Media Delivery Network Simulator

Sponsor - Ericsson
Point of contact - Vladimir Katardjiev, Alvin Jude
Faculty Advisor - Jia Zhang
Team - Jeremy Fu, Jigar Patel, Vinay Kumar Vavili, Hao Wang

INI Project Practicum, Fall 2014

Agenda

- Introduction
- System Design and Implementation
- Demo
- System Limitations
- Future Work

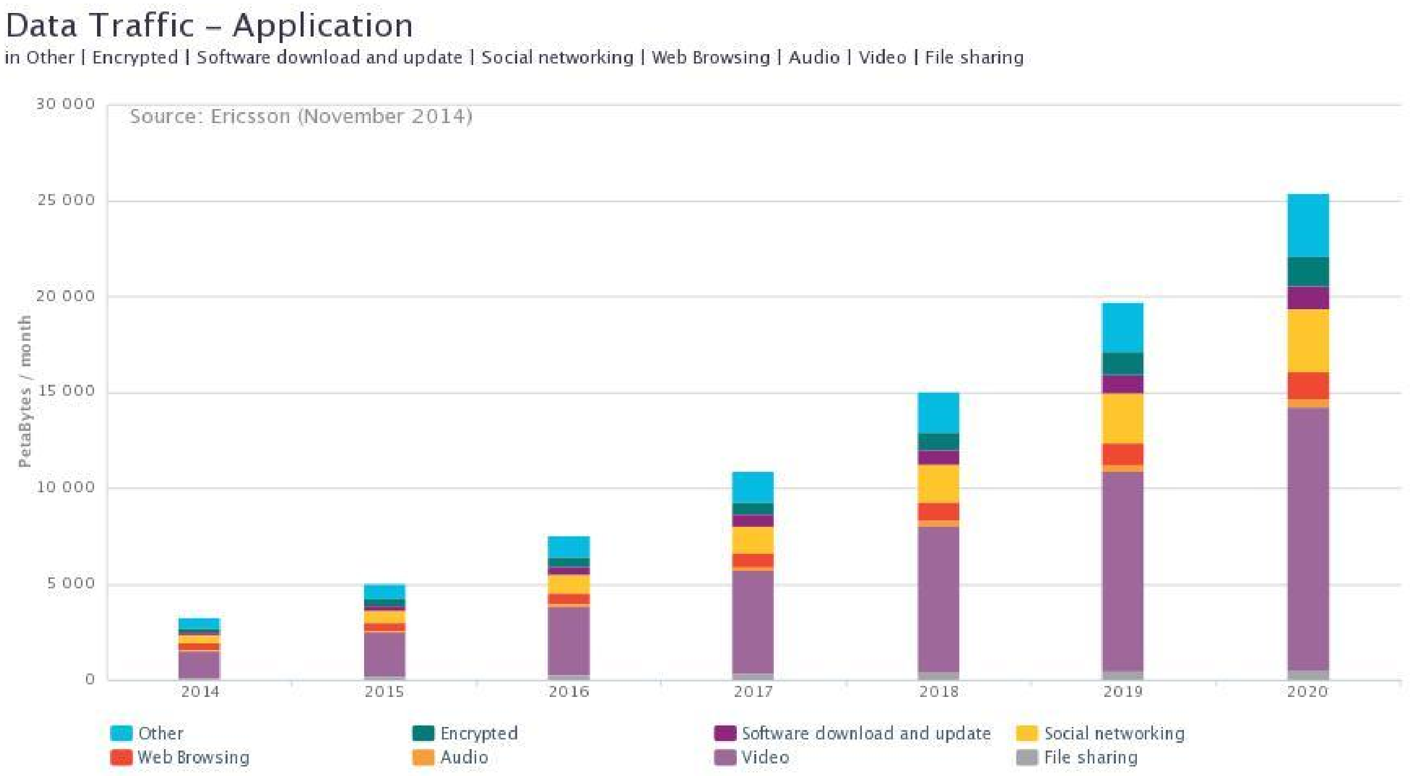
Introduction

Why media delivery network simulator?

