Media Delivery Network Simulator

Statement of work

Prepared by
Geng (Jeremy) Fu [geng.fu@sv.cmu.edu]
Jigar Patel [jigar.patel@sv.cmu.edu]
Vinay Kumar [vinay.kumar.vavili@sv.cmu.edu]
Hao Wang [hao.wang@sv.cmu.edu]

Clients
Vladimir Katardjiev [vladimir.katardjiev@ericsson.com]
Alvin Jude Hari Haran [alvin.jude.hari.haran@ericsson.com]
Nimish Radia [nimish.radia@ericsson.com]
Organization Affiliation: Ericsson

Faculty Advisor
Jia Zhang [jia.zhang@sv.cmu.edu]
Organization Affiliation: Electrical & Computer Engineering, Carnegie Mellon University

10/7/2014

Contents

Project 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.

Project 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.

To make it independent of Internet infrastructure, the simulator is to build an overlay network over the Internet, which is a distributed systems which consist of five main elements: a management layer, with a web interface, a "source" node capable of generating media data, a "processing" node that would perform some media processing on the data, a "relay" node that would consume the media data and multicast data, and a "sink" node that would consume the data.

The management layer is responsible for controlling the entire systems. 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, Ericsson will provide the model to predict the resource consumption for different processings. 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.

Client Needs

The following are the project goals that has been defined by our client,

- Design and document of data model and communication protocol to launch, capture and terminate the simulation, its agents and nodes.
- Implement the four basic node types mentioned below
    - Source
    - Processing
    - Relay
    - Client
- Implement the management framework and user-facing application, which forms the backbone of the simulation
- Perform at least two comparative simulations, verifying the flexibility of the simulator and system

The following are the functional goals of the software as defined by our client

- 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

Services to be provided

The services to be provided to the client will focus on satisfying the project goals and the functional requirements.

- Software Development

  The team will use the following tools and technologies for developing a fully functional simulator:

  - Java: Develop the backend for simulation, including the management layer, data transferring implementation.
  - HTML/CSS/JavaScript: Develop the web application for user interface
  - Git: Control the version of the project
  - Apache Maven: Manage build process and dependencies

- Weekly Update

  During the course of the project development, the team will host weekly update meetings with the client. The team will collect the feedback from the client on previous weeks iteration and decide on the next weeks iteration.

- Simulator Usage Tutorial

  The team will provide the tutorial of the usage of the simulator. The team will walk through the project in each release and provide demonstration on how to run the simulation.

- Documentation

  The source code will be documented so that development work on extending the simulator can continue after the period of the practicum. The simulation design, framework and usage will also be documented in a manner to facilitate reproduction.

Project Plan

Approach

We decided to follow agile approach overall. The project is divided into multiple iterations and each iteration has multiple milestones. The plan here is subject to continuous change as per feedback from client.

Iterations

Iteration-1 9/01/2014 – 9/21/2014

Practicum project kickoff:

- Talk to clients to understand the client needs
- Explore existing solutions to analyze the gap between the project requirements and the existing solutions
- Propose the initial design

Iteration-2 9/28/2014 – 10/6/2014

Prototype end-to-end solution:

- Prototype the media delivery network simulator as fundamental end-to-end solution
- Provide fundamental functionalities such as web interface, management layer and basic simulation scenario.
- Finalize (and learn) on technologies and tools based on evaluation feedback.

Iteration-3 10/06/2014 – 11/02/2014

Build full version of the system. The system can achieve the following functionalities:

- Provide a flexible and manageable configuration interface to start simulations
- Have a working system with scalability to at least 100s and possibly 1000s of nodes in the simulation cluster
- Render a user-friendly view to demonstrate the traffic statistics in the systems

Iteration 4 11/02/2014 – 12/03/2014

Test the system and refine the system:

- Test the systems and fix the bugs
- Deploy the system on the cloud environment
- Finish the documentation and prepare for the final deliverable

Milestones and Efforts

The following table breaks down the project into milestones with its estimated schedule and effort. The table will be updated as the project proceeds. The effort is calculated based on person hour. The number indicates the time put by one team member. We have four members in the team. The total effort in time will be timed by all four us. The id is the sequence number for each of the milestones from the beginning to the end.

