On the Security of Peer Sampling Services

ASCI a9 Course Report

By
Srijith K. Nair

Presented to
Advanced School for Computing and Imaging (ASCI)

Vrije Universiteit

February 27, 2007
Amsterdam, The Netherlands

1 Introduction

The rise of large-scale overlay networks in today's networked world has furthered the development of various protocols aimed at ensuring the reliability and robustness of such systems. Gossip [1] is one such paradigm that combines protocol simplicity as well as robustness.

Peer sampling [3] is one of the essential steps involved in any gossip-based protocols. In its simplest form it involves the selection of a random set of nodes from those participating in the overlay network and the subsequent exchange of information between the peers. In order to research various approaches aimed at making this step robust and efficient, previous studies have taken the path of abstracting the functionality into an independent service.

While several algorithms have been proposed to make the peer sampling service (PSS) robust by ensuring a node-degree distribution of a random graph, most of them fail to consider the effect of malicious nodes in the overlay. Unless and until it can been shown that an overlay network can withstand the effect of such a threat model, gossip based approaches would find it difficult to get accepted in the practical world.

The *Secure Peer Sampling Service* (SPSS) proposed by Jesi et al. [2] is one of the very few attempts at addressing this issue. However as shown in this report, SPSS does not solve the problem of malicious nodes completely.

In ∮ 2 we describe the NEWSCAST PSS and then the attack model considered in [2]. We then look at the SPSS proposal and explain why it still remains vulnerable to malicious attacks. In ∮ 3 we go back to the basic building blocks of a PSS protocol and look at how various design considerations influence the system's ability to withstand malicious attacks against it. Using the findings in ∮ 3 we discuss various ways to make the PSS tolerant to malicious attacks, in ∮ 4. We conclude in ∮ 5.

2 Peer Sampling Service

In order to use epidemic protocols to disseminate information, each node would need to know all other nodes in the overlay so as to be able to pick a node to gossip with, at random. However, for a large network to remain scalable, the 'all nodes are known' assumption has to be dropped and the next-best option of maintaining a local partial (size limited) view of its local neighbors has to be considered. The PSS is the background service which maintains this partial view. In order to maintain a robust overlay, the PSS aims to achieve a node-degree distribution that resembles a random graph. This ensures a network with high network connectivity, low node clustering and even node-degree distribution.

Several algorithms for implementing a PSS have been suggested in previous works. We first consider the NEWSCAST protocol mainly because SPSS, which claims to provide a secure version of the PSS, uses it as an example to explain its working.

2.1 NEWSCAST

In this PSS protocol, each node maintains a cache (partial view) of c IDs of other peers in the overlay, augmented with a logical time stamp (ts) representing its creation time. Periodically, each node A (i) selects a random peer B from its cache, (ii) replaces B's entry in the cache with its own ID and updated time stamp, (iii) performs a cache exchange with B (iv) merges the cache received from B with its own, and keeps the c IDs with the freshest time stamp, after removing duplicate ID entries. The protocol is shown to create a topology of almost a random graph with an out-degree of c and is also capable of detecting dead nodes since they can't inject their IDs with updated time stamp. Figure 1 explains the process in detail.

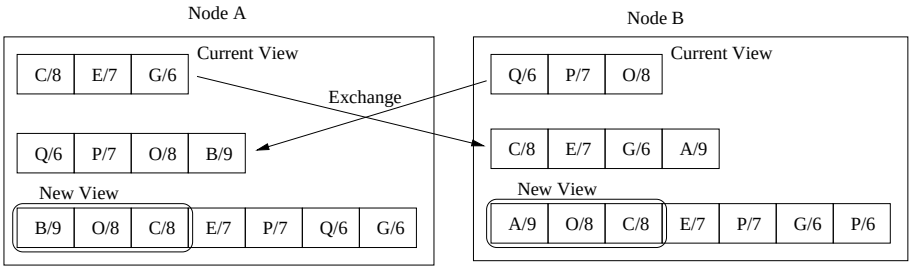

Figure 1: NEWSCAST Protocol

2.2 Attack Model

NEWSCAST and similar PSS protocols work well if most, if not all, nodes obey the rules of the game. However, it is naive to assume such a threat model in an open membership overlay. Jesi et al. [2] considers a more realistic attack model in which k colluding malicious nodes try to form hubs centered around them by using a combination of replay and forgery attacks. A malicious node could propagate wrong protocol-specific information by, for example, creating fresh timestamps of other colluding malicious nodes. When such a 'poisoned' cache is sent to a non-malicious node, due to the (wrong) fresh timestamp associated with the IDs of malicious nodes, the non-malicious node's new cache would contains all the k malicious node IDs. When this non-malicious nodes exchanges cache with another non-malicious node, it would unintentionally pollute this peer's cache too.

