ERICSSON

Media Delivery Network Simulator

Massive, parallel, real-time distribution of data

Abstract

The largest minority of traffic traversing the Internet today is video, in various shapes and sizes; combined, all media forms have a peak time majority of network traffic. The amount of media traffic is only set to increase, as more and more people choose IP as their delivery method of choice. The natural question, therefore, is whether we are transferring that traffic in a good manner, or if we can do better.

In order to answer the question of how to transfer the traffic well, we must first create the traffic. As experimenting on customer networks is very costly, and current models for traffic simulation assume traditional loads, creating a simulator will create a distinct opportunity to experiment with different algorithms and attempt to find how to better transfer media.

1 Background

Netflix. YouTube. Hulu. ESPN. Pandora. Spotify. The rise of media delivery over the Internet is, at this point, needs little introduction. What needs highlighting, however, is the increasing amount of friction between the needs of delivering the same content to many people – the exact reason why broadcast media was invented – and the fundamental structure of the Internet. This is best exemplified by the difference between Netflix' catalog size, which is between three and four petabytes in size, and its monthly streaming. Netflix streams over 114,000 *years* of video every month. That translates into almost 1,000 petabytes of bandwidth per month, or 30 petabytes per day, assuming every stream is SD only. According to Netflix, HD streams use three times that bandwidth. In short, it is a question of scale.

The wastefulness above is compounded by the fact that not all content is equal. Some content is more popular than other, and thus much of that repeated transfer is of the same content, streamed to the same devices. Things look even worse when looking at ESPN streams, which show the same content, at the same time. In short, streaming uses linear resources, at best. The question is, can we do better?

In order to produce a good conclusion to that question, we need good data.

2 Overview

The goal of this project is to build a "life sized" simulation of internet-based media distribution, with a flexible framework that will allow tinkering, experimentation and evolution. The simulator consists of multiple interconnected components to allow this.

Five main elements make up the basis of the simulator: a management layer, with a web interface, a "**source**" node capable of generating media data, a "**processing**" node that would perform some process on the data, a "**relay**" node that would consume the media data, and a "**sink**" node that would consume the data.

The management layer is responsible for controlling the entire simulation. It must be able to talk to all nodes and control them. All nodes must also report key performance metrics to the management layer, which is responsible for logging them and presenting them to the user in an understandable manner. Finally, the management interface should allow for setting up of nodes, connecting them, and initiating traffic simulation. Some scripting or automation will help the simulation be flexible.

When reading the following descriptions, all simulations refer to actual resource usage of the hosts. A "life sized" simulation entails creating the actual load.

Source nodes generate data. The data itself may be random or actual information, but must be interspersed with some form of metadata that allows tracking of jitter, latency, packet loss, and other factors. The data generation strategy must be configurable, to simulate different loads. Finally, the data generation should be triggerable, i.e. to occur only when a certain condition occurs.

Processing nodes perform an arbitrary operation on data flowing through them. This simulates various processes that may occur, such as quality checking, metadata, subtitle addition, ad insertion, re-encoding, or any other process. For each process so occurring, the node must consume a configurable amount of CPU and RAM, the exact method to be determined. A configurable delay and adjustment to the stream may be applied (for example, reducing the bandwidth to simulate compression.

Relay nodes take one input stream and send it out to multiple recipients. These nodes need no additional simulation parameters, as all the restrictions will occur from the actual hardware or cloud environment used to run them.

Client nodes request an input stream by generating an event and sending it out on an event bus. This event is captured by the management or its agent, which becomes responsible to modify the simulation, according to a given ruleset, to set up the appropriate stream.

The modular structure of the simulator allows it to be rearranged to simulate any number of larger scenarios. The simulator is responsible for generating and consuming data, while monitoring key metrics. The underlying hardware or cloud environment used will show the usefulness of each type of system configuration.

3 Scope

This project covers the following elements

1. Design and document of data model and communication protocol to launch, capture and terminate the simulation, its agents and nodes.

2. Implement the four basic node types mentioned above.

3. Implement the management framework and user-facing application, which forms the backbone of the simulation.

4. Perform at least two comparative simulations, verifying the flexibility of the simulator and system.

4 Goals

In addition to the functional goals below, the project should deliver the source and documentation in a manner that facilitates reproduction.