ID	Milestone	Estimated Start Date	Estimated End Data	Effort (in person hour)
1.1	Collect, Analyze and Understand requirements	9/1/2014	9/14/2014	80
1.2	Explore Existing Network Simulator & other solutions	9/7/2014	9/21/2014	80
1.3	Discuss the initial overall design with clients and prototype to show	9/7/2014	9/21/2014	16
2.1	Implement Initial Prototype	9/15/2014	9/28/2014	100
2.2	Implement changes based on feedback on prototype	9/15/2014	9/28/2014	100
2.3	Finalize on technologies and tools to use - setup project on all dev machines	9/21/2014	10/6/2014	10
3.1	Implement multiple sink nodes and demonstrate multiple data streams passing through the	10/6/2014	10/13/2014	100

	network simultaneously			
3.2	Adding all 4 types of nodes	10/13/2014	10/20/2014	100
3.3	Finalize and implement input part (Input formats - except for deployment of nodes related portion)	10/20/2014	10/27/2014	40
3.4	Implementing meta data packets in the data stream for facilitating jitter, latency metrics	10/20/2014	10/27/2014	50
3.5	Completion of output part (how to display all the metrics for all scenarios).	10/27/2014	11/02/2014	70
3.6	Implement memory and CPU usage in simulation. Collection and reporting of CPU and Memory usage	10/27/2014	11/02/2014	30
4.1	Functional Testing and bug fixing	11/02/2014	11/11/2014	100
4.2	Building deployment layer - Deploy the system in at least one cloud environment	11/11/2014	11/25/2014	200
4.3	Integration testing and documentation	11/25/2014	12/02/2014	100
5.1	Demo and Deliver to the client	12/03/2014	12/05/2014	10

Team Roles

We hope to rotate the roles in each iteration so that each teammate will benefit from getting the whole view of the project. For the current iteration, the team role is as following.

Project Manager: Jigar Patel

Project Implementation:

- Management Layer: Jeremy Fu and Vinay Kumar Vavili
- Data Transfer Implementation: Jigar Patel, Vinay Kumar Vavili and Hao Wang
- Web Interface: Jigar Patel

Special physical resources

- Message bus: This project uses Ericsson Warp to coordinate nodes.
- Cloud deployment environment: AWS, Google Cloud and Microsoft Cloud are potential deployment platforms of this project.

Project Deliverables

In conjunction with the CMU deliverables, initial prototype during the development iterations in parallel, the team will be producing the following deliverables for media delivery network simulator:

• Statement of Work: Documents the negotiated project scope, deliverables and milestones.

• Technical Report: Documents the design and implementation of the systems made in the projects, which also includes the things we tried and discarded along with the reason. And the report will provide the UML artifacts.

• Project Summary: Describe the purpose and outcomes of the project for inclusion on the CMU-SV public website.

• User Manual: Help user quickly get familiar with media delivery network simulator

• Source Code and Library Documentation: The functioning software that is developed by the team and documentation for the library.

Criteria for measuring outcome

This project involves well defined requirements and goals. This makes it a good candidate for the Agile methodology of software development. We were given a timeline for deliverables in the project proposal document by Ericsson. We have defined project milestones [Section 4: Project Plan and Cost] based on this timeline. The focus of the milestones will be “End to End development” and “demo-able” software development. This ensures that the software will be “release-able” at the end of the practicum timeline.

We have arranged for weekly scrum meetings with all stakeholders involved including the developers, project advisor and our clients). The agenda of the scrum meeting will be as follows,

- What has been done this week.
  - Project Demo
  - Design of modules developed
  - Design justification and alternatives prototyped
  - Technical challenges faced
- What will be done next week.
  - Review of project milestones defined
  - Additional requirements gathering
- Feedback

The meeting minutes are compiled and shared with all stakeholders for record keeping and tracking.

