1. Challenges in Evaluating Distributed Algorithms

Idit Keidar

Department of Electrical Engineering, The Technion, Haifa 32000, Israel
email: idish@ee.technion.ac.il

1.1 Introduction

Theoretical evaluation of performance, availability, and reliability of distributed algorithms is always based on models and metrics that make some simplifying assumptions. Such assumptions are needed 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. Using 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., [1.20, 1.14, 1.15]). In Section 1.2, we illustrate the weakness of this metric.

Another example is reliability metrics. In [1.13], 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 [1.3, 1.10].

The rest of this white paper is organized as follows: Section 1.2 presents an example of a new research effort that tries to better understand the performance of distributed algorithms over Internet. In Section 1.3, we outline directions for future work. Section 1.4 concludes the discussion.

1.2 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 network. 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 [1.7]. Secondly, end-to-end performance observed on the Internet exhibits great diversity [1.18, 1.22], 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 [1.4], 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., [1.15]), and the first phases in decentralized three-phase commit algorithms, (e.g., [1.21, 1.9]).
- *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 [1.8], and like the first two of three communication phases in three-phase commit algorithms, e.g., [1.21, 1.12].
- *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.
- *logical ring*, where messages propagate along the edges of a logical ring. This algorithm structure occurs in several group communication systems, e.g., [1.1].

Using the typical theoretical metric that counts message exchange steps, we get the following 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 [1.4] 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. We measure each algorithm's *overall running time*, that is, the time that elapses from when initiator initiates the algorithm, and until *all* the hosts terminate. In contrast to what the communication step metric suggests, we observe that all-to-all usually has the worst performance. In cases in which the initiator is a host with good communication links to other hosts, the leader algorithm performs best. If the initiator is a host, like the one in Taiwan, that has poor connectivity to most of the other hosts, then secondary leader algorithm achieves the best overall running time. The typical running time of ring was usually less than double the running times of the other algorithms. As an aside, we note that in case of failures, the all-to-all algorithm is the most robust of the four. The other algorithms may fail to complete in cases of failures that occur while the algorithm is running. Thus, there is a tradeoff between performance and robustness.

Why does the standard 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 [1.19, 1.15, 1.2], 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.

1.3 Future Directions and Research Goals

Our goal in the *Dalgeval (distributed algorithm evaluation)* project is to develop realistic ways to evaluate distributed algorithms. We believe that in order to succeed in this endeavor, a range of research techniques must be used: from gathering of data [1.4], through empirical evaluation in real environments [1.4, 1.15] and simulation using accurate models [1.10], to theoretical modeling and analysis. These techniques complement each other, and when used together can lead to more effective results. Most importantly, obtaining data on how real environments behave can lead to more accurate simulations and more realistic theoretical system models. We propose the following general research directions:

Obtaining data about how distributed algorithms behave in realistic environments. This research effort focuses on obtaining data, 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 gave one example of such a research effort in the Section 1.2; many others are yet to be explored.

Using the gathered data to evaluate a range of algorithms. The gathered data can be used, for example, in trace-driven simulations. Consider the results of the experiments described in the previous section [1.4]. Beyond the specific evaluation of four different distributed algorithms for propagating information, the data gathered in those experiments provides information regarding the nature of communication failures over the Internet, and the correlation among failures over distinct communication paths. This information is useful for evaluating many different kinds of algorithms. Indeed, trace data we gathered in those experiments is currently being used by other researchers [1.11] for evaluating the stability of group membership algorithms such as Moshe [1.15] over the Internet, and the effectiveness of different scalable master-worker algorithms that use group membership, e.g., [1.6, 1.17]. It is our hope that in the future, these traces and others will be used to evaluate various other algorithms.

Formulating better theoretical complexity and reliability metrics. Ultimately, we hope that such empirical results will lead to more realistic theoretical evaluation of distributed algorithms. 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.

We now propose one simple example of how our empirical results can be used to improve the accuracy of theoretical complexity analysis. As noted above, our results show 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. This observation can be leveraged in order to refine the way by which one analyzes algorithm time complexity. 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. 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., [1.14] 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., [1.16, 1.5]) achieve running times of $\Delta_2 + \Delta_3$.

Improving algorithm design. Finally, we hope that focusing on the "right" metrics will lead to the design of more effective distributed algorithms and systems.

1.4 Conclusions

Gathering data about network characteristics and the behavior of distributed algorithms in different networks is extremely important. Such data can be at the basis of more realistic simulations, as well as theoretical complexity and reliability metrics. Ultimately, using better performance metrics can lead to more effective design of distributed algorithms and systems.

References

1.1 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.
1.2 D. G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, pp. 131–145. ACM, Oct. 2001.
1.3 Ö. Babaoğlu. On the reliability of consensus-based fault-tolerant distributed computing systems. *ACM Trans. Comput. Syst.*, 5(4):394–416, 1987.
1.4 O. Bakr and I. Keidar. Evaluating the running time of a communication round over the Internet. In *ACM Symp. on Prin. of Dist. Comp. (PODC)*, July 2002.
1.5 T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, Mar. 1996.
1.6 S. Dolev, R. Segala, and A. Shvartsman. Dynamic load balancing with group communication. In *Intl. Coll. Struct. Inf. and Comm. Complexity*, 1999.
1.7 S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Trans. Networking*, 9(4):392–403, Aug 2001.
1.8 J. N. Gray. Notes on database operating systems. In *Operating Systems: An Advanced Course, LNCS 60*, pp. 393–481, 1978.
1.9 R. Guerraoui and A. Schiper. The decentralized non-blocking atomic commitment protocol. In *IEEE Intl. Symp. on Par. and Dist. Proc. (SPDP)*, Oct 1995.
1.10 K. W. Ingols and I. Keidar. Availability study of dynamic voting algorithms. In *21st Intl. Conf. on Dist. Comp. Sys. (ICDCS)*, pp. 247–254, Apr 2001.

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

1.11 K. Jacobsen, K. Marzullo, and X. Zhang. Private communication, 2002.
1.12 I. Keidar and D. Dolev. Increasing the resilience of distributed and replicated database systems. *J. Comput. Syst. Sci.*, 57(3):309–324, Dec 1998.
1.13 I. Keidar and K. Marzullo. The need for realistic failure models in protocol design. In *4th Intl. Survivability Wshop (ISW) 2001/2002*, March 2002.
1.14 I. Keidar and S. Rajsbaum. On the cost of fault-tolerant consensus when there are no faults – a tutorial. Tech. Rep. MIT-LCS-TR-821, MIT May 2001.
1.15 I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. Moshe: A group membership service for WANs. *ACM Trans. Comput. Syst.*, 20(3):1–48, August 2002.
1.16 L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.
1.17 G. Malewicz, A. Russell, and A. Shvartsman. Optimal scheduling for disconnected cooperation. In *Intl. Coll. Struct. Inf. and Comm. Complexity*, Jun 2001.
1.18 V. Paxson. End-to-end Internet packet dynamics. In *ACM SIGCOMM*, Sep 1997.
1.19 S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *ACM SIGCOMM*, pp. 289–299, Sep 1999.
1.20 A. Schiper. Early consensus in an asynchronous system with a weak failure detector. *Dist. Comp.*, 10(3):149–157, 1997.
1.21 D. Skeen. Nonblocking commit protocols. In *ACM SIGMOD Intl. Symp. on Management of Data*, pp. 133–142, 1981.
1.22 Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *ACM SIGCOMM Internet Measurement Wshop*, Nov 2001.