Computer Networks 54 (2010) 2086–2098

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Secure peer sampling

Gian Paolo Jesi [a,*], Alberto Montresor [a], Maarten van Steen [b]

[a] Dip. di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
[b] Dept. of Computer Science, VU University, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Available online 9 June 2010

Keywords:
P2P
Peer sampling
Overlay
Security
Gossip

ABSTRACT

Gossiping has been identified as a useful building block for the development of large-scale, decentralized collaborative systems. With gossiping, individual nodes periodically interact with random partners, exchanging information about their local state; yet, they may globally provide several useful services, such as information diffusion, topology management, monitoring, load-balancing, etc. One fundamental building block for developing gossip protocols is *peer sampling*, which provides nodes with the ability to sample the entire population of nodes in order to randomly select a gossip partner. In existing implementations, however, one fundamental aspect is neglected: security. Byzantine nodes may subvert the peer sampling service and bias the random selection process, for example, by increasing the probability that a fellow malicious node is selected instead of a random one. The contribution of this paper is an extension to existing peer sampling protocols with a detection mechanism that identifies and blacklists nodes that are suspected of behaving maliciously. An extensive experimental evaluation shows that our extension is efficient in dealing with a large number of malicious nodes.

1. Introduction

Would it make sense to implement large-scale collaborative applications like Facebook or Twitter in a completely decentralized way? From a social point of view, the answer is definitely affirmative: the privacy concerns raised by a company whose end-user agreement can be summarized as "all your data belongs to us" are astonishing. From a technological point of view, however, the answer is not so simple: the enormous scale (hundreds of millions of users) and dynamism of such systems pose enormous challenges to developers.

Gossip protocols have proven to be effective in dealing with these challenges, going beyond the basic dissemination services for which they have been originally designed [1] and implementing sophisticated services like topology management, aggregation and monitoring, load-balancing, semantic clustering, etc. [2–5].

A key requirement for gossip protocols is the ability to randomly select gossip partners from the overall system. The *peer sampling* service (PS) satisfies this requirement, by providing nodes with continuously up-to-date *samples* selected uniformly at random from the global node population [6]. Informally, gossip-based PS services work as follows. Each node stores a collection of *node descriptors*, called the (*partial*) *view*. Execution is divided in periodic *cycles* during which each node p selects a node q from its view and initiates a push–pull communication exchange with it: p sends a subset of its own descriptors to q, plus a fresh descriptor of itself, and q replies in the same way. Node p updates its view based on the message received from q, and symmetrically q does the same. Old descriptors are progressively replaced by new ones; this mechanism keeps views continuously up-to-date with respect to node joins and leaves.

* Corresponding author. Address: Dip. di Ingegneria e Scienza dell'Informazione, University of Trento, via Sommarive 14, 37128 Trento, Italy.
E-mail addresses: jesi@disi.unitn.it (G.P. Jesi), montreso@disi.unitn.it (A. Montresor), steen@cs.vu.nl (M. van Steen).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.03.020

Several implementations exist [7,8], that can be distinguished based on how they select the exchange partner (e.g., completely at random or based on a timestamp); how many and which descriptors are exchanged (e.g., all of them or a small subset); how the update operation is performed (e.g., by discarding old descriptors or by swapping).

By interpreting descriptors as edges, the PS service can also be seen as a mechanism to maintain a topology among nodes. In all existing implementations, the resulting topologies present characteristics similar to those of random graphs: small diameter and extreme robustness against partitions. They are even self-repairing in the sense that old descriptors tend to be discarded and information about new nodes is naturally spread through gossip. These properties make them the right platform for gossip protocol development.

An important issue of modern PS services is their potential exploitation by *malicious* nodes (or *attackers* for short). The characteristics of the topology depend on the way descriptors are exchanged; if some of the nodes do not behave according to the protocol, the sample process can be biased toward a specific group of nodes instead of being random; the resulting topology can fail to show the desired properties.

The most important kind of attack that can be pursued against gossip-based PS services is the *hub attack*, where attackers attempt to gain a leading position in the topology (they attempt to become *hubs*), to later exploit their lead to cause havoc to the system, such as performing a DoS attack that leaves the topology in a disconnected state.

For a concrete example of such attack, look at Fig. 1. When the system runs correctly, a random topology is formed, as illustrated in Fig. 1a.

Now assume that a small number f of colluding attackers join the system. Note that f can be as small as the partial view size, which is around 20–30 for most PS services.