Repeated exchange with polluted peers spreads the pollution. As a variant of the attack, the malicious node would fill up $c - k$ cache entries with fake IDs belonging to non-existent peers. Experiments show [2] that such an attack model is very effective in polluting and defeating peers in a very small number of exchange cycles.

When the hubs are formed, the malicious nodes drop out and leave the network partitioned.

2.3 'Secure Peer Sampling Service'

The SPSS protocol was proposed as a PSS that was safe from the attacker model mentioned above. SPSS assumes the existence of a central Certificate Authority (CA) and certified public key for each node. Each cache exchange is signed by the certified public key of the node. The protocol takes a two-pronged approach. When a node P receives a cache from peer Q, it checks a percentage α of the IDs in the received cache to ensure they are genuine members of the overlay. If more than a fixed percentage of these checked IDs turn out to be fake, the exchange is aborted and the newly received cache is discarded. If the cache passes this test, it is compared to the existing cache to check for overlap. Since a malicious node would be expected to replay the same cache, a significant repetition (global parameter threshold β) signals the possibility that the exchanging peer is malicious or polluted[1].

The peer reports this suspicion to a central TRUSTED PROMPT (TP), providing it with the original and received cache. The TP verifies the signature of the received cache and also that the ID matches the IP address of Q. If the checks fail, Q is 'confirmed' as a malicious node and is black-listed. Else, Q's ID is logged into a *frequency table* indicating how many times it has been reported as possibly malicious. TP then builds a cache for P, picking random nodes from the overlay with a probability $p = (1 - \#(X)/overlay\_size)$ where $\#(X)$ is the number of times node X has been recorded as being possibly malicious in the frequency table.

[1]From here on polluted nodes are also considered malicious

2.4 Security of SPSS

Any protocol that uses a CA and assumes that a verifiable certified public key for each node in the overlay exists can be expected to be denounced for its reliance on a pipe-dream of a working Public Key Infrastructure. However SPSS is not such a system. Though not explicitly specified in the original paper [2], it is reasonable to assume that the CA and the TP are the same entity and hence the checking of the signature on the reported cache of Q is a much simpler process that does not rely on P's ability to verify the signature. However, unfortunately, SPSS does suffer from other problems that leaves the protocol open to exploits and attacks.

One of the weakness is that TP blacklists the IDs of peers that fail the signature verification check. This allows a malicious peer M to launch an attack in which it creates a polluted cache and associate the ID of a valid peer A to it and the signs this cache with an *invalid key*. This ensures that the check done by TP would fail and that A, a valid node, would end up in the black-list! Even though the SPSS proposal does not state what this blacklist is to be used for, it is safe to assume that A's presence in the list would lead to a denial of service attacks against A. For example, entries in the blacklist could be ignored when creating the cache that is returned to P. Thus A would never be listed in these returned caches.

Another limitation with SPSS is that it assumes that the TP is able to compose the cache to be returned to P in an efficient manner. In order to produce a random draw of the overlay population TP would need to have a non-partial view of the peer population. By keeping CA and TP as the same entity, TP is able to know the potential upper bound of the network's population but TP would need to rely on a centralised mechanism in order to keep up with the dynamic nature of the membership. This would in turn increase the traffic within the overlay.

We will also see in the next section that the malicious peers can work in a co-ordinated manner to prevent P from noticing a cache replay, which in turn suppresses the trigger needed to contact the TP.

3 Factors Influencing PSS Security

The approach taken by SPSS to augment the basic NEWSCAST protocol by adding a phase to filter and prevent malicious and fake peers from polluting a correctly working node's cache, fails in principle because the security layer was an add-on instead of a feature of the protocol. In order to solve the problem caused by malicious nodes at a more fundamental level, security implications of various design choices of the protocol have to be considered.

3.1 Design Choices

Consider the main algorithm behind the PSS as an unstructured Gossip-based implementation [3].

```
while(){
    wait(T time units)
    p = selectPeer()
    if( push ){
        myDescriptor = (myAddress,0)
        buffer = merge(view,[myDescriptor])
        send buffer to p
    }else{
        send [] to p
    }
    if( pull ){
        receive view_p from p
        view_p = increaseHopCount( view_p )
```

```
        buffer = merge( view_p, view )
        view = selectView( buffer )
    }
}
```

Listing 1: Active thread of PSS protocol

As explained in the original paper, nodes store *descriptors* in its partial view, each node consisting of an address and a hop-count. Each node has two threads - the active thread that initiate exchange with other peers and the passive one that react to incoming events. Listing 1 shows the active thread. 'view' is a list of one descriptor per node, sorted in increasing order of hop count, while 'merge' returns a union of two views, discarding duplicate entries with higher hop count.

