James Keeler

Information about the FitNet Project.

Project Report

FitNet System

# Introduction

The main goals for this project were to expose the students to the GENI environment, OpenFlow, and Software Defined Networking, allow them to gain experience with the Agile development methodology, and to learn about the inner workings of network data transmission.

## FitSend

This program is intended to read a stored high definition video file with a resolution of 1920x1080 and a playback frame rate of 30 fps. Still images of this video are captured to the sender’s hard drive, then packetized, and sent to a list of receivers. Once sent, each image is deleted from the sender’s hard drive.

## FitReceive

This program is intended to receive the packetized images from FitSend and report on packet loss and throughput. Video playback was not implemented in this version of the software due to the nature of the project.

# Team FitNet

Team FitNet is comprised of three UTC students, James Keeler, Daniel Joyner, and Noah Falkie, and, the customer, Bob Summers. Using the Agile methodology means that the customer is an active, integral part of the team, and his contribution to the final product was critical to its success.

The team roles for the UTC students are designated as such:

* Team Leader – James Keeler
* Lead Developer – Noah Falkie
* Documentation Specialist – Daniel Joyner

While these roles all have well-defined responsibilities, none of the team members are fully responsible for all tasks associated with their role.

# Project Requirements

The prioritized list of requirements for this project was defined by the customer thusly:

* Use GENI and OpenVSwitch for the operating environment
* All code must be written in Python for portability
* A high definition video must be sent from one node to at least one other node with no loss of data
* GitHub must be used for code versioning and a central repository

# Product Design

The design of the system is quite simple. Using GENI, a small network is created with at least two nodes, one for sending data and one for receiving data. The receiver program, FitReceive, is launched on all receiving nodes and waits to begin receiving data. The sending program, FitSend, is launched on the sending computer and passed a list of IP addresses or host names of the receiving nodes.

The sending program reads the video file, captures still images of the frames of video, opens a UDP socket to each of the receiving nodes, packetizes the images, sends the packets, deletes the completed images, and closes the socket.

# Methodology

Team FitNet was created by Bob Summers with a central goal of exposing students to the real-world practice of Agile development. Using this development methodology allowed the UTC students direct access to the customer and allowed the customer to play a central role in the success of the project.

Constant communication is key, and to help facilitate this, Team FitNet conducted meetings every Tuesday and Thursday, with video conference calls made on Thursday to discuss issues with the customer. There were some ad hoc meetings on Sundays as well.

The Agile methodology doesn’t have the stringent requirement for documentation and diagrams like the iterative method or waterfall method. This lack of documentation contributed to miscommunication among team members, and delayed certain parts of the project. On the other hand, development resources were able to be applied directly to solving problems in the code rather than on tedious documentation.

# Lessons Learned

During the course of this project, Team FitNet had to overcome a number of obstacles which were used as teachable moments. Here are some of the lessons learned during this process:

* Never be afraid to speak candidly with your customer
* Be careful not to agree to an unachievable set of requirements
* Never under estimate the value of experience
* Don’t rely on other people to teach you about “bleeding edge” technology
* Don’t try to “boil the ocean” on your first attempt. Do something simple and allow the open source crowd to help drive it to completion over time.
* It’s best to develop in an environment that you control—one that is stable.
* Communicating changes to all stakeholders is critical, regardless of the project size.
* Don’t expect to develop stable code on an unstable system.