High Performance, Secure, Robust and Transparent Messaging Service

Yair Amir, Claudiu Danilov, Cristina Nita-Rotaru
Johns Hopkins University
{yairamir, claudiu, crisn}@cs.jhu.edu

Introduction

"The distributed systems research community knows a great deal about reliability and security. Yet, our tools often scale poorly, rarely are seen as practical, and even more rarely have any significant impact on general commercial practice. Nonetheless, the need for distributed systems offering strong guarantees has never been more acute."

The above paragraph is taken from the workshop's web page. It almost accurately reflects an observation that underlined our work over the last few years and shaped its intended goal: to build distributed systems tools that actually make a difference by being widely used. In the next couple of pages we draw with broad strokes a vision for a distributed infrastructure that will allow existing applications to better use current networks by exploiting distributed systems services in a way that is transparent to the application.

In our opinion, the main reason advanced distributed techniques are less successful is because most of them tried to force application builders to change their programming model and even their way of thinking. Therefore, we require our distributed infrastructure to be meshed in the current programming paradigm.

We propose a secure messaging infrastructure for unicast and multicast with near-optimal performance and stronger semantics, beyond what is naturally achieved over the Internet, and completely transparent to the application. This infrastructure is constructed by long-lived daemons, running as user-space programs. The daemons dynamically create and maintain a logical overlay network. The algorithms on this overlay network are not bounded by the standard Internet protocols. Rather, they exploit the more limited scalability of the overlay network (as opposed to the global Internet) to optimize performance (e.g. latency, throughput). Our infrastructure still can scale to much higher numbers than, for example, group communication systems, as we maintain much weaker global guarantees.

The target application domain is essentially any application that uses the Internet.

Relevant to our work are virtual private networks (VPN) [YS01] and the overlay network approaches in the USC/Xbone [TH98, Tou01] and the MIT/RON [ABKM01, And01]. VPN, Xbone and RON provide transparent service to the application. This property is critical for such a system to be practical. Our work is similar in this respect.

Virtual private networks usually focus on the security aspects, instantiating secure tunnels between different sites in order to extend the firewall domain over wide area networks. Our messaging infrastructure extends that by also focusing on the message performance and semantics such as low latency reliability, and near optimal routing and flow control. The USC/Xbone system instantiates a logical IP network on top of the current IP network (there can be multiple layers of overlays). This work is orthogonal to ours in that our infrastructure can be deployed in any of the Xbone logical layers. MIT/RON's goal is to provide resilience by utilizing redundant paths. Our goal includes resiliency, but also focuses on optimizing the performance of the reliable messaging latency and overall network throughput, while addressing security concerns.

Next, we discuss the main properties of our proposed infrastructure and how each of them is achieved: transparency, high performance, security, scalability and robustness.

Technical Approach

Our messaging infrastructure instantiates an overlay network that is dynamically constructed such that each overlay link connects two overlay nodes running our daemons over the underlying physical network (e.g. the Internet). We present a concise description of the main desired properties and some techniques to achieve them.

Transparency

We consider two approaches to provide seamless integration of our infrastructure with existing applications. The first approach intercepts the application's messaging in the socket level, directs messages to our messaging infrastructure, and finally delivers them to the application on the receiver(s) side. The second approach instantiates a logical Internet router that gets all of the packets originating from the machine, routes them regularly if they are not addressed to overlay network participants, and handles them if they are.

We will enable new applications that want to use more powerful semantics to specify their needs using setsockopt or out-of-band signaling.

High Performance

The messaging infrastructure currently optimizes both the latency of a reliable service and the global flow control. Work on near-optimal dynamic multi-path routing is in progress.

We use buffers at intermediate overlay daemons to ensure hop-by-hop reliability [ADS00]. This way, the end-to-end latency for lost packets is improved because of two reasons: we can detect the loss much faster on the hop compared with an end-to-end detection (which has to take into account the diameter of the network), and we can recover the loss much faster, as we only need to recover it from the last hop that received it, rather than from the source.

