Gossipping in Message Passing Systems: Seminar Report

Vijay D'silva[1]

Department of Computer Science, ETH Zürich
vdsilva@inf.ethz.ch

1 Overview

The paper presented[2] provides a solution to the gossip problem in message passing systems which uses a bounded number of labels. The solution is convincing and versatile, and uses an interesting data structure and insights but its utility is questionable.

1.1 The gossip problem: issues and motivation

The paper considers a message passing system with processes which can communicate asynchronously on point-to-point channels. A process may send or receive a message. Consider a point in a computation where process p has information (*gossip*) about process r. If p now receives a message from q with gossip about r, p must decide whose gossip is hot. The issues addressed by this paper are:

- Is a solution to this problem possible in the model?
- What restrictions does this solution require?
- Can the overhead required by the solution be bounded?

The problem and the approach adopted to solve it are motivated by the following reasons:

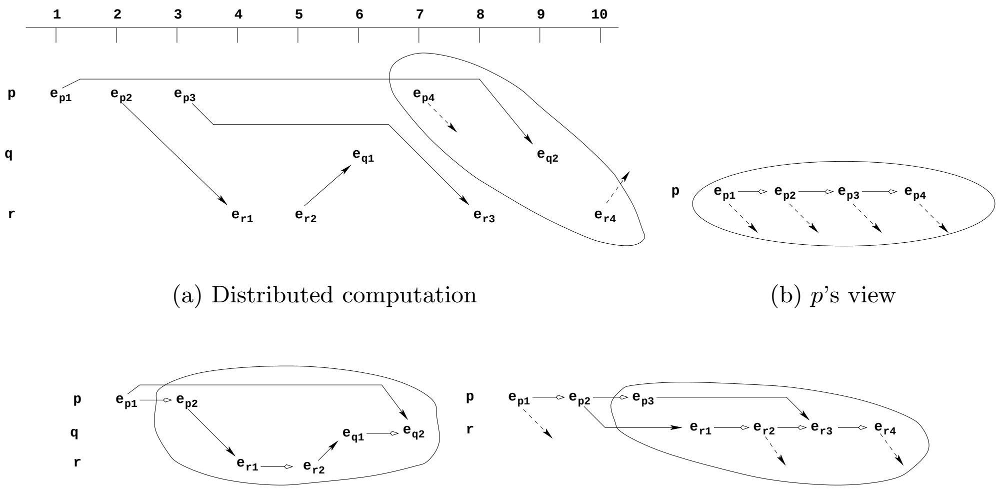
- The problem poses a theoretical question whose answer is known for the shared memory case[1] but not the message passing case.
- A solution to the gossip problem provides a way to compute the global state of the system, information which is important for many distributed algorithms in practise.
- The existence of a solution of the desired form would imply the existence of algorithmic solutions for decision problems relating to correctness of finite-state control systems.

In addition, gossip based protocols are being considered for use in sensor networks and networks-on-chip but populism is not a stated movtivation.

1.2 Restrictions and solution

A solution in a system with an infinite number of messages and an unreliable communication medium is not possible. The problem is made tractable by bounding the number of unacknowledged messages which are sent and requiring that the communication medium does not lose or randomly insert messages. However, unbounded message delays are permitted. The solution involves the following steps:

1. Introduce a data structure for temporally ordering events.
2. Show that processes can infer a part of this structure using information included in the messages they receive.
3. Show that only a part of the data structure is required.
4. Show that the number of unique labels required over time is can be bounded.



(a) Distributed computation (b) *p*'s view

(c) *q*'s view (d) *r*'s view

Fig. 1. Temporal ordering in a message passing system

2 Computing the latest gossip: details

Each process in the system may send or receive a message and a partial ordering of events can be constructed. Figure 1(a) illustrates the system model. Solid arrows indicate a message sent from one process to another and dotted arrows a message that has not been received. Let each process include a header in all its messages with all the information it has about the state of the system. The four steps in the solution mentioned previously are:

1. Construct the causal graph in Figure 1(a).
2. Each process can construct a causal graph shown in Figures 1(b), 1(c) and 1(d).
3. Only the circled part of the graph is sent and its depth is bounded by the number of unacknowledged messages sent.
4. The bound on unacknowledged messages implies the number of *current* events is bounded. To reuse a timestamp, a process has to ensure that the timestamp is not currenty used by any other process. This is achieved by collecting labels from the headers received and doing local book-keeping.

In a system with n processes which send at most b unacknowledged messages, at most $n((2b+1)n+bn^2)$ events are of interest. The system requires at least $(n+bn^2)^2$ unique labels to keep track of the latest gossip.

3 Discussion

I found this paper quite stimulating to read. I was initially disappointed as the result did not seem that exciting but I think it involves many deep insights. In particular, I liked:

– The observation that only a subset of events needs to be considered and its proof is very elegant. (Lemma 5.2)
– The observation that causality information can be inherited. (Proposition 5.5)
– The protocol which I say is versatile because it does not rely on any implicit ordering defined by event labels.

I would criticise the paper on the following grounds:

– Notation. Having only two days to read it before my presentation, understanding their notation was a big obstacle. The authors are aware of this and thank the referees for notational simplifications. Further simplications must be possible.
– Examples. One example is provided in the beginning which is not referred to during the complex constructions and is not representative. I was unable to apply their construction to it as it does not contain many cases of interest.
– The message overhead is rather large especially for application domains like sensor networks.
– They do not sufficiently contrast their solution with those that exist for the shared memory case.

In all, the paper makes an important contribution to what we know about message passing systems. The solution is rather complex but I believe the complexity arises from the general model they consider. Synchronous message passing, disallowing message reordering and additional restrictions may allow for for more effecient solutions. I would recommend the papers on bounded time-stamping in shared memory systems for further reading. It would be interesting to see a comparison of these solutions particularly discussing if there is a fundamental difference between the system models considered.

References

1. D. Dolev and N. Shavit. Bounded concurrrent time-stamp systems are constructible. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 454–466. ACM Press, 1989.
2. Madhavan Mukund, K. Narayan Kumar, and Milind Sohoni. Bounded time-stamping in message-passing systems. *Theoretical Computer Science*, 290(1):221–239, 2003.