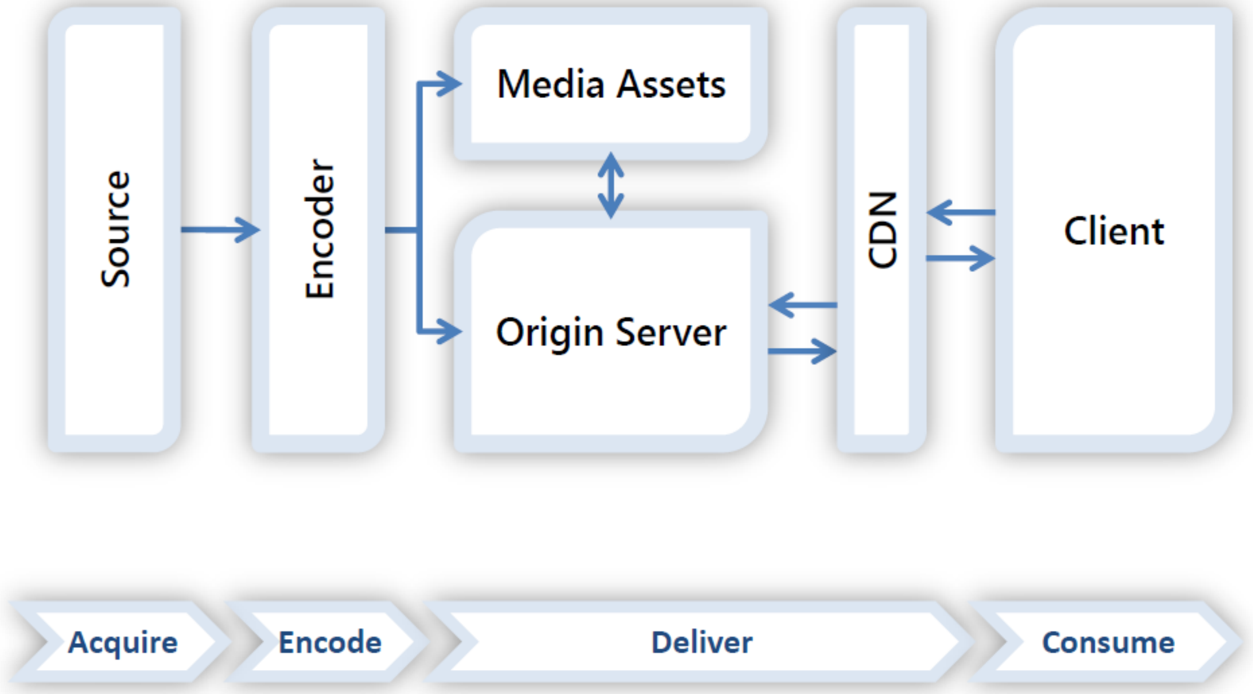
Problem Statement

To build a “life sized” simulation of Internet-based media distribution, with a flexible framework that will allow tinkering, experimentation and evolution.

