Challenges in Evaluating Distributed Algorithms

Idit Keidar
The Technion and MIT
E-mail: `idish@ee.technion.ac.il`

May 20, 2002

1 Introduction

Theoretical evaluation of performance, availability, and reliability of distributed algorithms is always based on models and metrics that make some simplifying assumptions. Making such assumptions is necessary in order to have simple abstractions for reasoning about algorithms. However, such assumptions often lead to models, metrics, and analyses that fail to capture important aspects of actual system behavior. Formulating realistic system models and metrics is important, since distributed algorithms and systems are often designed to optimize over such metrics.

One example is time complexity metrics. The typical theoretical metric used to analyze the running time of distributed algorithms is the number of *communication rounds* the algorithm performs, or the number of message exchange steps in case of a non-synchronous system (e.g., [20, 14, 15]). In Section 3, we illustrate the weakness of this metric.

Another example is reliability metrics. In [13], we highlight the fact that fault tolerant algorithms are often designed under the assumption that no more than t out of n processes or components can fail. This characterization of failures implicitly assumes that the probability of a component failing while a protocol is in progress is independent of the duration of the protocol; that all components that can fail have an identical probability of failure; and that failure probabilities of different components are mutually independent. These assumptions do not adequately reflect the nature of real-world network environments. In practice, the likelihood of t failures occurring while a protocol is running is highly dependent on the protocol's duration. Thus, while consensus protocols that execute more rounds can tolerate more faults, the occurrence of more faults with such protocols is also more likely, which can lead to reduced system availability or reliability, as observed, e.g., in [3, 11].

2 Research Goals

Our goal in the *Dalgeval (distributed algorithm evaluation)* project is to develop realistic ways to evaluate distributed algorithms. We hope that focusing on the "right" metrics will lead to the design of more effective distributed algorithms and systems. Our research approach combines a range of research techniques: gathering of data [4], empirical evaluation [4, 15], and simulation [11, 17], as well as theoretical modeling and analysis [5, 6]. We believe that these techniques complement each other, and when used together can lead to more effective results. E.g., obtaining data on how real environments behave can lead to more realistic theoretical system models and more accurate simulations. However, the transition from data to models is not easy; having gathered data about real systems, it is still challenging to find ways to model this data so it will be easy to reason about.

Another important research effort focuses on obtaining data about how distributed algorithms behave in realistic environments, and then analyzing the data to identify the factors that affect distributed algorithms' performance and availability, and how these factors come into play. Such experiments can teach us which aspects of system behavior are important and ought to be captured in a theoretical system model or metric, and which aspects have little impact and therefore can be simplified out. We give one example of such a research effort in the Section 3.

3 Example: Evaluating the Running Time of a Communication Round over the Internet

It is challenging to predict the end-to-end performance a distributed algorithm would achieve when run over TCP/IP in a wide-area setting. It is also not obvious to determine which algorithm would work best in a given setting. E.g., would a decentralized algorithm outperform a leader-based one? Answering such questions is difficult for a number of reasons. Firstly, performance prediction is difficult because end-to-end Internet performance itself is extremely hard to analyze, predict, and simulate [8]. Secondly, end-to-end performance observed on the Internet exhibits great diversity [18, 22], and thus different algorithms can prove more effective for different topologies, and also for different time periods on the same topology. Finally, different performance metrics can be considered.

In [4], we look at the running time of a communication round over the Internet. We consider a fixed set of hosts engaged in a distributed algorithm. A *communication round* is essentially a black box that propagates information from potentially every host to every other host. Every round is initiated at some host, called the *initiator*. We consider the following four common implementations of a communication round:

- *all-to-all*, where the initiator sends a message to all other hosts, and each host that learns that the algorithm has been initiated sends messages to all the other hosts. This algorithm is structured like decentralized two-phase commit, some group membership algorithms (e.g., [15]), and the first phases in decentralized three-phase commit algorithms, (e.g., [21, 10]).

- *leader*, where the initiator acts as the leader. In this algorithm, the initiator sends a message to all hosts, and all other hosts respond by sending messages to the leader. The leader *aggregates* the information from all the hosts, and sends a message summarizing all the inputs to all the hosts. This algorithm is structured like two-phase commit [9], and like the first two of three communication phases in three-phase commit algorithms, e.g., [21, 12].

- *secondary leader*, where a designated host (different from the initiator) acts as the leader. The initiator sends a message to the leader, which then initiates the leader-based algorithm. This algorithm structure is essentially a spanning tree of depth one, with the secondary leader being the root and all other hosts being leaves.

- *logical ring*, where messages propagate along the edges of a logical ring. This algorithm structure occurs in several group communication systems, e.g., [1].

Using the typical theoretical metric that counts message exchange steps, we get the following overall running times: 2 communication steps for the all-to-all algorithm; 3 for the leader algorithm; 4 for secondary leader; and $2n - 1$ steps for the ring algorithm in a system with n hosts.

In [4] we evaluate these four algorithms over the Internet. Our experiments span ten hosts, at geographically disperse locations – in Korea, Taiwan, the Netherlands, and several hosts across the US, some at academic institutions and others on commercial ISP networks. The hosts communicate using TCP/IP. In contrast to what the communication step metric suggests, we observe that in certain settings the secondary leader algorithm achieves the best overall running time, while all-to-all often has the worst performance. The running time of ring was usually less than double the running times of the other algorithms.

