Sources of Instability in Data Center Multicast

Dmitry Basin
Technion
Haifa, Israel
sdimbsn@tx.technion.ac.il

Ken Birman
Cornell University
Ithaca, NY
ken@cs.cornell.edu

Idit Keidar
Technion
Haifa, Israel
idish@ee.technion.ac.il

Ymir Vigfusson
IBM Research, Haifa Labs
Haifa, Israel
ymirv@il.ibm.com

I. Introduction

Data centers, and particularly the massive ones that support cloud computing, e-commerce, social networking and other large-scale functionality, necessarily replicate data. They do this for a great many reasons: to provision financial servers that respond to incoming read-mostly requests, to parallelize computation, to cache important data items, for fault-tolerance, and the list goes on.

Our basic premise is that since updates to replicated data can be thought of as reliable multicasts, data center multicast is a potentially important technology. Nonetheless, a series of recent keynote speeches at major conferences makes it clear that data center multicast is a troubled area [9, 4, 2]. One might expect such technologies to use IP multicast hardware, but in fact this is rare, mostly because of concerns about flow control [8, 4]. Those concerns reflect the ease with which tens or hundreds of thousands of 1Gbps endpoint nodes can overwhelm a central router or an unlucky receiver, which will drop packets [11]. Only TCP is really trusted today (because it backs down when loss occurs), and indeed, TCP is the overwhelming favorite among data center transport protocols [5, 2]. Using TCP to get reliable multicast with high throughput produces an implicit *TCP overlay tree*.

We make the following contributions. First, we analyze the use of TCP overlay trees for reliable multicast, and find that infrequent, short delays of a tenth of a second or less may cause significant (up to 90%) throughput degradation once trees become large (10^4-10^5 nodes; a size not unreasonable in cloud settings). Nodes can experience such *disturbances* for a variety of reasons, such as Java garbage collection pauses, Linux scheduling delays or flushing data such as logs to disk. The degradation occurs even if message loss is negligible. Under default TCP configurations in low-latency networks, message loss can cause problems also in very small trees: we demonstrate that multicast throughput collapses with as few as $30-100$ nodes if switches become congested. Next, assuming that processing delays follow a power-law distribution, we argue that irrespective of the reliability scheme used, the delivery latency scales almost linearly in the number of receivers.

Our paper explains why overlay trees built from reliable peer-to-peer links are quite likely to perform poorly, at least if implemented naïvely. We are not claiming that data center multicast must inherently be slow; indeed, approaches to overcome the limitations shown by our analysis exist. Rather, our goal is to address the discrepancy between optimistic theoretical predictions and the dire results obtained in the real world: we show that infrequent short disturbances, which were never modeled or simulated, can lead to correlated delays and be extremely disruptive in the context of a reliable protocol. We achieve this by refining the theoretical model for reasoning about such systems so as to capture phenomena that have a severe impact on performance in practice, and were previously overlooked. This approach, in turn, can be used to analyze future work that remedy the problems we highlight.

II. Multicast model

The multicast system consists of N nodes, $1, \dots, N$, placed in a high throughput low latency network with maximal bandwidth T_{max}. Like many previous works [1, 10, 6, 3], we assume that the nodes are structured into a balanced k-ary tree, and the root of the tree is the source of the multicast. The edges of the tree are reliable connections between nodes. Every internal node has one incoming buffer and outgoing buffers for all child nodes. The incoming buffer is, e.g., a TCP receive buffer for the upstream link, and the outgoing buffers can be, e.g., TCP send buffers for downstream links. The root does not have an incoming buffer and the leaf nodes do not have outgoing buffers. All incoming and outgoing buffers are of equal size, denoted by B_0. The maximal size of all buffers on a path from the root to a leaf of the tree is denoted by $B_{max}(N) = B_0 \cdot \Theta(\log N)$.

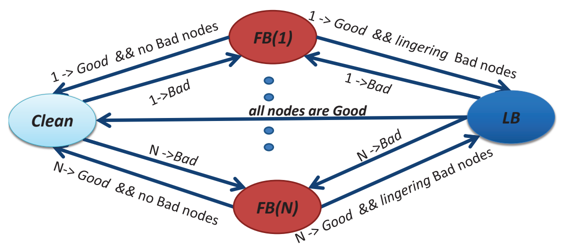

FB(1)
1 -> Good && no Bad nodes
1 -> Good && lingering Bad nodes
1->Bad
1 ->Bad
Clean
all nodes are Good
LB
N ->Bad
N ->Bad
N-> Good && no Bad nodes
FB(N)
N -> Good && lingering Bad nodes

Fig. 1: ***Model.*** *State machine modeling the scenarios considered in the analysis.*

We use a common model for multicast with reliable communication channels and flow-control [1, 10] in which the application on the root node transmits packets, intermediate nodes forward them without any application-level buffering (which would otherwise be equivalent to having larger TCP buffers), and the threads at the leaf nodes consume the packets. Threads will stall to prevent overflow of outgoing TCP buffers, whereas the flow-control mechanism will prevent overflow of incoming TCP buffers.