The above technique can achieve substantial latency improvement. For example, assume that a diameter of a US-wide network is 50 milliseconds and there are five hops connecting end nodes, each with 10 milliseconds latency. An end-to-end recovery of a lost message will take at least 50 milliseconds to detect, 50 more milliseconds to request the lost message from the source, and 50 milliseconds to recover it, for a total of 150 milliseconds. In contrast, in our overlay approach, it will take only 20 additional milliseconds to recover one loss on one hop, in addition to the 50 milliseconds of dissemination, for a total of 70 milliseconds.

We use a cost-benefit framework to implement global flow control over the overlay network [AADS02]. We make use of an exponential cost function for each overlay link such as the cost of the link increases exponentially as the capacity of the link is depleted. When the resource is not utilized at all its cost is zero and when it is fully utilized, its cost is prohibitively expensive. We assign benefit based on the parameter we want to optimize, which is the sending throughput or receiving throughput (these are not the same in the case of multicast). We have proved that this scheme is competitive with a logarithmic ratio compared with the optimal **offline** algorithm and have obtained good results both in simulations (using ns2) and in practice (over CAIRN and Emulab).

Security

We identify two types of security concerns. One is the authentication, confidentiality and integrity of the information passed between daemons (user data, routing information. etc). The other is the authentication and access control between client applications and the overlay network daemons.

We create on-demand secure tunnels over the dynamic overlay network to provide confidentiality. Since the channels are established between daemons, the cost of security is amortized for the various applications. The entity authentication between daemons can be addressed by using certificates. A certificate-based approach allows us to avoid the need for global knowledge regarding the potential participants of the overlay network. However, we still need a certificate authority in order to verify the certificates. What is more problematic currently is that we do not have a scalable solution to how to revoke certificates in a completely distributed environment.

Another important problem is protecting routing information. Current solutions use either expensive methods such as digital signatures, hash chains or HMAC [MvOV96] techniques that are more efficient while lacking source non-repudiation. We intend to use a mixed scheme that exploits the advantages of both as appropriate.

Scalability and Robustness

Strong semantics services (e.g. membership) come at the expense of overhead and scalability. Our approach is to base our messaging infrastructure on looser semantics, which is a better fit for the unicast model and for the weaker-semantics multicast. Our Spread Toolkit [AS98, AAH+00] provides an efficient solution for stronger, group communication semantics.

Instead of maintaining state and sending control traffic per application connection, we keep track of the links in the overlay network. This approach scales well with the number of application sessions, groups and connections, with the expense of an overhead proportional to the size of the overlay network or even the number of immediate neighbors (depending on the desired semantics). Scalability with the number of nodes in the regular Internet is an important issue. However, distributed infrastructure embedded in the Internet (IP multicast) is not scalable with the number of groups both in terms of its limited global IP address space and in terms of the state per group that has to be maintained in every intermediate router.

When an Internet route or an overlay daemon fails, our system reconfigures itself quickly, reestablishing routes if possible, allowing applications to use the networking services with little interruption since no global membership has to be agreed upon.

References

[AADS02] Y. Amir, B. Awerbuch, C. Danilov, and J. Stanton. Flow control for many-to-many multicast: A cost-benefit approach. In Proceedings of IEEE Open Architectures and Network Programming (OpenArch), June 2002. *To appear.*

[AAH+00] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. Nita-Rotaru, T. Schlossnagle, J. Schultz, J. R. Stanton, and G. Tsudik. Secure group communication in asynchronous networks with failures: integration and experiments. In Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS), pages 330-343, April 2000.

[ABKM01] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In Proceedings of the18th ACM SOSP, October 2001.

[ADS00] Y. Amir, C. Danilov, and J. Stanton. A low latency, loss tolerant architecture and protocol for wide area group communication. In Proceedings of the International Conference on Dependable Systems and Networks (DSN), pages 327-336, June 2000.

[And01] D. G. Andersen. Resilient overlay networks. Master's thesis, Massachusetts Institute of Technology, May 2001.

[AS98] Y. Amir and J. Stanton. The Spread wide area group communication system. Technical Report 98-4, Johns Hopkins University, Center of Networking and Distributed Systems, 1998.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

[TH98] J. Touch and S. Hotz. The X-bone. In the 3rd Global Internet Mini-Conference at Globecom, pages 59-68, November 1998.

[Tou01] J. Touch. Dynamic internet overlay deployment and management using the X-bone. Computer Networks, pages 117-135, July 2001.