Traffic Forecast



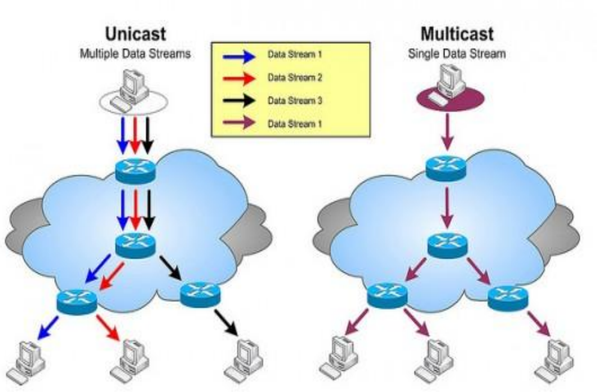
Live media streaming architecture



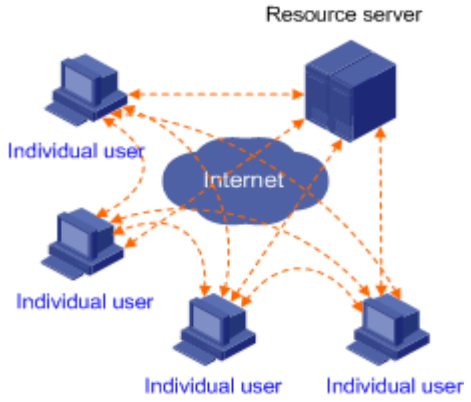
CDN Strategies

- Unicast
- Multicast
- P2P
- Cache-based

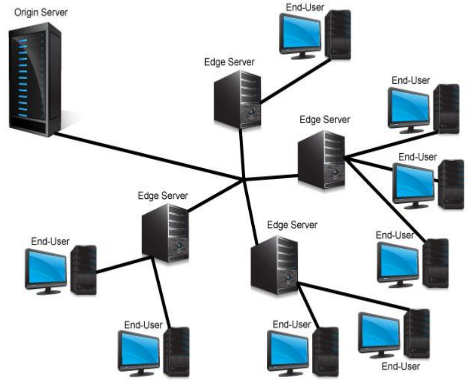
Traditional Unicast/Multicast



P2P



Cache Based



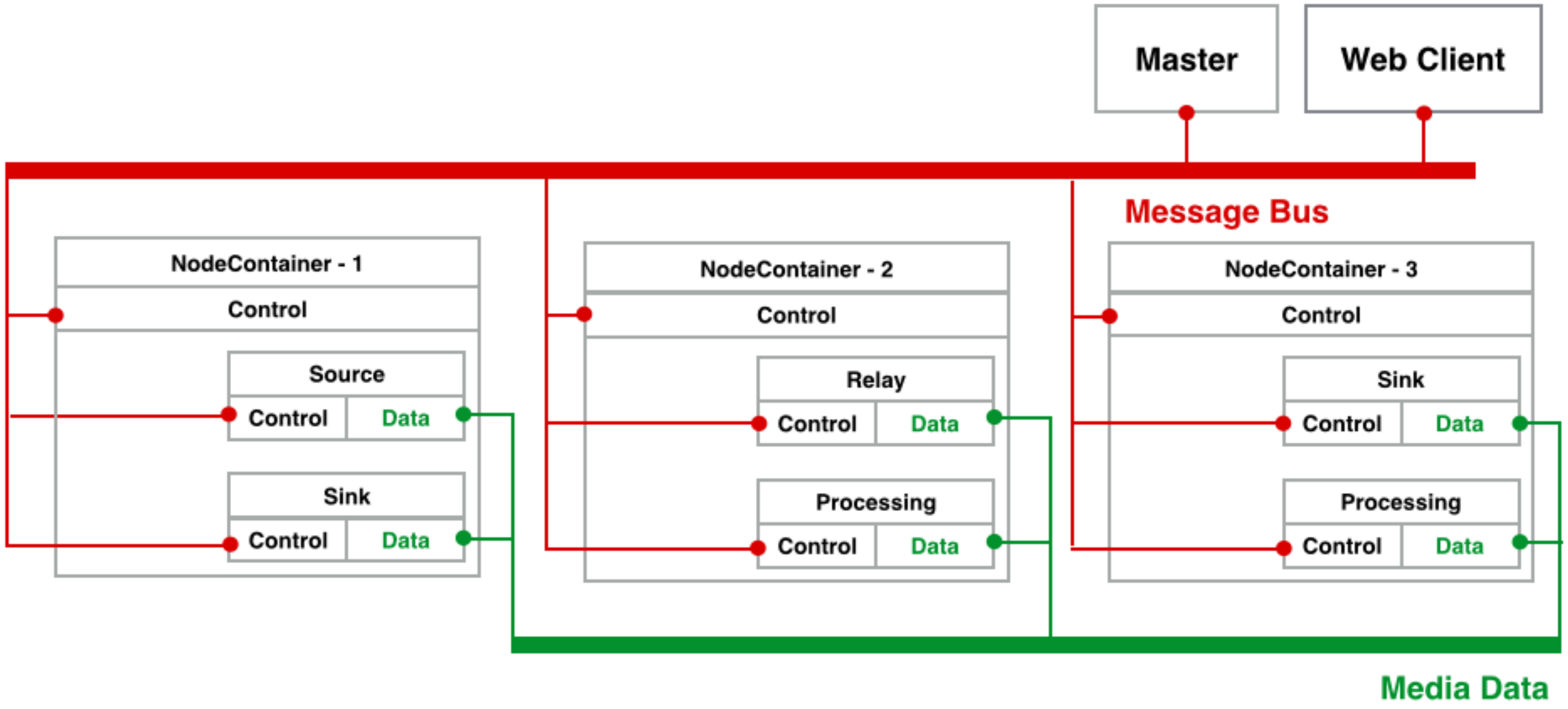
Evaluating media delivery networks

- To evaluate any MDN strategy, we have to
  - Setup source/processing/relay/sink nodes on-demand
  - Transfer data across the ip backbone
  - Measure packet loss, latency etc.,

