1 Description

The class project is an 8-week long endeavor during which you will explore a specific area, examine related works in the space, and either implement an idea from related work or come up with a new solution to the problem.

The project consists of four milestones, each of which is discussed in more detail below.

- (05%) Group/Partner Selection. Email the instructor and the TA your group members. You get full credit for submitting the group information on time. Your email should also include a list of potential projects of interest.

- (15%) Proposal (2 page write-up, references are included in the 2 page write).

- (20%) Final presentation (10 minutes).

- (60%) Paper (6 page write-up + 1 for the references).

1.1 Proposal

This deliverable requires you to read related works and to thoroughly understand the problem you will solve. Given this, you will describe what problem you aim to solve or what system you aim to implement. Then you will provide an timeline detailing how you plan to accomplish this over the next 8 weeks. Your proposal should have the following components:

- Introduction: Description of the problem being tackled and why it is an important problem to tackle.

- Related Works: Set of relevant papers and how what you intend to do is different from what others have done (e.g. develop a new idea or making an OpenSource version of an existing technique — FYI Hadoop is an OpenSourced version of the Google’s Map ).

- Proposed Work: What you intend to do: languages you will use, frameworks you will use, APIs/Libraries that you plan to use. The set of deliverables that will be the outcome of the project.

- Timeline: What you plan to achieve each week.

1.2 Final Presentation

This is a 15 minute presentation with slides. You will discuss the motivation (2 minutes), the architecture of the system developed (3 minutes), challenges faced in developing this architecture (5 minutes), and some initial results and open problems (5 minutes).
1.3 Final Paper

This deliverable builds on the proposal submitted earlier. However, it summarizes what you’ve done, the problems faced, and solutions to these problems. Unlike the proposal, the final paper does not include a timeline or a proposed work section. The final paper has the following sections:

- **Introduction**: Description of the problem being tackled and why it is an important problem to tackle. (This may be the same as the proposal)
- **Related Works**: Set of relevant papers and how what you intend to do is different from what others have done. (This should be more comprehensive than the version in the proposal)
- **Architecture+Prototype**: You will each build a system for processing data, for generating data, or for solving a problem. What is the architecture? what are the different components of the system? What problem does each component solve? How do the different components interact?
- **Data-Set**: What data does your system operate on? Over what time span was this data collected? where was the data collected from?
- **Evaluation or Initial Measurements**: If you are purely designing a system or a prototype how does it scale? If you are performing a measurement study, what are some observations?
- **Open Issues**: If you had another 8 weeks, what would you like to get done? What features would you like to add or improve? What new questions would you like to answer?

2 Potential Projects

2.1 Mapping of Services: Measurement Study

The Internet is comprised of many services, e.g. NetFlix, GitHub, and Reddit, that we interact with on a daily basis. These services are all supported by complex infrastructures with several of them sharing the same underlying hardware. For example, Reddit and NetFlix are both hosted atop the same amazon web-services (AWS). In this project, your goal is to reverse engineer the infrastructure of a service and understand the policies being used internally within the services.

For example, you would want to find out if the same servers are used for American visitors versus European visitors? Are the same servers used for HTML or Videos? Does uploading file (e.g., videos) use a different set of servers than downloading files or watching videos? Do common API calls use different deployment patterns than less common API calls? How are different services different? (how is GitHub’s deployment different from Facebook or Netflix or Reddit)

To reverse engineering these services, you will need to create programs that test out all the APIs provided by each services from a variety of locations across the globe. For example, test the different YouTube API or the GitHub API.

**Related Works**

- Next stop, the cloud: Understanding modern web service deployment in EC2 and Azure
- PlanetLab: A testbed for setting up global experiments
- WhoWas: A platform for measuring web deployments on IaaS clouds

The Deliverables for this project include:
The source code, the data, scripts for running the experiments, and a description of how to run your code on the data.
2.2 Analysis of BGP changes: Measurement Study

In this paper, you will analyze BGP updates and trace-routes over the last 10 years. The goal is to understand the nature of BGP changes over the last decade. How has the number of updates per second changed over time? How are the type of updates changed? Have the prefixes being advertised changed? Has the withdrawal patterns changed over time?

