Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Compact routing messages in self-healing trees

Armando Castañeda [a,1], Danny Dolev [b,2], Amitabh Trehan [c,*,3]

[a] Instituto de Matemáticas, UNAM, Mexico
[b] The Hebrew University of Jerusalem, Israel
[c] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK

ARTICLE INFO

Article history:
Received 7 May 2016
Received in revised form 15 November 2016
Accepted 16 November 2016
Available online xxxx

Keywords:
Self-healing fault-tolerant P2P algorithm
Distributed message passing algorithm
Compact Thorup–Zwick routing
Low memory compact algorithm
Compact Forgiving Tree
Internet of things

ABSTRACT

Existing compact routing schemes, e.g., Thorup and Zwick [4] and Chechik [6] often have no means to tolerate failures, once the system has been set up and started. This paper presents, to our knowledge, the first self-healing compact routing scheme. Besides, our schemes are developed for low memory nodes and are compact schemes, meaning they require only $O(\log^2 n)$ bits memory.
We introduce two algorithms of independent interest: The first is *CompactFT*, a novel compact version of the self-healing algorithm Forgiving Tree of Hayes et al. [7] that uses only $O(\log n)$ bits local memory. The second algorithm (*CompactFTZ*) combines CompactFT with Thorup–Zwick's tree-based compact routing scheme [4] to produce a compact self-healing routing scheme. In the self-healing model, the adversary deletes nodes one at a time and the affected nodes self-heal locally by adding few edges. We introduce the *bounded-memory self-healing model*, where the memory each node need to use for the self-healing algorithm is bounded. CompactFT recovers from each attack in only $O(1)$ time and Δ messages, with only $+3$ degree increase and $O(\log \Delta)$ graph diameter increase, over any sequence of deletions (Δ is the initial maximum degree).
Additionally, CompactFTZ guarantees delivery of a packet sent from sender s as long as the receiver t has not been deleted, with only an additional $O(y \log \Delta)$ latency, where y is the number of nodes that have been deleted on the path between s and t. If t has been deleted, s gets informed and the packet is removed from the network. CompactFTZ uses only $O(\log n)$ bits memory for local fields (such as routing tables) and $O(\log^2 n)$ bits for the routing labels, thus requiring $O(\log^2 n)$ bits overall.

* Corresponding author.
E-mail addresses: armando.castaneda@im.unam.mx (A. Castañeda), danny.dolev@mail.huji.ac.il (D. Dolev), amitabh.trehaan@gmail.com (A. Trehan).

[1] Supported partially by PAPIIT-UNAM IA101015. This research was partially done while the first author was at the Department of Computer Science of the Technion, supported by an Aly Kaufman post-doctoral fellowship.

[2] Danny Dolev is Incumbent of the Berthold Badler Chair in Computer Science, The Rachel and Selim Benin School of Computer Science and Engineering Edmond J. Safra Campus. This work was supported by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by The Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

[3] Research partly done as an I-CORE fellow supported by The Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11). This project has also received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 671603 (AllScale).

http://dx.doi.org/10.1016/j.tcs.2016.11.022
0304-3975/© 2016 Published by Elsevier B.V.

1. Introduction

Routing protocols have been the focus of intensive research over the years. Efficient and robust routing is critical in current networks, and will be even more so in future networks. Routing is based on information carried by the travelling packets and data structures that are maintained at intermediate nodes. The efficiency parameters change from time to time, as the network use develops and new bottlenecks are identified. It is clear that the size of the network makes the use of centralised decisions very difficult, and we are close to giving up on maintaining long distance routing decisions. We are a few years before a full-scale deployment of the Internet of Things (IOT), which will introduce billions of very weak devices that need to have routing capabilities. The size of the network and the dynamic structure that will evolve will force focusing on local decisions, pushing for the use of protocols that do not require maintaining huge routing tables.

Santoro and Khatib [1], Peleg and Upfal [2], and Cowen [3] pioneered the concept of compact routing that requires only a minimal storage at each node. Moreover, the use of such routing protocols imposes only a constant factor increase in the length of the routing. Several papers followed up with some improvements on the schemes (cf. Thorup and Zwick [4], Fraigniaud and Gavoille [5], and Chechik [6]). These efficient routing schemes remain stable as long as there are no changes to the network.