The design choices concern the strategy to be used for *selectPeer()*, *push/pull* and *selectView()*. selectPeer() could choose from choosing randomly (rand) from view, choosing peer with lowest hop count (head) or the one with highest hop count (tail). selectView() similarly can follow similar strategy (rand, head, tail) when choosing c peer for its cache from the merged view. Each peer can decide to proactively send its cache to another peer or receive it from another or alternative, send and receive one after the other. Each of these design choices influence the power of the overlay to withstand the attack model mentioned in $\oint$ 2.2.

View selection

Once a possibly polluted cache has been received from node Q, P decides which peers in the merged view makes it into the local cache. If a 'head' or 'tail' policy is followed, Q can instrument the cache (by updating the time-stamp accordingly) it sends to P to ensure that P would choose peers from its cache from the merged view. This increases P's chance of getting polluted. A malicious node could ensure that all other malicious peer IDs will end up in P's final cache. On the other hand, if 'rand' approach is used, Q cannot influence P pick any further than filling the exchanged cache completely with IDs of colluding member nodes and hoping that they get picked by P. If k of the entries in Q's cache is that of other malicious nodes or that of non-existent ones, the probability that all k are present in the new view (the worst pollution) is 1 for head or tail strategy and ${}^{k}C_{k} \bullet^{c} C_{c-k}$ for rand strategy. For example, when c =20 and k=5, this probability is as low as 0.29.

Peer Selection

When a peer wishes to sends its cache (as a part of pushpull or push), it selects a peer from its cache using head, tail or rand method. As in the previous case, the use of head or tail strategy allows a malicious peer to ensure that P always contacts a malicious peer in every exchange cycle. Consider an exchange between P and malicious Q taking placing at t_1. Q, knowing the strategy used by P to choose its peers sends a cache with hop counts assigned in such a way as to ensure that when P performs a selectPeer() at t_2, it would choose a malicious peer R to exchange cache with, which in turn manipulates the cache in a similar to ensure that P chooses another malicious peer S in the next cycle t_3. This strategy can be used by the malicious peers to evade the β tolerance percentage check proposed in the SPSS protocol by distributing the responsibility of sending k malicious peer IDs across Q, R and S as k_1, k_2 and k_3, keeping $k_n < \beta$.

As another attack strategy the malicious Q could also manipulate P to contact a non-malicious peer in the next exchange cycle to ensure that the polluted cache propagates to non-malicious nodes faster.

View propagation

Peers can choose between push, pull or pushpull as their strategy to update the cache. In push, when the active thread of P times out, it selects a peer using the strategy already discussed and sends its cache to Q. When Q replies with its cache, the passive thread receives it, performs a merge and updates the view. In pull, on timeout the active thread selects a peer and sends a null cache to trigger a response from the other side. It then receives the response from Q and performs the merge and updates the cache. This is shown in Listing 1. In parallel, the passive thread waits for a message to arrive. When it does, if P is using pull strategy it sends it current cache to Q and then uses the received cache to update the cache it has. In pushpull, the active and passive thread performs the push and pull related actions.

When P accepts an unsolicited cache from Q, it puts itself into a dangerous position as the protection offered by selecting a save strategy for peer selection is nullified. The safer strategy to follow is not to accept a cache message from Q unless it has been requested by P itself either by sending its cache (pushpull) or by sending a null cache to trigger a cache reply (!push).

3.2 Simulations

In order to confirm the intuitive logic used in the arguments above, we performed some simulations using the PeerSim [4] peer-to-peer simulator. Due to resource constrains, we were only able to simulate the proof of the peer selection recommendation. Consider Figure 2. It shows the state of an overlay of 500 nodes with cache size c=20 in the presence of 20 malicious nodes, after 100 exchange cycles. The non-malicious nodes were performing Shuffle algorithm [5], which uses the tail strategy for peer selection. All the malicious nodes left the network after 25 cycles, leaving the overlay network, as expected, in a disconnected state.

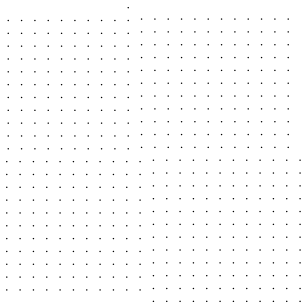

Figure 2: State of overlay network with 500 nodes using shuffle, c=20, k=20. Malicious nodes left the network after 25 cycles.

When the peer selection algorithm was changed to rand, as reasoned above, it was found that the resulting protocol was able to withstand a similar attack. Figure 3 captures the network state after 100 cache exchange cycles, the malicious nodes having left after 25 cycles as before. As we can see, the network remains well-connected.

4 Towards a Safe PSS

As seen from the discussion in §3, protocol design choices effect the ability of the system to withstand the attack from malicious peers trying to pollute its cache. In this section we present some ideas on how to make the protocol safe against such attacks.