Why does the communication step metric fail to capture the actual algorithm behavior over the Internet? Firstly, not all communication steps have the same cost, e.g., a message from MIT to Cornell can arrive within 20 ms., while a message from MIT to Taiwan may take 125 ms. Secondly, the latency on TCP links depends not only on the underlying message latency, but also on the loss rate. If a message sent over a TCP link is lost, the message is retransmitted after a timeout which is larger than the average round-trip time on the link. Therefore, if one message sent by an algorithm is lost, the algorithm's overall running time can be more than doubled. Since algorithms that exchange less messages are less susceptible to message loss, they are more likely to perform well when loss rates are high. This explains why the overall running time of all-to-all is miserable in the presence of lossy links. Additionally, message latencies and loss rates on different communication paths on the Internet often do not preserve the triangle inequality [19, 15, 2], because routing policies at Internet routers often do not choose the best possible path between two sites.

This explains why secondary leader can achieve better performance by refraining from sending messages on very lossy or slow paths.

One general lesson from our study is that some communication steps are more costly than others. E.g., it is evident that propagating information from only *one* host to all other hosts is faster than propagating information from *every* host to each of the other hosts.

We suggest to refine the communication step metric as to encompass different kinds of steps. One cost parameter, Δ_1, can be associated with the overall running time of a step that propagates information from all hosts to all hosts. This step can be implemented using the most appropriate algorithm for the particular setting where the algorithm is deployed; the results of the study in [4] can help choose the most appropriate algorithm. A different (assumed smaller) cost parameter, Δ_2, can be associated with a step that propagates information from one host to all other hosts. Another cost parameter, Δ_3 can be associated with propagating information from a quorum of the hosts to all the hosts[1], etc.

This more refined metric can then be used to revisit known lower and upper bound results. E.g., [14] presents a tight lower bound of two communication steps for failure-free executions of consensus in practical models. Under the more refined metric, the lower bound is $2\Delta_1$, whereas known algorithms (e.g., [16, 7]) achieve running times of $\Delta_2 + \Delta_3$.

4 Conclusions

Gathering data about Internet characteristics in general, and the behavior of distributed algorithms over the Internet in particular, is extremely important. Such data can be at the basis of more realistic theoretical complexity metrics, and can lead to more effective design of distributed algorithms and systems. We have described a research effort that studies one aspect of distributed algorithm behavior over the Internet; others are yet to be explored.

References

[1] D. A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia. The Totem multiple-ring ordering and topology maintenance protocol. *ACM Trans. Comput. Syst.*, 16(2):93–132, May 1998.

[2] D. G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, Oct. 2001.

[3] Ö. Babaoğlu. On the reliability of consensus-based fault-tolerant distributed computing systems. *ACM Trans. Comput. Syst.*, 5(4):394–416, 1987.

[4] O. Bakr and I. Keidar. Evaluating the running time of a communication round over the Internet. In *ACM Symposium on Principles of Distributed Computing (PODC)*, July 2002. To appear.

[5] Z. Bar-Joseph, I. Keidar, T. Anker, and N. Lynch. QoS preserving totally ordered multicast. In F. Butelle, editor, *5th International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 143–162, December 2000. Special issue of *Studia Informatica Universalis*.

[6] Z. Bar-Joseph, I. Keidar, and N. Lynch. Ealy-delivery dynamic atomic broadcast. Technical Report MIT-LCS-TR-840, MIT Laboratory for Computer Science, April 2002.

[7] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, Mar. 1996.

[8] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001. An earlier version appeared in Proceedings of the 1997 Winter Simulation Conference, December 1997.

[1] In future experiments we intend to evaluate a primitive that waits for responses from a quorum of hosts.

[9] J. N. Gray. Notes on database operating systems. In *Operating Systems: An Advanced Course, Lecture Notes in Computer Science*, volume 60, pages 393–481. Springer Verlag, Berlin, 1978.

[10] R. Guerraoui and A. Schiper. The decentralized non-blocking atomic commitment protocol. In *IEEE International Symposium on Parallel and Distributed Processing (SPDP)*, October 1995.

[11] K. W. Ingols and I. Keidar. Availability study of dynamic voting algorithms. In *21st International Conference on Distributed Computing Systems (ICDCS)*, pages 247–254, April 2001.

[12] I. Keidar and D. Dolev. Increasing the resilience of distributed and replicated database systems. *J. Comput. Syst. Sci. special issue with selected papers from ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS) 1995*, 57(3):309–324, Dec. 1998.

[13] I. Keidar and K. Marzullo. The need for realistic failure models in protocol design. In *4th International Survivability Workshop (ISW) 2001/2002*, March 2002.

[14] I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults – a tutorial. Technical Report MIT-LCS-TR-821, MIT Laboratory for Computer Science, May 2001. Preliminary version in SIGACT News 32(2), pages 45–63, June 2001 (published May 15th 2001).

[15] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. Moshe: A group membership service for WANs. *ACM Trans. Comput. Syst.*, 2002. To appear.

[16] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[17] C. Livadas, I. Keidar, and N. Lynch. The case for exploiting packet loss locality in multicast loss recovery. In preparation, 2002.

[18] V. Paxson. End-to-end Internet packet dynamics. In *ACM SIGCOMM*, September 1997.

[19] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *ACM SIGCOMM*, pages 289–299, September 1999.

[20] A. Schiper. Early consensus in an asynchronous system with a weak failure detector. *Distributed Computing*, 10(3):149–157, 1997.

[21] D. Skeen. Nonblocking commit protocols. In *ACM SIGMOD International Symposium on Management of Data*, pages 133–142, 1981.

[22] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.