How To Make Your Testbed Grow: Driving Testbed Growth Through Open Participation

Justin Cappos
NYU Poly
jcappos@poly.edu

Abstract

The scale and cost of experimental facilities are intimitely intertwined. To date, existing testbeds have represented a significant per-node cost incurred which prevents large scale expansion without large expense.

This work describes experiences using an open development and resource model which has resulted in a small per-node cost as the scale increases. By leveraging significant contributions from community members in Austria, Switzerland, England, Germany, Canada, China, and the US, our Seattle testbed has seen significant growth. For example, despite receiving orders of magnitude less support than the GENI project, the Seattle testbed is larger according to many node count and use metrics. The purpose of this talk is to share valuable lessons learned that can substantially enhance other testbeds.

I. Introduction

Existing testbeds, such as PlanetLab, Emulab, and various GENI efforts, are critical for evaluating networked and distributed systems. These testbeds, however, are composed of dedicated resources. As a result, they are expensive to scale as the management and equipment costs dominate. Using dedicated hardware also makes it difficult for a testbed to be representative of the Internet without constantly purchasing and adding the latest hardware. As such, testbeds composed of dedicated hardware do not have representative connectivity, processing power, and use patterns of existing Internet hosts. While this is not a major concern for researchers that want to focus on clean-slate design, such as through GENI, it significantly complicates other research.

Over the past three years, we have constructed a testbed using a different approach. Namely, our Seattle testbed utilizes computational resources provided by end users on their existing devices. As such, our testbed's growth and success is enabled through end user participation. Our testbed is designed to incentivize participation while minimizing risk to end users. To this end, Seattle is designed to preserve user security and to minimally impact the performance of the user's other applications. Despite the security restrictions that are necessary to make our platform safe for end users, our experience indicates that it is an easy to use and powerful platform for education and research. Our testbed provides researchers with the ability to run experiments on a variety of different laptops, desktops, mobile phones, and tablets.

After three years of adoption by home users and researchers, our testbed has roughly 2K registered developers and about 9K nodes. From a size perspective, this makes Seattle larger than PlanetLab, Emulab, and all other GENI projects — combined. Perhaps most remarkably, we have scaled to this size with less than $200K per year, more than two orders of magnitude less funding than GENI.

The growth and direction of our testbed follows a common model in the open source community. There is a small team which serves to work toward the general improvement of the core platform. More than 50 different researchers and open source enthusiasts around the world have contributed code to extend Seattle. As a result, the core development effort serves as a multiplier and catalyst for progress by outside groups. For example, researchers in Europe have ported our testbed software to run on new types of phones and tablets, added support for new network interfaces, integrated Seattle into existing G-Lab testbeds, constructed new services on top of the platform, and submitted patches to the core software itself.

II. Seattle Overview

The Seattle testbed consists of several major components that are similar to those on other platforms. Namely, the nodes that participate in the testbed run node manager software which performs authentication and resource control. Experimenters use experimenter services (often a shell) to run code on nodes. Experimenter code runs inside of a virtual machine. In order to acquire nodes, a user will most likely contact a clearinghouse. Several services run to coordinate and enhance the behavior between components.

The Seattle testbed has been constructed due to the efforts of more than one hundred developers from all around the world. Some of the major contributors are listed in Figure 1. These developers invested significant time and energy in implementing facilities for Seattle.

A. Educational Use

Seattle was first publicly promoted at SIGCSE in 2009 as an educational testbed. Seattle helps instructors fill a void in the networking curriculum by allowing students to get practical experience with real world network situations and conditions. The response from the educational community has been overwhelmingly positive. Seattle has been used in nearly two dozen classes, mostly on networking. The feedback that we have received from instructors is that the students like it

Component	Primary Developer(s)
Repy V1	Justin Cappos (NYU Poly), Armon Dadgar (U Wash)
Repy V2	Conrad Meyer (U Wash), Moshe Kaplan (NYU Poly)
Lind	Chris Matthews, Andi Bergen (U Victoria)
tun-tap bindings	Dennis Schwerdel (Kaiserslautern)
Node Manager	Justin Cappos (NYU Poly), Cosmin Barsan (Microsoft)
Software Updater	Justin Cappos (NYU Poly), Geremy Condra (Google)
Overlord	Alan Loh (U Wash), Justin Samuel (Berkeley)
Seash	Justin Cappos (NYU Poly), Alan Loh (U Wash)
Try Repy!	Lukas Pühringer, Albert Rafetseder (U Vienna)
Custom Installer	Alex Hanson (U Wash)
GeoIP	Evan Meagher (U Wash)
Owl	Scott Baker (U Arizona)
Shims	Monzur Muhammad (NYU Poly), Danny Huang (UCSD)
Time Service	Justin Cappos (NYU Poly)
Zenodotus	Sebastian Morgan (U Wash)
SeattleGENI	Justin Samuel (Berkeley), Sean Ren (U Wash)
Centralized Advertise	Sebastian Morgan (U Wash), Justin Cappos (NYU Poly)
Digital Object Registry	Giridhar Manepalli, Larry Lannom (CNRI)
OpenDHT	Sean Rhea (Meraki), Arvind Krishnamurthy (U Wash)
Secure Lookup Service	Sebastian Morgan (U Wash)
CheckAPI	Jeff Rasley (U Wash), Eleni Gessiou (NYU Poly)
iOS Port	Lukas Pühringer (U Vienna)
Android Port	Ákos Lukovics, Albert Rafetseder (U Vienna)
Nokia N800 Port	Armon Dadgar (U Wash), Justin Cappos (NYU Poly)
Nokia N900 Port	Derek Cheng, Jukka Nurminen (Nokia Research)

