The Power of Epidemics: Robust Communication for Large-Scale Distributed Systems

Werner Vogels
Dept. of Computer Science
Cornell University
vogels@cs.cornell.edu

Robbert van Renesse
Dept. of Computer Science
Cornell University
rvr@cs.cornell.edu

Ken Birman
Dept. of Computer Science
Cornell University
ken@cs.cornell.edu

ABSTRACT

Building very large computing systems is extremely challenging, given the lack of robust scalable communication technologies. This threatens a new generation of mission-critical but very large computing systems. Fortunately, a new generation of "gossip-based" or epidemic communication primitives can overcome a number of these scalability problems, offering robustness and reliability even in the most demanding settings. Epidemic protocols emulate the spread of an infection in a crowded population, and are both reliable and stable under forms of stress that will disable most traditional protocols. This paper describes some of the common problems that arise in scalable group communication systems and how epidemic techniques have been used to successfully address these problems.

1 INTRODUCTION

Distributed computing will be central to advances in a broad range of critical applications, including intelligence information systems, military command and control, air traffic control, electric power grid management, telecommunications, and a vast array of web-based commercial and government applications. Indeed, a massive rollout of such systems is already underway. Yet while impressive capabilities have been easy to develop and demonstrate in small-scale settings, once deployed these systems often stumble badly.

Software that runs securely and reliably in small-scale mockups may lose those properties as numbers of users, the size of the network and transaction processing rates all increase. Whereas small networks are well behaved, any sufficiently large network behaves like the public Internet, exhibiting disruptive overloads and routing changes, periods of poor connectivity and throughput instability. Failures rise in frequency simply because the numbers of participating components are larger. A scalable technology must ride out such forms of infrastructure instability.

Our studies reveal that very few existing technologies have the necessary properties. Most, including the most prevalent commercial software, exhibit scalability problems when subjected to even modest stress. This finding reveals an imminent (and growing) threat to the full spectrum of emergent mission-critical computing systems. If we can't solve the scalability problem, and develop a methodology yielding applications that remain secure and robust even when failures occur – indeed, even under attack, or during denial-of-service episodes – the very technologies that hold the greatest promise for major advances will prove to be the Achilles Heel of a future generation of mission-critical military and public-sector enterprises.

In the past 4 years the Spinglass project has worked to overcome scalability barriers, starting with an idea that was first proposed in the context of replicated database systems. These systems employed what were called "epidemic-style" or "gossip" update algorithms, whereby sites periodically compare their states and reconcile inconsistencies, using a randomized mechanism for deciding when and with whom each participant will gossip. Traditionally, the database systems used gossip protocols at low speeds. Our work employs gossip at very high speeds, yielding a new generation of protocols that have an unusual style of probabilistic reliability guarantees –guarantees of scalability, performance, stability of throughput even under stress, and remarkable scalability. These properties hold even on Internet-like platforms. Epidemic protocols lend themselves to theoretical analysis, making it possible to predict their behavior with high confidence. However, the focus of our work at Cornell is mostly practical: we are using epidemics, together with other more traditional mechanisms, to develop new generations of scalable communications software and middleware for a wide variety of settings.

In the initial toolset epidemic techniques were used for failure-detection, group communication and distributed state-management. Now that these tools are maturing, we are applying the techniques to other distributed systems areas that face similar scalability challenges, notably the areas of sensor networks and world-wide publish/subscribe systems. Our objective in the present paper is to review the scalability problem for group communication, and to summarize how we used epidemic techniques to solve it. The need for brevity limits the technical detail here, but other publications are available for the interested reader who wishes to learn more.

2 SCALABILITY PROBLEMS IN CURRENT COMMUNICATION SYSTEMS

The scalability of distributed protocols and systems is a major determinant of success in demanding systems. For example, consider the September 2000 field-test of the Navy's Cooperative Engagement Capability (CEC). During a two week period this system (which offers an over-the-horizon cooperative targeting capability for naval battleships) was subjected to a very modest stress test. The value of this system depends upon timely identification of threats and rapid decisions about which ship will respond to which threat: threats may be incoming missiles moving at several times the speed of sound. This translates to internal deadlines of about a second for the communication subsystem, which had been demonstrated easily capable of meeting the requirements under laboratory conditions with small numbers of participating computing systems. Yet under load, when even small numbers of battleships were added to the system, the underlying Data Distribution System (DDS) became unstable, failing outright, or delivering data after much more than the one-second threshold. (Defense News, October 16, 2000). The result was that the CEC basically failed – and this under rather benign conditions in which the only variable that changed was the number of participants.

This paper focuses on scalability of distributed protocols providing some form of guaranteed reliability when communication among multiple participants is required. Use of these reliable group communication (multicast) protocols is very popular in distributed systems, as there is a natural match between the group paradigm and the way large distributed systems are structured. These protocols allow systems to be built in pure peer-to-peer manner, removing the need for centralized servers, removing one of the bottlenecks in system scalability.