- **Primary**: Show a functioning, user-friendly simulator that will format the simulation output in a way that is both easy to grasp, but where details can be viewed when needed.
- **Secondary**: Show the flexibility of the simulation by allowing customization of the simulation parameters before runtime.
- **Tertiary**: Show extensibility of the simulation by how additional nodes, types or extensions to existing nodes can be made to simulate use cases not covered.

5 Technologies & Skills

This is a non-exhaustive list of technologies and skills that this project may encompass, as well as why. Merely because a technology is included here does not warrant its inclusion in the project, and the lack of inclusion in this list does not mean a different technology is irrelevant.

- **Java**. The entire simulator is to be written in Java. While no version is specified, using later versions may offer a learning experience.
- **Web Applications**. The management tool shall be a web application, thus HTML/JS/CSS
- **Distributed Software Design**. The simulator is effectively a large, distributed system. A message bus shall be used for communication between the nodes.
- **Networking**. HTTP/WebSockets/TCP/UDP knowledge. The system will be a distributed system comprising of a number of nodes. While a library will most likely handle the message bus issues, the stream data itself shall be sent via UDP.
- **Data Management/Analytics**. In order to make a good presentation, the data from the system should be gathered, analyzed and shown in real time.

6 Team Size

This project would be ideal for a group of three to six people.

7 Roles & Responsibilities

The team would be responsible for establishing a project structure within the team. Recommended roles include project responsible, task manager, maintainer (code review, merges, qc). A designated person should ensure there are short, daily team meetings, and status updates to be shared with Ericsson.

Ericsson will designate at least one contact person for the daily updates. They will participate in at least one weekly meeting, preferably at the site of development, to assist with progress, issues or other project needs.

Finally, Ericsson may supply hardware or software to assist in the project. The team should not be constrained by the hardware or software Ericsson can or is willing to provide (although hopefully it will be useful, and thus used).

8 Impact

TBD

9 Preliminary Roadmap

The following is intended as an illustrative timeline for the project, and may need to be adjusted. However, key milestones are critical.

- **Week 1**: Walkthrough, discussion and brainstorming about the project. Write use-cases, sketches and initial ideas. Familiarize with technologies. Write experiments. Identify components (Ericsson, OSS or others) that can be leveraged to build the simulator faster.
- **Week 2**: Begin work on individual system components.
- **Week 3**: Begin parallel work on message bus/convergence layer.
- **Week 4 (key)**: Individual component demonstrations. Components may use static configuration in lieu of the management layer.
- **Week 6**: First integrated system demonstration. Should show a proto-management interface and how that interface can be used to set up the system.
- **Week 7**: Reflect on initial system prototype. Consider any changes/improvements.
- **Week 8 (key)**: Demonstration of revised system. Should include a more advanced management system, and show some flexibility in terms of the simulation.

- **Week 9**: Revision of the simulation. Ensure flexibility, extensibility and documentation.
- **Week 11**: Feature complete by end of week. Bugs fixed, stability and accuracy of data checked.
- **Week 12 (key)**: Final demonstration. Packaged delivery of source & Documentation.

10 Success Criteria

The primary success factor is a successful end-to-end simulation, with the data from the simulation gathered, analyzed and presented by the management interface.

A secondary aspect is the system extensibility. Ideally, the simulation would be flexible and extensible enough to be used with more node types. A successful decoupling of management and nodes to allow such changes is a great success.

The final aspect is the quality of the deliverables themselves: the ability to compile and run the software on different machines, the usefulness of the documentation and the stability of the system.

11 About Ericsson

When you make a call or browse the Internet on your mobile phone, tablet or laptop, you will most likely use one of our solutions. 40% of the world's mobile traffic uses networks that are built by Ericsson. The 10 largest operators in the world use our solutions. Ericsson was founded in 1876 and is a world-leading provider of information and communications technology and services. Our research and development made mobile communications and broadband possible.

Today, we are roughly 100,000 employees and we are the fifth-largest software company in the world. We have customers in more than 180 countries. We build and run mobile networks everywhere: deep in the Amazon as well as in megacities. This is important as increased broadband coverage has a positive impact on a country's GDP and on people's lives.

There are more than 6 billion mobile subscriptions across the world today. In the future, everything that benefits from a connection will be connected. People, business and society will start using the networks in new and unexpected ways, which will fundamentally change the world we live in. This Networked Society will result in new possibilities for everyone and we, together with our customers, are making that happen.