- The “life-sized simulator” can do the above in a managed and distributed environment based on user input

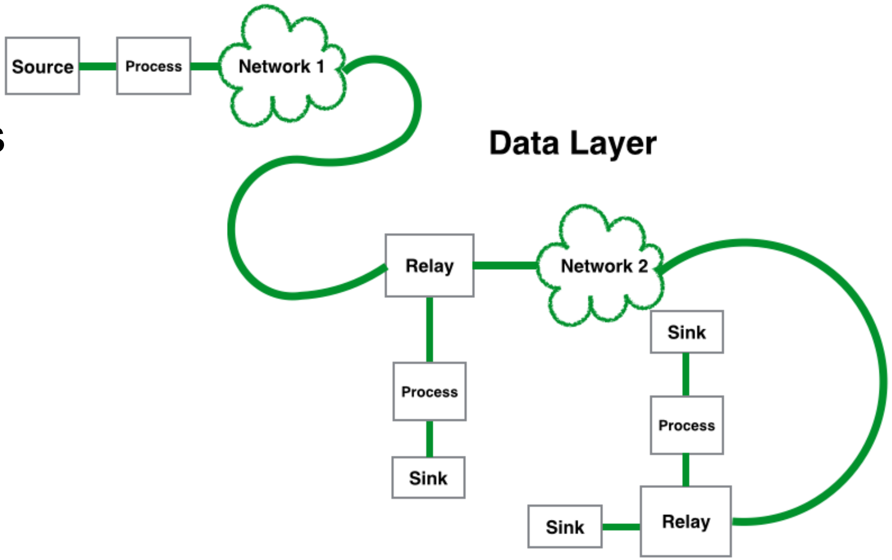
System Design and Implementation

Overview of MDN Simulator



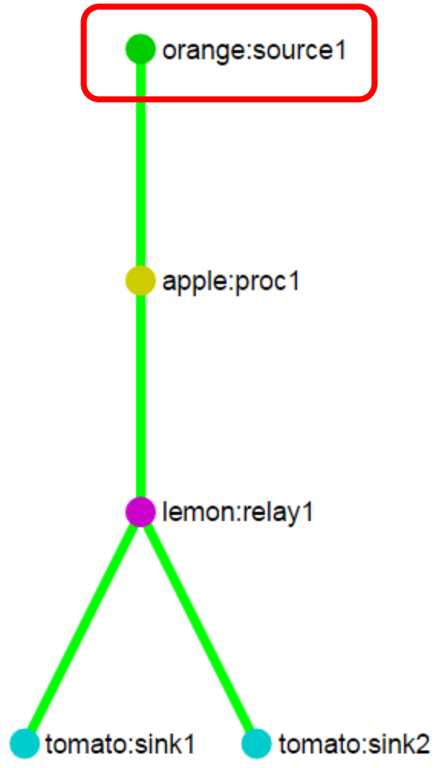
Anatomy of Simulator

- Data Layer
  - Generate traffic
  - Process and deliver to clients
  - Use UDP
- Management Layer
  - Control all components
  - Collect metrics
- User Interface
  - Visualize network
  - Present metrics to user
  - Input to control simulation



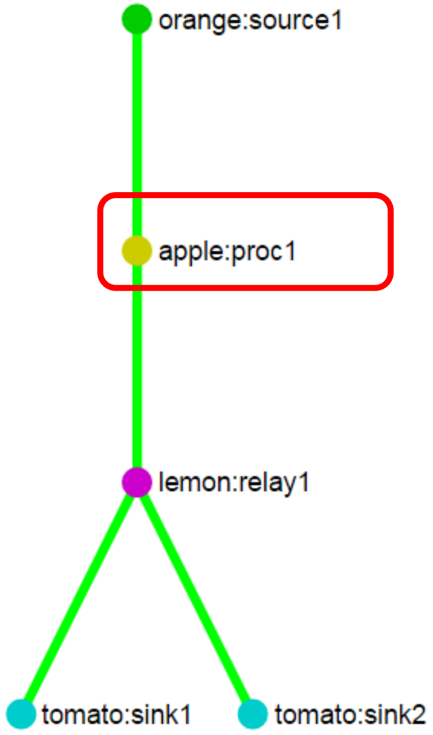
Anatomy of Simulator - Nodes

- **Source Node**: Generate media data
  - Configurable size and bitrate
- **Processing Node**: Perform operations on data
  - Consume CPU and Memory
  - Configurable processing load
- **Relay Node**: Duplicate data and broadcast
- **Sink Node**: Consume media data
  - The entity that requested the stream