Instead of running the regular protocol, the attackers completely ignore the descriptors they receive and keep sending the descriptors of the malicious group members. The partial views in the entire system are progressively polluted by the attackers' descriptors, which keep being generated by malicious nodes and propagated by correct ones. As the percentage of malicious-node descriptors in a partial view grows, the probability of contacting malicious nodes grows proportionally, facilitating their job:

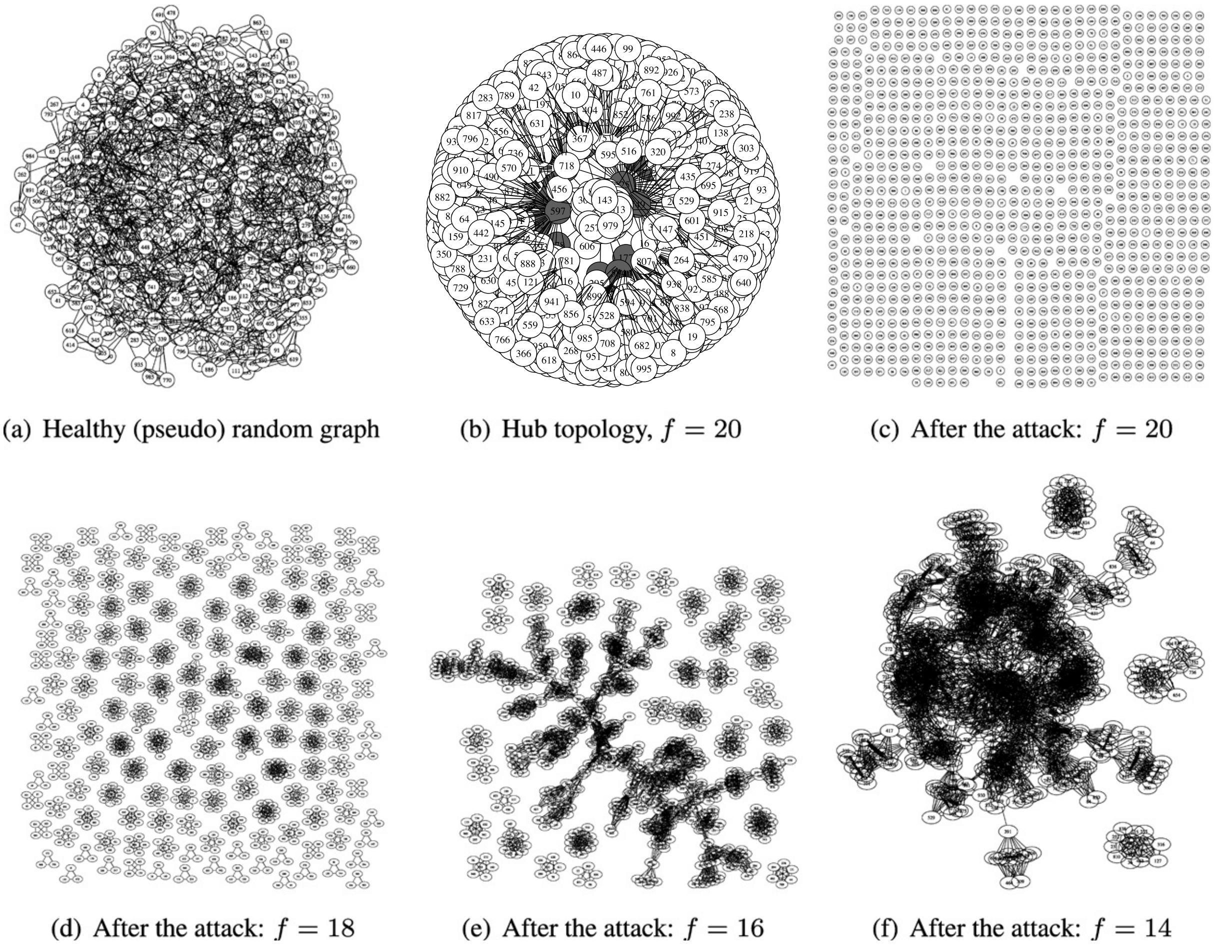
(a) Healthy (pseudo) random graph
(b) Hub topology, $f = 20$
(c) After the attack: $f = 20$
(d) After the attack: $f = 18$
(e) After the attack: $f = 16$
(f) After the attack: $f = 14$

Fig. 1. Overlay topology before (a), during (b) and after (c–f) the attack. The healthy random graph (a) is mutated in a hub-based overlay (b). The graphs (c), (d), (e) and (f) show what happens if the hubs leave the system (with $f = \{20, 18, 16, 14\}$ colluding attackers). Only 3 links per node are displayed for clarity. Network size is 1000 nodes.