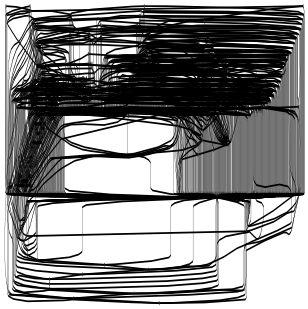

Figure 3: State of overlay network with 500 nodes using rand for peer selection, c=20, k=20. Malicious nodes left the network after 25 cycles.

4.1 Choosing Right Protocol Strategy

The strategy adopted for view selection, view propagation and peer selection influences the security of the protocol. Setting aside the other desired properties like convergent behavior, resulting communication graph, degree distribution and clustering coefficient, the best strategy to adopt, with security in mind would be:

Peer selection: rand
View selection: rand
View propagation: *, but don't use unsolicited cache from other peers

Previous work [3] has however shown that *head* strategy is preferred for view selection since the resulting overlay is much less vulnerable to directed attacks targeting large-degree nodes. Pushpull strategy is also shown to be better than just pull, while push is the worst. In the case of peer selection not much of a difference was seen between different strategies.

In order to incorporate the view selection recommendation of [3] with the finding of ∮3, a compromise strategy as outlined below can be followed. Instead of choosing c random peer IDs from the merge of current and received cache, a peer could first perform a filter based on a threshold value η for the hopcount, dropping all peer IDs below the threshold. The value of η should be chosen in such a way as to obtain a merged cache pool that satisfies the probability requirements discussed in ∮3.

In passing we observe that in shuffle [5], the peer uses tail strategy for peer selection, in order to increase the self healing capacity of the protocol, and replaces its own cache with that received from Q in order to ensure that links are not removed from both the initiating and the receiving node. While this builds a protocol with good random graph and connectivity properties, it is inherently weak against attacks from colluding malicious peers, as [2] has been shown.

4.2 Decoys

The safety provided by choosing an appropriate design strategy may not work in all overlay networks, in particular in those networks where the desired connectivity properties are not met by these choices. There, security need to be augmented into an already chosen protocol design.

Depending on the nature and security requirement of the overlay network, a system of decoy nodes can be used to bait the malicious nodes into divulging their identity. Decoys looks like any other nodes with the difference that they are aware of the identity of the other decoys. Instead of using the peer cache received from other nodes to update its own cache, the decoys use it to try and identify the malicious nodes using statistical means, with the

helps of cache received by other decoys. The identity of the malicious nodes found in this manner can be used in a setup similar to the one suggested by the SPSS protocol. The advantage is that instead of relying on other nodes to report suspicious activity, the decoys can be trusted by the CA to behave as expected (as they are trusted). Furthermore, since the decoys can use powerful analytical methods to find out the identities of the malicious nodes, there are lesser chance of mistakes.

4.3 Recovery

Despite the efforts taken to secure the PSS protocol, there is still the possibility that nodes can become polluted and cause the network to partition. In order to recover from the partition, a mechanism to 'reboot' the disconnected nodes could be considered.

One of the possible ways to do this would be to keep aside the cache (call it *reboot-cache*) that is received from the peer when P joins the overlay via the use of a introducer node [5]. When P sees that it has been disconnected from the overlay network, it could then use this cache to restart the cache exchange mechanism with another peer present in the reboot-cache, using a keyword 'reboot-cache exchange' in its message to let the other peer know that it needs to reply with its reboot-cache even if it is still not in the disconnected state. This potentially allows the node to reset itself to a state when the cache pollution was not rampant.

5 Conclusion

In this report we looked at the security of peer sampling services and showed that the ability to withstand malicious node attacks that tries to pollute the cache depends on the choice of peer selection, view selection and view propagation strategy. We identified the choices that result in a safe system. We also showed that the Secure Peer Sampling Service protocol, proposed as a secure alternative to traditional PSS protocols, is still vulnerable to attacks.

Without going into great detail we also discussed a couple of non-algorithm based approaches to bolster the security of the PSS protocols.

References

[1] M. Jelasity and O.Babaoglu. T-Man: Gossip-based overlay topology management, In *Proc. of Third International Workshop on Engineering Self-Organising Applications (ESOA'05)*, July 2005.

[2] G. P. Jesi, D. Gavidia, C. Gamage and M. van Steen. A Secure Peer Sampling Service, To be published.

[3] M. Jelasity, R. Guerraoui, A. M. Kermarrec and M. van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-based Implementations. In *Proc. of the Fifth Inernational Middleware Conference*, October, 2004.

[4] *PeerSim: A Peer-to-Peer Simulator*, available at http://peersim.sourceforge.net/

[5] S. Voulgaris, D. Gavidia and M. van Steen, CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays, *Journal of Network and Systems Management, Vol. 13, No. 2*, June 2005.