MiNEMA Scientific Report of Visit to the Distributed Systems Group INESC-ID/IST Lisboa

Mouna Allani
Université de Lausanne, Switzerland
E-mail: mouna.allani@unil.ch

1 Introduction

This document reports my visit to the Distributed Systems Group INESC-ID/IST at Lisbon, between the 16th and the 18th of April 2008. The goal of the visit was to come up with a reliable broadcast solution combining the advantages of both the epidemic and the deterministic broadcast approaches. Indeed, gossip, or epidemic, protocols have emerged as a powerful strategy to implement highly scalable and resilient reliable broadcast primitives [10, 5, 7, 2]. In a gossip protocol, when a node wants to broadcast a message, it selects t nodes from the system at random (this is a configuration parameter called *fanout*) and sends the message to them; upon receiving a message for the first time, each node repeats this procedure [10]. Gossip protocols are an interesting approach because they are highly resilient (these protocols have an intrinsic level of redundancy that allows them to mask node and network failures) and distribute the load among all nodes in the system.

Unfortunately, gossip based broadcast protocols are less efficient than other approaches that rely on some sort of deterministic tree-based broadcast to disseminate information [4, 14, 13], as the intrinsic redundancy of gossip protocols produces more network traffic, which might exhaust network capacity, making any sort of broadcast impossible. This is the price to pay in order to avoid the high cost and additional complexity of construction, and also the time costs for repair, such tree-based approaches.
Furthermore, in order to provide some sort of guarantees concerning such high level metrics as latency, adaptive reliability and load-balancing, a gossip approach might not be enough, as one has to take into consideration that peers in such a system have resource limitations (e.g. memory, bandwidth), whereas gossip approaches are random in nature, not taking these constrains into account.

A deterministic broadcast solution, on the contrary, could adapt itself to the available resources. Considering the unreliable behavior of components[1], a deterministic broadcast could define a routing scheme, including the most reliable path (links and nodes) so as to ensure the maximum reliability to reach all processes [6, 1]. Regarding the bandwidth, such a broadcast

[1]Nodes and communication links can fail, unexpectedly ceasing their operation and dropping messages respectively.

solution should distribute the forwarding load with respect to the available bandwidth between any two nodes [3, 8, 1]. The available bandwidth could be expressed in terms of quota of messages to use for one message broadcast [1].

2 Purpose of the visit

The purpose of the visit was to develop a peer-to-peer broadcast solution that ensures *reliability* [2] in a constrained environment. The environment constraints could be related to the dynamic nature of the underlying system, i.e., users can join or leave the system, processes can crash and recover and links can lose messages. They also could be related to the limitation of resources such as a hardware constraints (e.g., processing power or memory limitations), or links constraints (e.g. limited bandwidth).
The way a system takes into account these limitations have a large impact in the overall performance of a broadcast scheme. For this, we defined a solution that defines the reliability as a function of the maximum use of available resources. Furthermore, as mentioned previously, to ensure scalability, a broadcast solution should not be based on a global knowledge about the system. For this, we assume that each process has only a partial view of the system, which is given to him by a peer sampling service such as [9].

For this purpose, we investigated how to combine two previous works. The first [11], named Plumtree, allows one to approach the efficiency of gossip-based approaches to that of tree-based approaches, by building and maintaining a broadcast tree embedded on a gossip-based overlay [3]. The Plumtree protocol operates by sending message payloads preferably via overlay links that are branches of the embedded tree while using the remaining links of the gossip overlay for fast recovery and expedite tree healing.
The second [1], provides a reliable and resources-aware solution that takes into account both the probabilistic behavior of the environment components (processes and links) and the limited bandwidth. This latter is translated into a quota of messages at the disposal of each process for a single message diffusion. The adaptiveness of this solution relies on a tree construction technique that builds progressively[4] a propagation graph covering the whole system. Both [11] and [1] assume that each process has only a partial view about the system. In particular, [11] relies in a peer sampling service called HyParView [12], which is able to provide partial views to nodes that have specific properties (e.g. small sized, symmetry) that allow a gossip-based broadcast scheme to aim at a reliability of 100%.

3 Work carried out during the visit

Hereafter, we describe the resulting work of this collaboration. For a better understanding of this work, we started by defining the model we are considering.