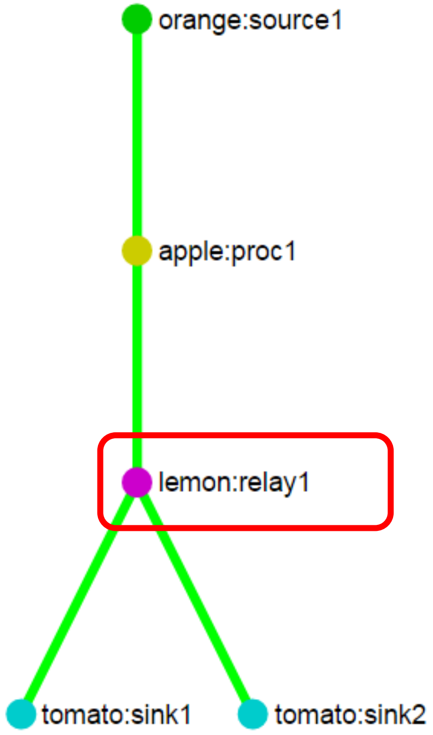
Anatomy of Simulator - Nodes

- **Source Node**: Generate media data
  - Configurable size and bitrate
- **Processing Node**: Perform operations on data
  - Consume CPU and Memory
  - Configurable processing load
- **Relay Node**: Duplicate data and broadcast
- **Sink Node**: Consume media data
  - The entity that requested the stream



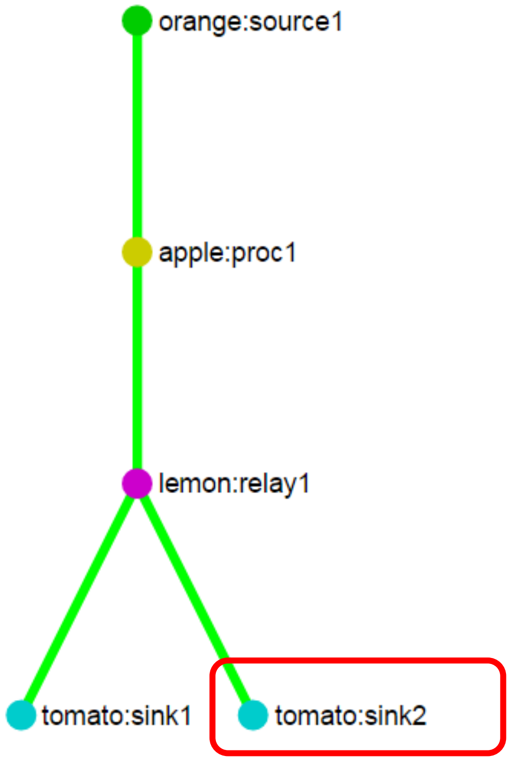
Anatomy of Simulator - Nodes

- **Source Node**: Generate media data
  - Configurable size and bitrate
- **Processing Node**: Perform operations on data
  - Consume CPU and Memory
  - Configurable processing load
- **Relay Node**: Duplicate data and broadcast
- **Sink Node**: Consume media data
  - The entity that requested the stream



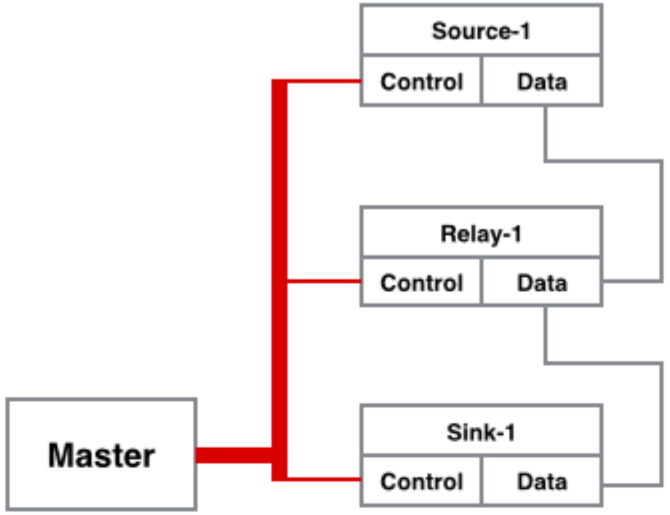
Anatomy of Simulator - Nodes

- **Source Node**: Generate media data
  - Configurable size and bitrate
- **Processing Node**: Perform operations on data
  - Consume CPU and Memory
  - Configurable processing load
- **Relay Node**: Duplicate data and broadcast
- **Sink Node**: Consume media data
  - The entity that requested the stream