Fig. 1. Major Seattle Contributions.

because it is easy to learn and is fast to get their code up and running. Our most popular assignment (https://seattle.cs.washington.edu/wiki/EducationalAssignments/TakeHome) covers non-transitive connectivity and NATs, requires no programming, and takes the students only about 90 minutes.

There have been four educational workshops / tutorials for Seattle, with more than two dozen presentations, posters, or talks at educational venues. We hosted the first three workshops ourselves, however the most recent workshop is being run by educators who use the platform. These workshops serve to promote the platform and demonstrate how simple it is to use in the classroom. Educators have also given positive feedback about Seattle on other forums. As of October 1st, 2011, Seattle is the top ranked ACM SIGCOMM educational resource according to their website. We are working with Kurose and Ross to integrate assignments on Seattle into their networking textbook, the most popular networking text.

B. Research Use

Seattle has been presented and promoted our testbed at a variety of developer and research conferences, including GENI and research conferences. As a result, there are currently more than a dozen papers that leverage Seattle that are either published or under submission.

C. End User Adoption

The current implementation of Seattle has a wide distribution of users. We categorize our node types by performing a reverse DNS lookup and then examining the characteristics of the name. From this analysis, Seattle consists of 1720 nodes at universities, 791 nodes that are part of a testbed like PlanetLab or Emulab, 2916 nodes that are clearly home nodes (67 of which are phones), and 3370 nodes that do not respond to reverse DNS lookups (also likely home nodes. (However, since our nodes often have diurnal patters, our number of online nodes at a time is less than half of the total.) This gives a total of 8797 nodes, which also have significant geographic distribution. The most popular countries for our nodes are the US, followed closely by China, and then Germany, Canada, and Austria. However, while these countries have a significant number of nodes, the distribution is across 6 continents, with hundreds of nodes in Africa, South America, and Australia.

III. Incentivizing (and Understanding) Users

There are four major lessons that helped to grow Seattle.

First, provide a system that matches incentives and concerns. We were not getting significant end user adoption initially for Seattle. One concern was that consuming 30% of the resources seemed too high despite our data showing this had negligible performance impact. Choosing a more psychologically acceptable value (10%) increased our user base. Similar balancing acts have helped educational and research use.

Second, truly know (and cater to) thy userbase. One of the main criticisms of many existing testbeds is that they are designed without the eventual user in mind. They expose needless complexity and pain. We worked hard to streamline our user experience and documentation to turn an initial spark of interest into an excited and invested user / developer. This is to the point that one of our users, Kurt Tutschku, presented an impropmtu demo for our platform to our funding agency.

Third, promote your platform whenever it is appropriate. Seattle has been presented at more than 60 venues. This has resulted in a prominent Chinese blogger mentioning the software (leading to many Chinese downloads), a large percentage of our educational adoption through evangelism at educational venues, and many of the contributions from outside users. The counter point of this is to remember to do this only when it is appropriate. Be clear about limitations so that potential users don't become frustrated (and spread negative publicity) when trying to (mis-)use your platform.

Fourth, realize that you don't know everything (and what you do know may be wrong). Seattle was architected to allow the components to be repurposed and reused. Major components have been used independently, altered, and added based upon the actions of other developers. If we instead had forced a software stack or API, this would have made collaboration difficult and alienated potential contributors. By providing simplistic and malleable interfaces, other groups can set up instances and run our code in a variety of different scenarios and use cases beyond what we envisioned.

IV. Conclusion

The over-riding lesson learned from building Seattle is that by embracing and incentivizing collaboration, substantial gains can be achieved. This requires balancing the needs of multiple parties including end users, researchers, and educators. By applying a psychologically acceptable mix of *incentives* and *protections*, one can simultaneously foster growth in multiple communities simultaneously. Our talk will further describe how these lessons apply to other testbeds.