Traditional reliable multicast protocols, including those developed by our research group, all exhibit severe scaling problems, especially when applied in large Internet-style settings. Even though some of these protocols appeared to be very promising in terms of scalability they all failed to operate as soon as the network conditions were no longer ideal. In general the performance analyses of the protocols only focuses on two extreme cases: performance of the protocol under ideal conditions, when nothing goes wrong, and the disruptive impact of a failure. Reality forces use to take a look at these protocols from a different perspective: what happens to these protocols under mundane transient problems, such as network or processor scheduling delays and brief periods of packet loss. One would expect that reliable protocols would ride out such events, but we find that this is rarely the case, particularly if we look at the impact of a disruptive event as a function of scale. On the contrary, reliable protocols degrade dramatically under this type of mundane stress, a phenomenon attributable to low-probability events that become both more likely and more costly as the scale of the system grows.

Because of the need for brevity, we'll limit ourselves to a summary of our finding with respect to the growth rate of disruptive overheads for a number of widely used multicast protocols. Elsewhere [3], we present a more detailed analysis of a variety of scenarios, modeled after the work of Gray *et. al.* [4], where a similar conclusion is reached with respect to database scalability. It is clear that scalability represents a widespread problem affecting a broad range of technologies and systems.

2.1 Common Problems

When subjected to transient problem that are related to scaling the environment, there are important categories of problems that appear:

- *Throughput Instability*. All of the protocols that implement reliable group communication are subject a breakdown of message throughput as soon as one or more members experience perturbation. A single slow receiver, which can be cause by CPU overhead or some localized message loss, will eventually have an impact on the throughput to the overall group. The results in [3] show that this impact is even more dramatic and happens more rapidly if we scale up the system, making the protocol stability extremely volatile under even moderate scaling conditions.

- *Micropartitions.* In reaction to the throughput instability problem, designers often go for the approach to as soon as possible to remove the trouble-causing member from the group by using more aggressive failure detection mechanisms. However when scaling the system to moderate Internet environments, one quickly discovers that this has the adverse effect that transient network problems, which occur frequently, frequently trigger incorrect failure-suspicions. Erroneous failure decisions involve particularly costly "leave/rejoin" events, where the overall system constantly needs to reconfigure itself. We will term this a *micropartitioning* of the group, because a non-crashed member effectively becomes partitioned away from the group and later the partition (of size one) must remerge. In effect, by setting failure detection parameters more and more aggressively while scaling the system up, we approach a state in which the group may continuously experience micropartitions, a phenomenon akin to thrashing.

  Costs associated with micropartitions rise in frequency with the square of the size of the group. This is because the frequency of mistakes is at least linear in the size of the group, and the cost of a membership change is also linear in the group size: a quadratic effect.

- *Convoys*. An obvious response to the scalability problems just presented is to structure large systems hierarchically, as trees of participant groups. Unfortunately this option is also limited by disruptive random events, albeit in a different way. Experimentation has shown that such a tree structure, when confronted with moderate network instability, exhibits an amplification of the burstiness of the message traffic. Even though messages enter the system at a steady rate, the reliability and buffer strategies at each of the intermediate nodes in the trees have compressing effects on messages rates, especially when confronted with message loss. The term "convoy" has been used by the database community to describe this phenomenon, which is also well known to the packet routing community.

2.2 Unsuccessful Solutions

Some reliable multicast protocols have been successful at larger scale, but only by limiting the functionality of the protocols. Techniques the developers have resorted to achieve scalability are:

- *Anonymous Membership*. The protocol basically streams out information to whoever wants to receive messages, without any notion of admission control, failure detection or session state management.
- *Single Sender Groups*. These protocols build a single dissemination tree per sender where each node is a potential receiver who cooperates in localized retransmission schemes. Groups with multiple senders, which are very common in distributed systems, are treated as multiple groups with single sender, requiring an explosion of state managed at each participant, and making it impossible to correlate messages from different senders.
- *Infinite Retransmission Buffers*. One of the more complex problems in multi-sender protocols is the management of the messages stored for retransmission. The messages can be released once the system is certain that no retransmission requests can arrive any more. Given that many protocols have no knowledge about the receivers, this certainty can never be achieved and they resort to a method called application level framing to require the application to reconstruct message for retransmission. This was applicable to some multi-user collaboration tools, but was unusable for the majority of distributed systems.
- *Complete Lack of Security*. In most protocols that combine application level framing with localized retransmission (others than the sender can retransmit lost messages), it is impossible guarantee secure communication, as nodes that retransmit can not sign the message with senders key. The original messages are not kept in lack of a garbage collection protocol, and the retransmission node cannot sign it with its own key as they are anonymous, and as such the receiver has no mechanism for checking the signature.