Anatomy of Simulator

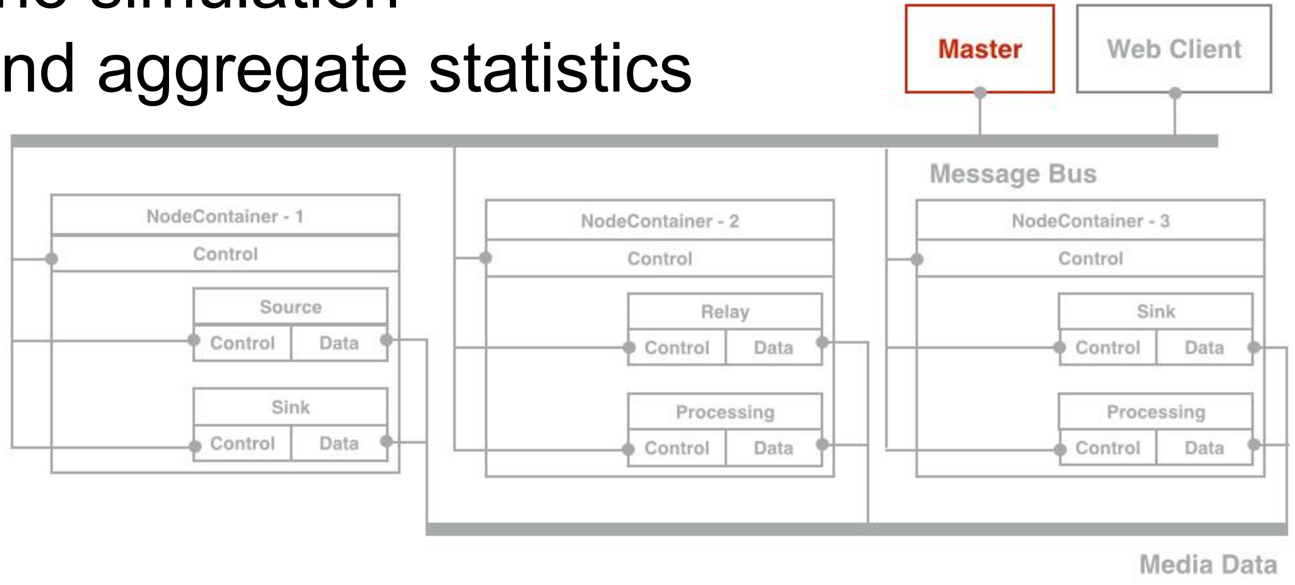
- Data Layer
  - Generate traffic
  - Process and deliver to clients
- Management Layer
  - Control all components
  - Collect metrics
- User Interface
  - Visualize network
  - Present metrics to user
  - Input to control simulation