III. Analytic model

In this section we model the system dynamics and illustrate oscillatory behavior. We introduce node states to model local node events, and system states to model global system behavior.

The key property captured in our model is *disturbances* that nodes may experience. Disturbances prevent nodes from forwarding packets. This can happen for various reasons, e.g., when the application thread does not respond or when packets do not reach the node because of a link problem. By modeling disturbances we capture dependencies among delays of messages that arrive close together. This differs from traditional models, which simply assume that message delays are independently sampled from some distribution (often exponential).

In order to model disturbances, we define two possible states of a node: *Good* and *Bad*. Nodes that experience a disturbance are in the *Bad* state, whereas nodes without disturbances are in the *Good* state. Every node in the multicast tree alternates between these states. Nodes in the *Good* state forward packets from upstream nodes to downstream nodes as described in the multicast model, whereas nodes in the *Bad* state do not forward packets. We assume that the time spent in a *Good* state is exponentially distributed; in a *Bad* state we assume an arbitrary distribution with finite mean.

The throughput of the multicast system depends on the dynamics of local node states. To capture these dynamics, we introduce global system states. The global states transitions are described in Figure 1. The state of the multicast system is called *Clean* if there are no *Bad* nodes. When a node i becomes *Bad* in a *Clean* state, the system transitions to state *FB(*i*)* (i is the "First *Bad*" node), which ends when i returns to a *Good* state. If at the end of the *FB(*i*)* state there are no *Bad* nodes, the system returns to a *Clean* state. Otherwise, after i becomes *Good*, there are additional (lingering) *Bad* nodes in the system, the system transitions to state *LB* (with "Lingering *Bad*" nodes). The *LB* state continues until one of two events occurs:

- All current *Bad* nodes become *Good*. In this case, the system passes to a *Clean* state.
- Some node j becomes *Bad*. In this case, the system passes to *FB(*j*)* (possibly $i = j$).

An interval of time from the beginning of an *FB()* state till the beginning of the successive *FB(*)* state is called a *period*. All possible period sequences are presented in Figure 2, where we also schematically depict the system's oscillatory behavior.

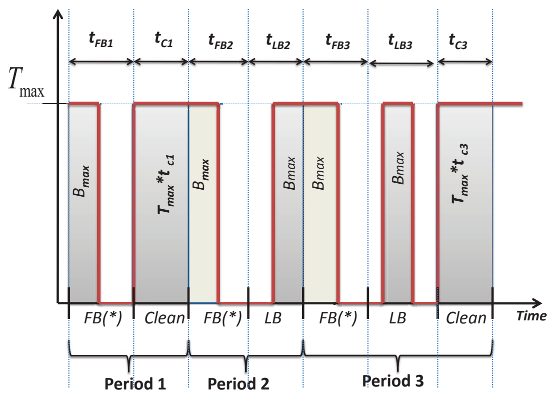
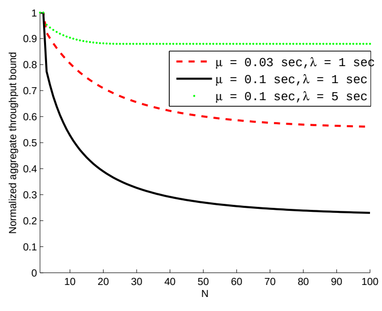
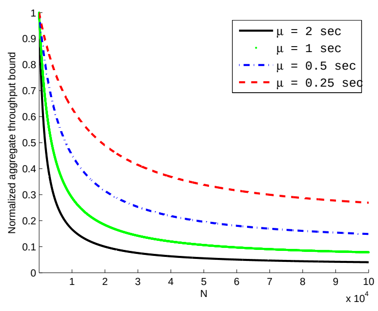


Fig. 2: *All possible sequences of system states with one of possible throughput scenarios. As seen in the figure, throughput oscillates from the peak to zero and back. In our full-length paper, we show that oscillations are easily provoked in TCP overlay scenarios, but the analysis is beyond the scope of this brief announcement.*

In our model, oscillations arise as follows. Suppose we are in a *Clean* state and some internal node i becomes *Bad* (the system transitions to state *FB(*i*)*). As a result, i stops forwarding packets. There are still incoming packets from the upstream link, which fill i's buffers. When i's incoming buffer fills up, the TCP flow control mechanism causes i's parent node to stop sending, which in turn causes its buffers to fill up. If i's disturbance persists, then eventually all the buffers on the path from the root to i become full, and the root's sending throughput drops to zero. Only after the system returns to a *Clean* state, the root may again send packets.