[YS01] R. Yuan and W. T. Strayer. Virtual Private Networks: Technologies and Solutions. Addison-Wesley, 2001.

Issues for the meeting:

Lesson Learned: What Worked and What did Not Work for Us in the Past

The Backhand project (www.backhand.org) is a research project we built for cluster support. The first component of the project, mod_backhand is a load-balancing module for the Apache web server that converts a cluster of separate servers into one logical server by balancing the load. All the system manager needs to do is to drop the module in the correct directory of Apache and add one line to the web server's setup file. mod_backhand was reported on 100 web sites on April 2000 and grew 40% per month, being reported on 10,000 web sites by April 2001. It is conservative to say that by now the number of web sites powered by it is in the low 10,000s. In addition to it being on several operating system distributions, we had around 4000 downloads of the software from our web site.

Another component of the Backhand project is a distributed logging software for a web cluster. We do not have an accurate number of clusters using this component, but it probably is in the low 1000s. It is one of the popular applications using the Spread toolkit, for which we currently record about 10 downloads by different organizations per day.

The common aspect of all of the Backhand components is that they are completely transparent to the user and the system programmer, and require very little from the system administrator in order to setup. This was key in its considerable success in the Internet community outside of the academic realm. While a lot of research went into the load balancing algorithms, their near-optimal behavior was not a factor in the decision of people to use it.

Our Spread toolkit (www.spread.org) also has achieved considerable attention outside academia with over 150 people on its developer mailing list. In addition to having the toolkit on several operating system distributions, we have recorded around 3500 downloads from our web site, most of them in the last nine months. As opposed to the Backhand project, Spread is only useful as a programming platform to build other applications. In order to do that, developers need to learn a new programming paradigm (group communication). Spread's API is very simple, requiring only six calls to be able to write efficient applications. Nevertheless, most of the usage of Spread is in powering existing applications (Apache-SSL, distributed logging, etc.) that a few sophisticated programmers wrote. Only now, after several years, we see cases of people that are using Spread to develop their own original applications.

Our lesson is that if a non-transparent tool provides a truly unique functionality that is really needed, there is a chance people will adopt it, but it probably will take considerable time to pick up. If the tool is transparent and does not require a change of programming model, or better yet, does not require any programming, it has much better potential.

Real Life Impact as a Measure for Success: Points for Discussion

If the measure of success for the field is how it influences the current and future practice in building distributed systems, it seems that a few subjective observations can be raised for discussion:

- Almost no academic credit was achieved as a result of our systems going beyond a simple prototype. If we want academic research to achieve real-life goals, we need, as a community, to reward such achievements in academy. In some cases, the fact that a real system is built (and is even successful) may actually hurt the academic creator as evaluators question the suitability of such work to academy.
- It seems that our community gives more credit to simulations compared with real-life results. The reason is, of course, that simulations seem very clean (even though they hide assumptions) and real-life systems are not producing "theoretical" results because of hidden costs. There is more work involved in benchmarking a system over a network with 100 computers than in simulating it over a network with 10,000 simulated computers. However, most of the time, this fact is not acknowledged when acceptance/rejection decisions are made.
- Building a system in an academic environment can serve as a bridge between the academic community and the industrial one. Our successful systems (in that sense) were created because of a real need with deep involvement of people with first hand knowledge of the particular need, usually based on real-life commercial experience. In such cases, we, the academicians, learned what are the problems that are worth investigating, and the industrial community got a superior solution that they were not aware of.
- For systems that went beyond a prototype and became widely used, most of the effort was invested toward the usability and robustness of the system rather than the algorithmic or conceptual work.
- The more seamless the system, the more successful it can become. We have success with Spread that is a non-seamless system, but it took several years to educate users outside of the research community to use its programming model. Moreover, Spread got a lot of exposure from a few popular applications early on.

It is clear from the above that the required activities and measure for academic success and research success are almost orthogonal to the required activities and measure for impact on real-life systems. Without the researchers themselves converting their concepts and algorithms into useful systems, at least in the distributed systems field, the concepts and algorithms may stay on paper. Therefore, if the community wants real-life impact achieved from academic research it may need to re-think they way research is evaluated.