Message Bus

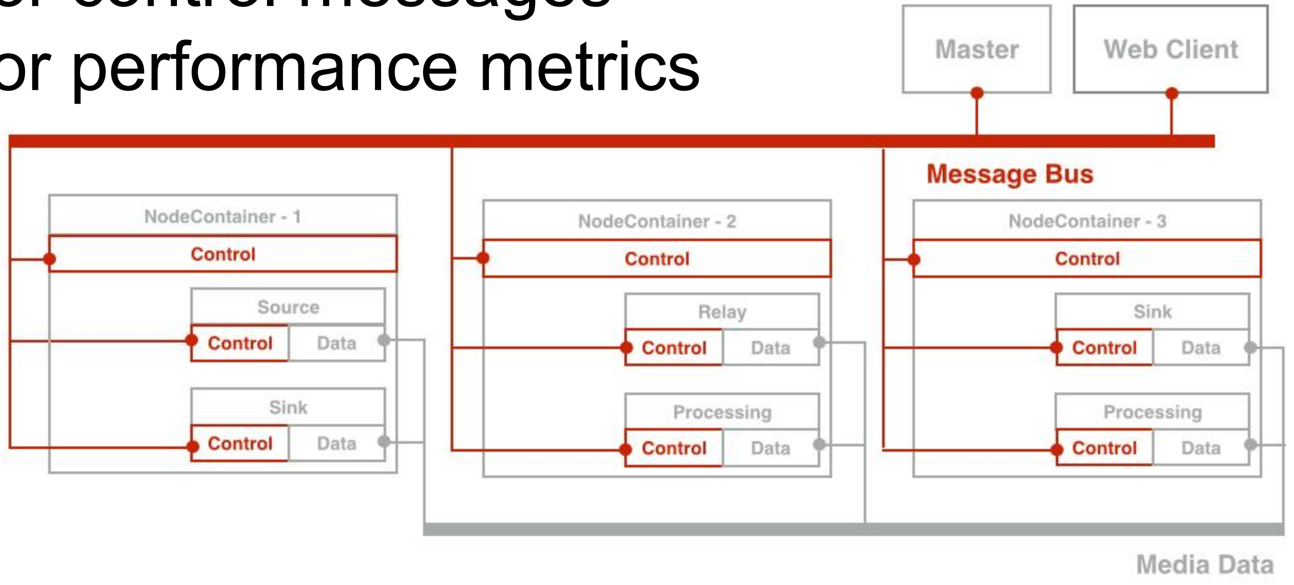
Anatomy of Simulator - Master

- Instantiate nodes
- Controls the simulation
- Collects and aggregate statistics



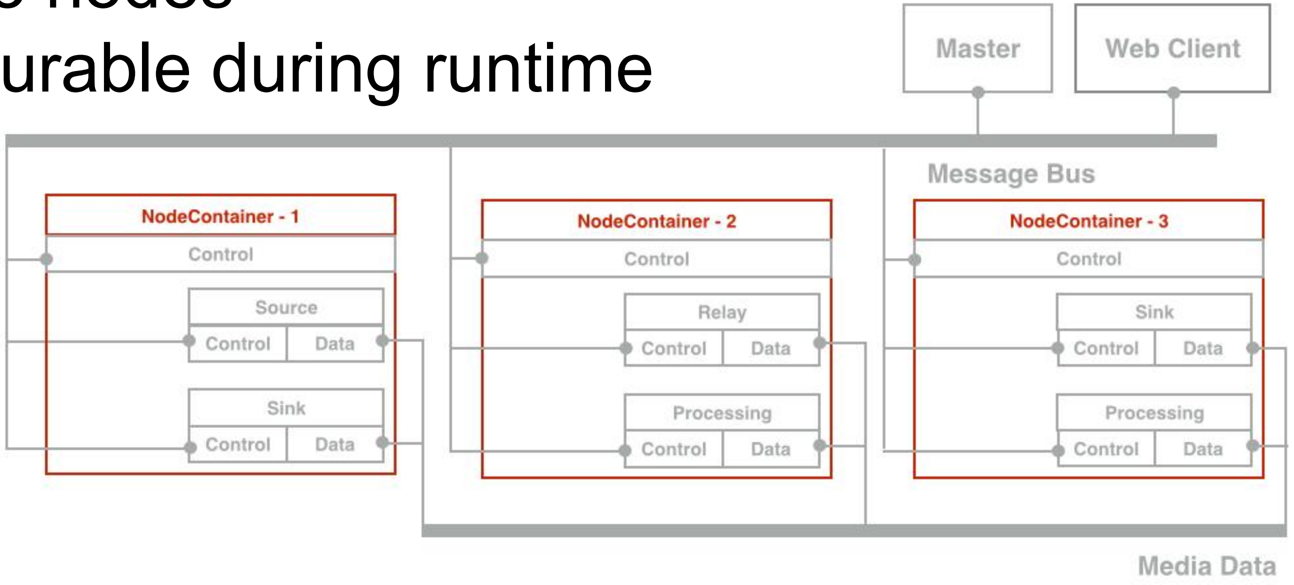
Anatomy of Simulator - Message Bus

- Connect every components of the system
- Channel for control messages
- Channel for performance metrics