[2] In this context we define *reliability* as the percentage of active nodes that deliver a gossip broadcast.
[3] For instance, the gossip-based overlay created and maintained by the HyParView protocol.
[4] This propagation graph is built collaboratively by some processes.

3.1 Basic system model

We consider an asynchronous distributed system composed of processes (nodes) that communicate by message passing. Our model is probabilistic in the sense that processes can crash and links can lose messages with a certain probability. More formally, we model the system's topology as a connected graph $G = (\Pi, \Lambda)$, where $\Pi = \{p_1, p_2, ..., p_n\}$ is a set of n processes and $\Lambda = \{l_1, l_2, ...\} \subseteq \Pi \times \Pi$ is a set of bidirectional communication links.

Process crash probabilities and message loss probabilities are modeled as *failure configuration* $C = (P_1, P_2, ..., P_n,\ L_1, L_2, ..., L_{|\Lambda|})$, where P_i is the probability that process p_i crashes during one computation step and L_j as the probability that link l_j loses a message during one communication step.

Furthermore, we consider two other constraints in our model: the limited bandwidth and the limited memory available to each process.
Formally, the *limited bandwidth constraint* is modeled as $Q = (q_1, q_2, ..., q_n)$, the set of quotas associated to processes in the system; q_i is the *individual quota of messages* at the disposal of process p_i to forward a message.
In order to take into account the *limited memory* constraint, we further assume that each process maintains knowledge only about its direct neighbors. Formally, the limited knowledge at each process p_i is modeled with two sets N_i and LN_i, which denote, respectively, the set of processes directly connected to p_i and the set of links relying p_i to them. When positioned at p_i, we note $l_k \in LN_i$, the link relying p_i to its neighbor n_k (where $n_k \in N_i$). We assume that knowing about an environment component (link or process) includes knowing all its properties. That is, if a process p_i knows a neighbor n_k it means that p_i knows the n_k reliability, noted P_k, and the n_k quota of messages, noted q_k. It, consequently, also knows l_k, the link relying p_i to n_k and its reliability L_k.

3.2 Resource-Aware Epidemic Broadcast

3.2.1 Solution Overview

Our protocol operates as any pure gossip protocol, in the sense that, in order to broadcast a message, each node gossips with a set of nodes. However, in our strategy, the selected gossip nodes are within direct neighbors and are not randomly chosen. Each node uses a combination of eager push and lazy push gossip. Eager push is used with a subset of neighbors to ensure the broadcast, while lazy push is used for the remaining neighbors. The nodes used for eager push are selected in such a way that their closure effectively builds a reliable broadcast tree embedded in the random overlay network. Lazy push links are used to ensure gossip reliability when nodes fail and also to quickly heal the broadcast tree. Furthermore, contrary to other gossip protocols, the set of peers is not changed at each gossip round. Instead, the same peers are used until failures are detected.

3.2.2 Initialization

For the correct operation of the algorithm, each node classifies its direct neighbors in two sets: the *eagerPushPeers*, with which the node uses eager push gossip of the broadcast message and *lazyPushPeers*, with which it uses lazy push gossip. As already mentioned in our model, nodes are limited in number of payload messages to send for one broadcast. This impacts the number of neighbors to include in *eagerPushPeers* set. For initialization, if enough messages are available, e.g., $q > |N|$, the *eagerPushPeers* is initialized with the set of direct neighbors N. Otherwise, *eagerPushPeers* contains the best reachable neighbors, i.e., connected to the current process with the most reliable links (the lowest L_i). The *lazyPushPeers* will then include the remaining neighbors not included in *eagerPushPeers*.

3.2.3 Tree Construction

After the initialization of the *eagerPushPeers* and *lazyPushPeers* sets described above, nodes construct the maximum probability tree by moving neighbors from *eagerPushPeers* to *lazyPush-Peers* and vice versa, in such a way that, after the protocol evolves, the overlay defined by the first set becomes the aimed reliable tree. When a node receives a message for the first time it includes the sender in the set of *eagerPushPeers*. This ensures that the link from the sender to the node is bidirectional and belongs to the broadcast tree. When a duplicate is received from another neighbor, we check wether it offers a better reliability than the previously sender of the message. In such a case, we switch the new one and the old sender of that message. That is, we add the just received message sender to the *eagerPushPeers* set and we move the corresponding old sender to the *lazyPushPeers*. Otherwise the new sender is moved to the *lazyPushPeers*. Furthermore, a PRUNE message is sent to that sender such that, in response, it also moves the current node to its *lazyPushPeers*.