The scale and mobility of future networks such as IOT will necessarily lead to continuous changes to the network and regular (possibly isolated) failures of components/devices. Thus, maintaining connectivity and ensuring a smooth running of already running protocols will be important. This is where the self-healing capabilities of the network will be important for efficient, responsive repair of the network without hindering regular operation.

The target of the current paper is to introduce an efficient *compact* scheme that combines small local memory and packet headers with the ability to update the local data structures stored at each node in a response to a change of the network. Throughout this paper, when we say compact, we imply schemes that use $o(n)$ local memory per node and packet headers of size $o(n)$, where n is the number of nodes in the network. Our new scheme has similar cost as previous compact routing schemes. In this work, we will focus on node failures. In general, node failures may be considered more challenging to handle than edge/link failures, and are analysed less often.

Our algorithms work in the *bounded memory self-healing model* (Section 2). We assume that the network is initially a connected graph over n nodes. All nodes are involved in a preprocessing stage in which the nodes identify edges to be included in building a spanning tree over the network and construct their local data structures. After the preprocessing is complete, the adversary repeatedly attacks the network. The adversary knows the network topology and the algorithms and has the ability to delete arbitrary nodes from the network. To enforce a bound on the rate of changes, it is assumed the adversary is constrained in that it deletes one node at a time, and between two consecutive deletions, nodes in the neighbourhood of the deleted node can exchange messages with their immediate neighbours and can also request for additional edges to be added between themselves. The adversary can attack only after the system has self-healed itself from the previous attack. Thus, the self-healing process should be efficient enough so that the system recovers quickly after any possible attack.

Our self-healing algorithm CompactFT ensures recovery from each attack in only a constant time and Δ messages, while, over any sequence of deletions, taking only a constant additive degree increase (of 3) and keeping the diameter as $O(D \log \Delta)$, where D is the diameter of the initial graph and Δ is the maximum degree of the initial spanning tree built on the graph. Moreover, CompactFT needs only $O(\log n)$ local memory. Theorem 4.1 states the results formally.

CompactFTZ, our compact routing algorithm, is based on the compact routing scheme on trees by Thorup and Zwick [4], and ensures routing between any pair of existing nodes in our self-healing tree without loss of any message packet whose target is still connected. Moreover, the source will be informed if the receiver is lost, and if both the sender and receiver have been lost, the message will be discarded from the system within at most twice of the routing time. Our algorithm guarantees that after any sequence of deletions, a packet from s to t is routed through a path of length $O(d(s,t)\log \Delta)$ where $d(s,t)$ is the distance between s and t and Δ is the maximum degree of any node, in the initial tree. Though CompactFTZ uses only $O(\log n)$ local memory, the routing labels (and, hence, the messages) are of $O(\log^2 n)$ size, so nodes may need $O(\log^2 n)$ memory to locally process the messages. Theorem 6.2 states the results formally.

In CompactFTZ, we use a slight variant of Thorup and Zwick. This variant is for no particular reason but to exemplify that our construction to obtain CompactFTZ is not tailor-made for a particular routing protocol but for any *direct* compact routing protocol for trees based on a DFS labelling (as Thorup and Zwick is). A routing protocol is *direct* if the header of a packet is not modified during the routing. Instead of using a concrete protocol, we could have considered an abstract compact routing protocol for trees and derive our construction, however, we use a concrete example to make the exposition concise. The same approach applies to the compact routing algorithm for trees of Fraigniaud and Gavoille [5], or any DFS-based interval routing for trees (see for example [1]), although the latter does not necessarily achieve compactness.

1.1. Related work

CompactFT uses ideas from the *Forgiving Tree* [7] (FT, in short) in order to improve compact routing. The main improvement of CompactFT is that no node uses more than $O(\log n)$ local memory and thus, CompactFT is compact. CompactFT achieves the same bounds and healing invariants as FT, however, taking slightly more messages (at most $O(\Delta)$ messages as opposed to $O(1)$ in FT) in certain rounds. Table 1 compares between the algorithms.