maintain connectivity when a node looses a certain number of neighbours; or, to maintain inter-node distances similar to the original network or at least below a maximum distance. As before, this approach does not recover failed nodes and hence does not preserve network topology.

[13] proposed a self-healing protocol for Software Defined Networks (SDNs). It aimed to maintain a given topology in two steps: 1) using multi-cast for network discovery and state data collection; and 2) using an autonomic failure recovery mechanism. Experiments were performed for Scale-free networks. Our approach goes deeper into exploring and evaluating alternative data-collection mechanisms (based on Trickle and Mobile Agents), to reduce resource overheads. We adapt Trickle and Mobile Agents algorithms to provide nodes with topological data, as necessary to maintain a targeted network topology. Moreover, we analyse the efficiency of these algorithms for different topologies, showing that structural characteristics, though often ignored, do matter.

Trickle is a scaleable and robust algorithm for propagating and maintaining information in low-power, lossy networks (e.g. wireless sensor networks). It was defined under the RFC6206 standard [12], with common applications including traffic timing control, multicast propagation and route discovery. We adopt Trickle as a baseline for evaluating our Mobile Agents approach.

3 NETWORK SELF-HEALING MODEL, DESCRIBED VIA THE ODD PROTOCOL

We propose a generic self-healing approach for maintaining a network's topology as close as possible to the original one, in case of node failures. We evaluate our proposal via a simulation model, implementing two variants of data-collection and dissemination methods: one adopted from our previous work (Mobile Agents) and one from the literature (Trickle) – both were adapted to our self-healing application. Experimental simulations compare the two variants and highlight their advantages and disadvantages in terms of network self-healing abilities and resource overheads. The ultimate goal is to show that topological self-healing is possible via decentralised algorithms, provided that topological knowledge can be gathered in time (i.e. before nodes fail for the first time).

The simulation model defines three types of entities: one external entity representing the *network environment*, and two agent types – *static nodes* and *mobile agents*. The *network environment* represents the network's state during runtime. It defines the network's topology as a graph, which allows evaluating its similarity with the original network topology, to be maintained. The network environment representation also indicates whether a node is *alive* or in a *failed* state. The environment is parametrised with a *failure probability* p_f, which determines the likelihood of node failures at each round. *Nodes* are static agents that represent the different processes (or platforms) of a selected network. Each node has: a unique *id*; a *visited* status flag; a *message queue* for communicating with other node agents; *memory* space for storing topology data; and an *algorithm* (or program) for processing incoming messages, simulating the node's failure, detecting missing neighbours and recreating them. A node's algorithm program is executed at every round, or simulation step. We implemented three variants of this node program, varying in the way in which network topology information is acquired. In the first variant, all topology information is provided statically; in the second one, topology information is propagated via the Trickle algorithm; and, in the third one, via the Mobile Agents approach.

In the Mobile Agents variant, each *mobile agent* is endowed with: *memory space* to store data collected from nodes on the network topology; and an *algorithm* (or program) to communicate with visited nodes, simulate agent failures, and select the next node to visit.

Simulation time is defined via discrete rounds, or steps. In each round, the network environment updates statistics stored about the consumed resources: memory (in bytes) used by nodes and by agents, communication bandwidth (in bytes), and number of messages received by nodes. It also determines which nodes fail, with a probability p_f, corresponding to node crashes in real systems.

Agents are implemented in a multi-agent system simulator (multi-threaded)[1] running on a single machine. In each round, each agent executes its program once. Nodes and mobile agents run in parallel, in random order, and are synchronised at the end of each step, for collecting metrics.

4 EXPERIMENTS AND RESULTS

Via a wide range of simulations, the paper analyses and compares the two algorithms – Trickle and Mobile Agents, adapted for network self-healing – in terms of resource consumption (i.e. communication bandwidth, message numbers and memory overheads) and ability to recover the network from node failures. Results show that the network can recover its topology, up to a maximum node failure rate, provided that the remaining nodes were able to acquire the necessary information about the network topology beforehand. To ensure this, in the initial experiments we started removing nodes only after an initial data-acquisition period estimated to suffice for collecting knowledge of the entire network topology.

In further experiments we showed that the network could also recover completely even if node failures started before nodes could acquire local knowledge about the entire network topology. This is possible because, by their nature, both data-collection algorithms acquire *local* topological knowledge first, then extend this knowledge progressively to the entire network. As the self-healing process uses local topological information, the extended knowledge about the entire network is superfluous. As previous work highlighted the significant impact of network topology on the performance of decentralised algorithms (e.g. data-collection [15]), experiments were carried-out on a diverse selection of network topologies (i.e. shown to make a difference in [15]).