These techniques are unacceptable if one wants to build robust, reliable, secure distributed systems that can be the basis for the mission critical enterprise systems.

But the picture is not entirely bleak. After presenting these arguments, we shift attention to a new class of protocols based on an idea from Clearinghouse [2], the database replication technology developed at Xerox Parc in the 1980's, and later used by Golding for the refDBMS system [5], and from NNTP, the gossip-based algorithm used to propagate Usenet "news" in the Internet. These turn out to be scalable under the same style of analysis that predicts poor scalability for their non-gossip counterparts.

3 EPIDEMIC TECHNIQUES FOR SCALABLE MULTICAST PROTOCOLS

Not all protocols suffer the behavior seen in these reliability mechanisms. In the class of reliable multicast protocols, epidemic multicast protocols scale quite well and easily rides out the same phenomena that cause problems with these other approaches to reliability and scalability.

For example, *Bimodal Multicast*, a protocol reported in [1], uses an epidemic control protocol that somewhat resembles the old NNTP protocol, but is running at much higher speeds. The multicast protocol consists of two sub-protocols. One of them is an unreliable data distribution protocol similar to IP multicast, or based on IP multicast when available. Upon arrival, messages are buffered, and they are delivered to the application layer in FIFO order. The buffered messages are garbage collected after some period of time.

The second sub-protocol, is based on epidemic techniques, and is used to repair gaps in the message delivery flow and to assist the garbage collection. It operates as follows: Each process in the system maintains a list containing some random subset of the full system membership. In practice, we weight this list to contain primarily processes from close by – processes accessible over low-latency links – but these details go beyond the scope of this paper. At some rate (but not synchronized across the system) each participant selects one of the processes in its membership list at random and sends it a *digest* of its current message buffer contents. This digest would normally just list messages available in the buffer: "messages 5-11 and 13 from sender *s,* …" for example. Upon receipt of a *gossip* message, a process compares the list of messages in the digest with its own message buffer contents. Depending upon the configuration of the protocol, a process may *pull* missing messages from the sender of the gossip by sending a retransmission *solicitation*, or may *push* messages to the sender by sending unsolicited retransmissions of messages apparently missing from that process.

This simplified description omits a number of important optimizations to the protocol. In practice, we use gossip not just for multicast reliability, but also to track system membership and perform failure detection based on it [6]. We sometimes use unreliable multicast with a regional TTL value instead of unicast, in situations where it is likely that multiple processes are missing copies of the message. A weighting scheme is employed to balance loads on links: gossip is done primarily to nearby processes over low-latency links and rarely to remote processes, over costly links that may share individual routers [11]. The protocol switches between gossip pull and gossip push, using the former for "young" messages and the latter for "old" ones. Finally, we don't actually buffer every message at every process; a hashing scheme is used to spread the buffering load around the system, with the effect that the average message is buffered at enough processes to guarantee reliability, but the average buffering load on a participant decreases with increasing system size.

An epidemic-style protocol has a number of important properties: the protocol imposes constant loads on participants, is extremely simple to implement and rather inexpensive to run. More important from the perspective of this paper, however, such a protocol overcomes the problems cited earlier for other scalable protocols. Bimodal Multicast has tunable reliability that can be matched to the needs of the application (reliability is increased by increasing the length of time before a message is garbage collected, but this also causes buffering and I/O costs to rise). The protocol gives very steady data delivery rates with predictable, low, variability in throughput. For real-time applications, this can be extremely useful. And the protocol imposes constant loads on links and routers (if configured correctly), which avoids network overload as a system scales up. All of these characteristics are preserved as the size of the system increases.

4 OVERCOMING LIMITATION TO SCALE

We can generalize from the phenomena enumerated above. Distilling these down to their simplest form, and elaborating slightly:

- With the exception of the epidemic protocols, the different reliability model of group communication systems involves a costly, but infrequent fault-recovery mechanism:
  - Virtual synchrony based protocols employ flow control, failure detection and membership-change protocols; when incorrectly triggered, the cost is proportional to the size of the group.
  - Local repair based protocols have a solicitation and retransmission mechanism that involves multicasts; when a duplicate solicitation or retransmission occurs, all participants process and transmit extra messages.
  - FEC-based reliability mechanisms try to reduce retransmission requests to the sender by encoding redundancy in the data stream. As the group size grows, however, either the average multicast path length increases, hence so too the risk of a multi-packet loss. The sender will see increasingly many retransmission requests (consuming a scarce resource), or the redundancy of the stream itself must be increased (resulting in a bandwidth degradation and a system-wide impact).
- Again with the exception of the epidemic protocols, the mechanisms we've reviewed are potentially at risk from convoy-like behaviors. Even if data is injected into a network at a constant rate, as it spreads through the network, router scheduling delays and link congestion can make the communication load bursty. Under extreme condition this behavior can even trigger message loss at the end nodes. To smoothen burstiness of messages from multiple senders one needs a global view of the system, which most reliable protocols have found impossible to implement. Epidemic techniques however are ideal to implement this global state sharing and allow the overall system to gracefully adapt to changes in the network.
- Many protocols depend upon configuration mechanisms that are sensitive to network routing and topology. Over time, network routing can change in ways that take the protocol increasingly far from optimal, in which case the probabilistic mechanisms used to recover from failures can seem increasingly expensive. Periodic reconfigurations, the obvious remedy, introduce a disruptive system-wide cost.