(a) Fine-grained disturbances.



(b) Coarse-grained disturbances.

Fig. 3: *Normalized aggregate throughput as a function of the system size in presence of disturbances using default Linux buffer size of* 64*KB. Throughput collapse occurs even for small tree sizes (*≈ 60 *nodes), if an average node experiences frequent (e.g. every* $\lambda = 1$ *sec) and short disturbances (left) or single brief* $\mu = 250ms$ *delay per hour (*$\lambda = 3600$ *sec) (right).*

During each period k, let FB_k, LB_k and C_k denote the number of bits transmitted by the root during the period in the *FB(*)*, *LB*, and *Clean* states, respectively, and t_{FB_k}, t_{C_k} and t_{LB_k} denote their duration in seconds. The aggregate throughput of the root during m successive periods is $\textsc{Aggr}(m) \triangleq \frac{\sum_{k=1}^{m} FB_k + LB_k + C_k}{\sum_{k=1}^{m} t_{FB_k} + t_{LB_k} + t_{C_k}}$.

IV. Results

Our main result is a bound on aggregate throughput. We also investigate the latency of dissemination. Let $\mu \geq \frac{B_{max}}{T_{max}}$ and λ denote the expected delay duration and time between delays, respectively.

Theorem 1:

$$\lim_{m\to\infty} \frac{\textsc{Aggr}(m)}{T_{max}} \leq \min_{1\leq n\leq N} \frac{2 \cdot \frac{B_{max}(n)}{T_{max}} + \frac{\lambda}{n}}{\mu + \frac{B_{max}(n)}{T_{max}} + \frac{\lambda}{n}} \leq \frac{2 \cdot \frac{B_{max}(N)}{T_{max}} + \frac{\lambda}{N}}{\mu + \frac{B_{max}(N)}{T_{max}} + \frac{\lambda}{N}}. \tag{1}$$

Figure III shows the middle throughput bound in (1). Fine-grained disturbances model minor message loss that arises in switches and routers when default TCP stack implementation is used in high-throughput low-latency networks (the *Incast problem* [7]). In this scenario, throughput degradation can be up to 70%. Coarse-grained disturbances model rare (once per hour) forwarding delays in a loss-free environment and may cause significant throughput degradation (up to 90%) in large systems. Even when buffer sizes are increased to 2MB per connection (32-fold increase), the degradation is still 15–30% in both scenarios.

Theorem 2: The expected delivery latency of any reliable multicast scheme with bounded memory will grow like $\Theta(\log N)$ with the size of the system when node processing delays are exponentially distributed, and as $\Theta\left(\sqrt[\alpha]{N}\right)$ when node processing delays follow a power-law distribution with parameter α.

V. Conclusion

Our work responds to a widely observed (but poorly explained) problem: data centers replicate information using overlay trees constructed from reliable links, but sometimes perform poorly even when everything seems to be operating smoothly. Our model offers possible explanations: a reliable multicast constructed in this manner often experiences throughput collapse and the problem grows worse as a system scales up.

References

[1] F. Baccelli, A. Chaintreau, Z. Liu, and A. Riabov. The one-to-many TCP overlay: a scalable and reliable multicast architecture. In *INFOCOM*, pages 1629–1640. IEEE, 2005.
[2] K. Birman, G. Chockler, and R. van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.
[3] A. Chaintreau, F. Baccelli, and C. Diot. Impact of TCP-like congestion control on the throughput of multicast groups. *IEEE/ACM Trans. Netw.*, 10(4):500–512, 2002.
[4] A. Greenberg. Networking the cloud, 2009. Keynote address at ICDCS 2009, Montreal, Canada.
[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, 2009.
[6] G.-I. Kwon and J. W. Byers. Roma: Reliable overlay multicast with loosely coupled tcp connections. In *INFOCOM*, 2004.
[7] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan. Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, Feb. 2008.
[8] S. Ratnasamy, A. Ermolinskiy, and S. Shenker. Revisiting IP multicast. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, New York, NY, USA, 2006. ACM.
[9] M. Theimer. Some lessons learned from running Amazon Web Services, 2009. Keynote address at LADIS 2009, Big Sky, Montana.
[10] G. Urvoy-Keller and E. W. Biersack. A Multicast Congestion Control Model for Overlay Networks and its Performance. In *NGC*, October 2002.
[11] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, G. Chockler, H. Li, and Y. Tock. Dr. Multicast: Rx for data center communication scalability. In *Proc. of EuroSys 2010: European Conference on Computer Systems*, Paris, France, April 2010.