To measure the outcome every week, we will use the feedback for the “demo-able” software received from our clients and faculty advisor. To measure the overall outcome, the following deliverables will be considered,

- Source code’s maintainability, modularity, documentation
- Final “release-able” software that can simulate 2 different MDN
- Extensibility of simulator
  - Node Types
  - Deployment platform
  - Configurability

Risk Identification and Handling

ID	Risk Description	Mitigation Strategy
1	Improper understanding of requirements	Spending more time initially on discussions so that everyone is comfortable with big picture of what needs to be done and involving client and all other stakeholders in initial meetings
2	Uncertainty on what tools and technologies to use	Prototype with some combinations and evaluating them from ease of use and maintenance point of view
3	Deployment Layer – bringing up nodes dynamically as per user needs	To separate simulation layer from deployment and focus on completing simulation first with fixed number of nodes
4	Testing of the simulation	Need to come up with some test plan which will help on deciding whether the simulator meets the client requirements

Appendix

Requirements

Note: H - High, M - Medium, L - Low

Id	Description	Priority
1	Management Server	
1.1	Create a master node responsible for communicating with web client and all other nodes.	H

1.2	Master node will keep track of active nodes	H
1.3	Master node should log information about all the nodes (whether active or not)	L
1.4	Master node will store all the metrics reported by different nodes with timestamp and push them to web client at configurable periodic intervals	H
1.5	Master node will read the user input (simulation script) from files and start simulation accordingly	M
1.6	Master node will frame appropriate control messages which will be sent to the various nodes as per the input simulation script	H
1.7	Decide on format of input file - it should contain information pertaining to setting up of initial network of nodes (with their types). It will then contain information regarding which nodes will do what and at what time.	H
1.8	All the communication to be done using JSON format.	M
2	Media Streaming Part	
2.1	All nodes should be able to send some common metrics like CPU, Memory Usage to the master node at configurable intervals.	H
2.2	Source node should be able to generate data at different rates for given data sizes.	H
2.3	Source node should be able to send data to sink nodes on appropriate ports and able to distinguish between different data streams.	H
2.4	The generated data will be random but there should be facility to add some form of metadata to allow tracking of jitter, latency and packet loss	L
2.5	The source node should report the start time to the master node for tracking purposes	H

2.6	Source node will send data only when some other node sends it a request to send data with appropriate parameters.	H
2.7	Sink node will request data when it is instructed to do so by master node.	H
2.8	Sink node will open a separate port for each request and receive the data independently of other requests going from same node.	H
2.9	Sink node will report end time and packet loss metrics to the master node for each request.	H
2.10	Processing node will perform mainly two types of processing on data - one which consumes memory of machine and another which consumes CPU.	M
2.11	Exact consumption of resources should be configurable (mapped to some parameter range which will be specified in the input file).	L
2.12	Relay node should be able to send data to multiple sink (or other relay) nodes as per the control message received by it from master node.	M
3	Web Client	
3.1	User should be able to start different simulations from the front end	M
3.2	User should be able to view all the nodes in some graph format	H
3.3	User should be able to hover over nodes and edges. Nodes will display metrics like CPU and memory usage and Edges will display metrics like latency and packet loss.	H
3.4	The graph should be updated periodically as per the message received from master node.	M
4	Deployment Layer	
4.1	Entire simulator should be packaged in one executable file which	M

	can be deployed to any machine.	
4.2	The deployment layer should be able to start/stop nodes as requested by the master node on any underlying platform.	L
4.3	It should use generic labels to communicate with master node and depending on underlying platform it should be able to convert these labels into appropriate instructions.	L
5	Non-functional Requirements	
5.1	The source code will be published on Github	M
5.2	The documentations will follow the requirement specified by Ericsson and CMU	M
5.2	The source code should be easy to maintain and extend	H