In contrast, the epidemic mechanisms used in NNTP, the Xerox Clearinghouse system, the Astrolabe state management systems [7,8], and Bimodal Multicast protocol appear to scale without these kinds of problems. Throughput is stable if measured over sufficiently long periods of time – gossip protocols can be rather *unstable* if metered on a short time scales. Overheads are flat and predictable, and can be balanced with information about network topology, so that links and routers won't become overloaded. And, the levels of reliability achieved are very high – indeed, potentially as high as those of the protocols purporting to offer stronger guarantees.

Probabilistic guarantees may sound like a contradiction in terms, because one's intuition suggests that anything but an absolute reliability guarantee would be the equivalent of no reliability at all. Our work suggests that this is not at all the case. First, it is possible to design mechanisms that have stronger guarantees, such as virtual synchrony, and yet reside in an end-to-end manner over the basic network architecture afforded by our gossip infrastructure.

An important observation, contributing to the overall success of this approach, is also that the epidemic tools exploit scalability, and as such turn scale into an advantage instead of a problem that must be overcome.

5 OTHER APPLICATIONS OF EPIDEMIC TECHNIQUES

We have applied the epidemic techniques to develop a large set of protocols that each has excellent scalability properties for their target domains. This reaches from multicast and failure detection, through state-management and data-fusion to ad-hoc routing protocols and protocols for controlling power grids. An extensive list of relevant publications can be found at:

http://www.cs.cornell.edu/Info/Projects/Spinglass.

In our current activities we are expanding the application of epidemic techniques to other areas where scalability plays an important role. Two of these areas are power and coverage aware communication for sensor networks [9] and world-wide peer-to-peer publish-subscribe networks [10].

This research is supported by DARPA/ONR under contract N0014-96-1-10014, by the National Science Foundation under Grant No. EIA 97-03470 and by grants from Microsoft Research.

REFERENCES

[1] Birman, Ken, Hayden, Mark, Ozkasap, Oznur, Xiao, Zhen, Budiu, Mihai and Minsky, Yaron "Bimodal Multicast" ACM Transactions on Computer Systems, Vol. 17, No. 2, pp 41-88, May, 1999.

[2] Demers, A. et. al. Epidemic Algorithms for Replicated Data Management. Proceedings of the 6th Symposium on Principles of Distributed Computing (PODC), Vancouver, Aug. 1987, 1-12.

[3] Gupta, Indranil, Birman, Ken, and van Renesse, Robbert, "Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits", Special Issue of Quality and Reliability of Computer Network Systems, Journal of Quality and Reliability Engineering International, May/June 2002, Vol. 18, No. 3, pp 165-184

[4] Gray, Jim, Helland, Pat, O'Neil, P., and Shasha, D., The dangers of replication and a solution. Proceedings 1996 SIGMOD Conference, June 1996.

[5] Golding, Richard and Taylor, Kim,. Group Membership in the Epidemic Style. Technical report UCSC-CRL-92-13, University of California at Santa Cruz, May 1992.

[6] van Renesse, Robbert, Minsky, Yaron, and Hayden, Mark, “A Gossip-Based Failure Detection Service”, in the Proceedings of Middleware '98. England, August 1998.

[7] van Renesse, Robbert, and Birman, Kenneth P, “Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining”, Submitted to ACM TOCS, November 2001

[8] van Renesse, Robbert, and Birman, Kenneth P, Dumitriu, Dan and Vogels, Werner, “Scalable Management and Data Mining Using Astrolabe” in the. Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS),. Cambridge, Massachusetts. March 2002.

[9] Robbert van Renesse, Power-Aware Epidemics, In the Proceedings of the International Workshop on Reliable Peer-to-Peer Systems, Osaka, Japan. October 2002.

[10] Werner Vogels, Chris Re, Robbert van Renesse and Ken Birman, .”A Collaborative Infrastructure for Scalable and Robust News Delivery”. In the Proceedings of the IEEE Workshop on Resource Sharing in Massively Distributed Systems (RESH'02), Vienna, Austria, July 2002.

[11] Xiao, Zhen and Birman, Ken. A Randomized Error Recovery Algorithm for Reliable Multicast. In the Proceedings of FTCS 2001. July 2001.