More over, there have been a number of BGP issues caused by security, privacy, and code-bugs. Your goal is to understand the implication of each of these issues on BGP updates and on the end-to-end performance (via trace-routes). Similar to HW#2, you will try to correlate problems to specific updates and to determine how these updates impacted trace routes to the relevant prefixes.

Data-Source

- BGP Updates: http://archive.routeviews.org/
- Trace-routes: https://www.planet.com/

Related Works

- Characterizing Large-scale Routing Anomalies: A Case Study of the China Telecom Incident
- Analysis of RouteViews BGP data: policy atoms
- BGP molecules: Understanding and predicting prefix failures
- Analysis of country-wide internet outages caused by censorship

The deliverables for this project include:

The source code, the data, and a description of how to run your code on the data

2.3 Hedera: Implementation

For this project, you will be implementing an SDN application that identifies large flow and performs load balancing for these large flows. This SDN Application is based on the paper below. You will implement this on the ONUS controller (which is a different controller platform than that used in HW#3). This will require understanding and looking at the tutorials for the ONUS controller. You will also need to determine how to implement Hedera in a distributed manner.

The paper below includes very specific experiments, you will try to create these experiments with your prototype.

The deliverables for this project include:

The source code, a working VM containing the setup in which you tested your code.

Related Works/Links

- Hedera

2.4 Contextual Networks for Mobile Networks: Implementation.

In our home networks and in the mobile networks, we are often in resource constraints environments. Consider, the following two scenarios:

Scenario 1, you are in a location with poor cellular coverage (3 bars) your Internet connectivity is poor. While checking email, you may not want the images just the text. When reading websites you may want poor resolution images. Essentially, you want things to load quickly.
Scenario 2, you are roaming. You want a similar lower quality experience because you are paying much more and all you care about is performance.

Unfortunately, most mobile platforms today only run in one “mode”. That is to provide the best possible user experience. You can turn down video quality but it is not easy or straightforward to turn off all pictures or to request lower quality pictures.

In this project, you will modify the Android OS to detect changing in network experience or to detect roaming conditions. Given this, then you will change the browser and mail client to operate in a degraded mode.

The deliverables for this project include:

- Modified Browser/mail-client
- configuration files for specifying conditions for going into degradation mode

To understand the performance and evaluate your deliverables: use the browser/mail-client under Wifi-networks that simulate poor performance and measure the times to load different webpages and emails with and without your framework.

Related Works/Links

- procrastinator
- infonames

2.5 ChaosMonkey For BGP Networks: Implementation.

Failures are prevalent in networks. Unfortunately, many of these failures only occur in production environment. Motivated by this, Netflix, Microsoft, and Google have started to create frameworks to inject failures into the production environment – The main idea is to inject fake failures at a controlled time in a controlled manner. This way, you have engineers and developers available to fix any problems and if things become catastrophic you can fix it by undoing the failure.

In this project, you will learn to use Quagga to setup a small network with 4-5 routers, each router is a VM. Then you will learn to configure Quagga on each of these VMs – they will be configured in 2 different topologies: a mesh and a hub topology (I can explain these topologies to you). You will then add a REST-API to the Quagga code base that allows you to trigger node and link failures, below we describe each of these failures. Quagga is in C++, alternatively there are BGP implementations in Python.

- Node Failure: Take down all TCP connections and keep them down.
- Link Failure: Take down a specific link

In addition to the REST-API calls to create failures, you will also create calls to undo the failures. You will need to write code that captures BGP updates right after the failure and measure the amount of time it takes for updates to stop.

In testing and evaluating your system, you will load different numbers of BGP prefixes in each of the different routers and analyze how the system performs under different load.

Related Works/Links

- Netflix Chaos Monkey
- Network Chaos Monkey
- Quagga OpenSource Router