Results show that while Mobile Agents are significantly lighter in terms of bandwidth consumption, they consume more memory and initiate more messages than Trickle during the data-collection process. This makes Multi Agents more suitable for environments where intended modifications of network topology are relatively rare and limited, requiring limited knowledge re-acquisition efforts. These insights allow system designers to select the most suitable algorithm variant depending on their particular application context.

[1] Source code is available at: https://github.com/arleserp/NetworkRecoverySim

REFERENCES

[1] Amazon. [n.d.]. Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region. https://aws.amazon.com/es/message/41926/

[2] Ya Nan Bai, Ning Huang, Lina Sun, and Lei Wang. 2019. Reliability-based topology design for large-scale networks. *ISA Transactions* 94 (2019), 144–150. https://doi.org/10.1016/j.isatra.2019.04.004

[3] Albert-László Barabási and Eric Bonabeau. 2003. Scale-Free Networks. *Scientific American* 288, 5 (may 2003), 60–69. https://doi.org/10.1038/scientificamerican0503-60

[4] Zhenhao Chen, Jiajing Wu, Zhihai Rong, and Chi K. Tse. 2018. Optimal topologies for maximizing network transmission capacity. *Physica A: Statistical Mechanics and its Applications* 495 (2018), 191–201. https://doi.org/10.1016/j.physa.2017.12.084

[5] Lazaros K. Gallos and Nina H. Fefferman. 2015. Simple and efficient self-healing strategy for damaged complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 92, 5 (2015). https://doi.org/10.1103/PhysRevE.92.052806 arXiv:1511.06729

[6] Mohsen Ghaffari and Bernhard Haeupler. 2013. Near optimal leader election in multi-hop radio networks. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 748–766.

[7] Volker Grimm, Uta Berger, Donald L. DeAngelis, J. Gary Polhill, Jarl Giske, and Steven F. Railsback. 2010. The ODD protocol: A review and first update. *Ecological Modelling* 221, 23 (2010), 2760–2768. https://doi.org/10.1016/j.ecolmodel.2010.08.019

[8] Volker Grimm, J Gary Polhill, and Julia Touza. 2013. Documenting social simulation models: the ODD protocol. *Simulating social complexity: a handbook* (2013), 117–34.

[9] Yukio Hayashi. 2016. Spatially self-organized resilient networks by a distributed cooperative mechanism. *Physica A: Statistical Mechanics and its Applications* 457 (2016), 255–269. https://doi.org/10.1016/j.physa.2016.03.090 arXiv:1603.08329

[10] Fabian Kuhn. 2020. Faster Deterministic Distributed Coloring Through Recursive List Coloring. , 1244–1259 pages. https://doi.org/10.1137/1.9781611975994.76

[11] Philipe Lalanda, Julie A Mccann, and Ada Diaconescu. 2013. *Autonomic Computing: Principles, Design and Implementation*. Springer.

[12] Philip Levis, Thomas Clausen, Jonathan Hui, Omprakash Gnawali, and J Ko. 2011. The trickle algorithm. *Internet Engineering Task Force, RFC6206* (2011).

[13] Leonardo Ochoa-Aday, Cristina Cervello-Pastor, and Adriana Fernandez-Fernandez. 2018. Self-Healing Topology Discovery Protocol for Software-Defined Networks. *IEEE Communications Letters* 22, 5 (may 2018), 1070–1073. https://doi.org/10.1109/LCOMM.2018.2816921

[14] Luiz A. Rodrigues, Elias P. Duarte, and Luciana Arantes. 2018. A distributed k-mutual exclusion algorithm based on autonomic spanning trees. *J. Parallel and Distrib. Comput.* 115 (2018), 41–55. https://doi.org/10.1016/j.jpdc.2018.01.008

[15] Arles Rodríguez, Nathaly Botina, Jonatan Gómez, and Ada Diaconescu. 2019. Improving data collection in complex networks with failure-prone agents via local marking. *Journal of Intelligent and Fuzzy Systems* 36, 5 (may 2019), 5081–5089. https://doi.org/10.3233/JIFS-179053

[16] Arles Rodríguez, Jonatan Gómez, and Ada Diaconescu. 2020. A decentralised self-healing approach for network topology maintenance. *Autonomous Agents and Multi-Agent Systems* 35, 1 (2020), 6. https://doi.org/10.1007/s10458-020-09486-3

[17] Remco Van Der Hofstad. 2016. Random Graphs and Complex Networks Vol. I. *Available on http://www. win. tue. nl/rhofstad/NotesRGCN. pdf* I (2016). http://www.win.tue.nl/{~}jkomjath/NotesRGCN2013may.pdf

[18] Tianyu WANG, Jun ZHANG, and Sebastian WANDELT. 2017. Exploiting global information in complex network repair processes. *Chinese Journal of Aeronautics* 30, 3 (2017), 1086–1100. https://doi.org/10.1016/j.cja.2017.03.007