3.2.4 Misses and Failures Repair

When the quota of messages available at each node is enough to include all neighbors in to the *eagerPushPeers* at the initialization, the above procedure ensures that, when the first broadcast is terminated, a tree has been created. In the other case, however, some nodes risk to not be included in any *eagerPushPeers* set. Hence not covered by the broadcast tree. We name this phenomena as miss. The following algorithm presents a solution for misses and failures that are due to the probabilistic behavior of our model and that may disconnect the tree.

When a failure occurs, at least one tree branch is affected. Therefore, eager push is not enough to ensure message delivered in face of failures. The lazy push messages exchanged through the remaining nodes of the gossip overlay are used both to recover missing messages but also to provide a quick mechanism to heal the multicast tree.

When a node receives an IHAVE message, it simply marks the corresponding message as missing. It then starts a timer, with a predefined timeout value, and waits for the missing message to be received via eager push before the timer expires. The timeout value is a protocol parameter that should be configured considering the diameter of the overlay and a target maximum

recovery latency, defined by the application requirements.

When the timer expires at a given node, that node selects the source of the IHAVE announcement connecting to it via the most reliable link. It then sends a GRAFT message to that source. The GRAFT message has a dual purpose. In first place, it triggers the transmission of the missing message payload. In second place, it adds the corresponding link to the broadcast tree, healing it. When a GRAFT message is sent, another timer is started to expire after a certain timeout, to ensure that the message will be requested to another neighbor if it is not received meanwhile. This second timeout value should be smaller that the first, in the order of an average round trip time to a neighbor.

Note that several nodes may become disconnected due to a single failure, hence it is possible that several nodes will try to heal the tree degenerating into a structure that has cycles. This is not a problem however, as the natural process to build the tree will remove any redundant branches produced during this process by sending PRUNE messages (*i.e.*, when a message is received by a node more than once).

4 Future collaboration

The research discussed during this visit was very fruitful. It also allowed us to identify several common research interests that justify a future collaboration between the **Dop lab** of the University of Lausanne and the **DSG lab** of INESC-ID/IST Lisbon. Regarding our broadcast solution, this visit was, unfortunately, not enough to develop a performance measurement model. We argue that our solution will offer a reliable and scalable broadcast. To prove this, we plan to apply for another visit in order to validate our solution by running several simulations scenarios. Moreover, with this collaboration we aim at submitting a joint paper to a conference, such as the 28th Conference on Computer Communications (INFOCOM 2009).

References

[1] M. Allani, B. Garbinato, F. Pedone, and M. Stamenkovic. Scalable and reliable stream diffusion: A gambling resource-aware approach. In *Proceedings of 26th IEEE Symposium on Reliable Distributed Systems (SRDS'2007)*, pages 288 – 297, Beijing, CHINA, October 2007.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2), May 1999.

[3] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proceedings of ACM SOP'03, October 2003*, October 2003.

[4] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics, June 2000*, pages 1–12, June 2000.

[5] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374, 2003.

[6] B. Garbinato, F. Pedone, and R. Schmidt. An adaptive algorithm for efficient message diffusion in unreliable environments. In *Proceedings of IEEE DSN'04 , June 2004)*, pages 507–516, June 2004.

[7] Mark Hayden and Kenneth Birman. Probabilistic broadcast. Technical report, Ithaca, NY, USA, 1996.

[8] J. Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and Jr James W. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proceedings of OSDI, October 2000*, October 2000.

[9] Märk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[10] Anne-Marie Kermarrec, Laurent Massoulié;, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(3):248–258, 2003.

[11] J. Leitão, J. Pereira, and L. Rodrigues. Epidemic broadcast trees. In *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'2007)*, pages 301 – 310, Beijing, China, October 2007.

[12] J. Leitão, J. Pereira, and L. Rodrigues. Hyparview: a membership protocol for reliable gossip-based broadcast. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 419–429, Edinburgh, UK, June 2007.

[13] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[14] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV*, June 2001.