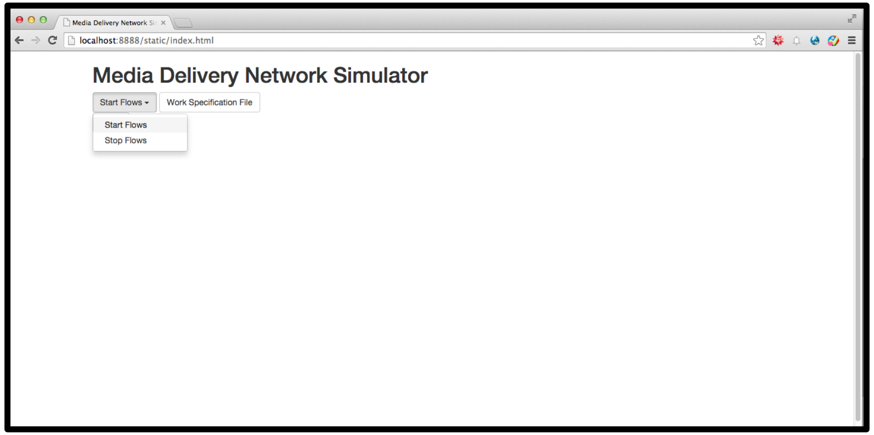
Anatomy of Simulator - Node Container

- Represent one machine in deployment
- Composite nodes
- Be Configurable during runtime



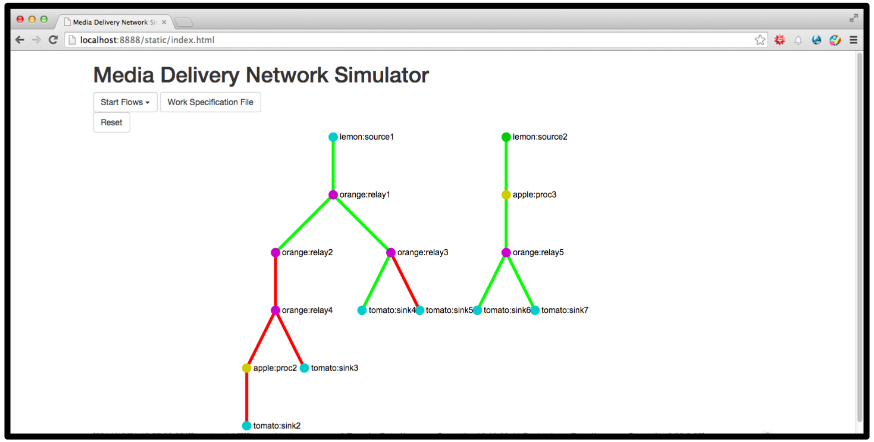
Anatomy of Simulator

- Data Layer
  - Generate traffic
  - Process and deliver to clients
- Management Layer
  - Control all components
  - Collect metrics
- User Interface
  - Input to control simulation
  - Visualize network
  - Present metrics to user


Media Delivery Network Si
localhost:8888/static/index.html
Media Delivery Network Simulator
Start Flows
Work Specification File
Start Flows
Stop Flows

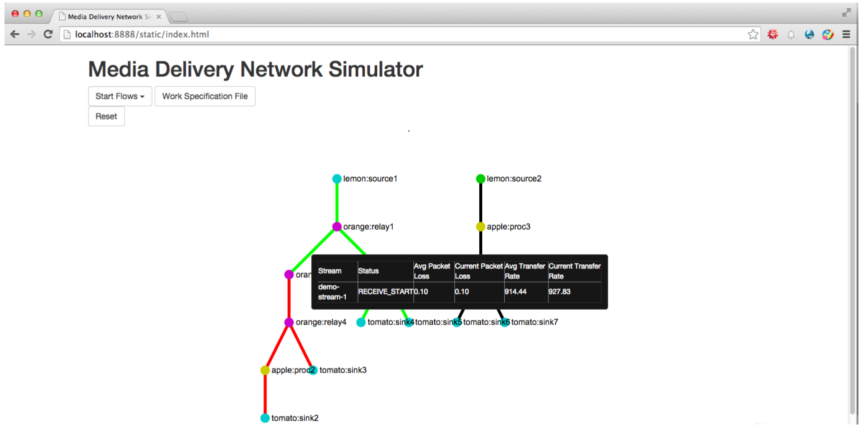
Anatomy of Simulator

- Data Layer
  - Generate traffic
  - Process and deliver to clients
- Management Layer
  - Control all components
  - Collect metrics
- User Interface
  - Input to control simulation
  - Visualize network
  - Present metrics to user



Anatomy of Simulator

- Data Layer
  - Generate traffic
  - Process and deliver to clients
- Management Layer
  - Control all components
  - Collect metrics
- User Interface
  - Input to control simulation
  - Visualize network
  - Present metrics to user



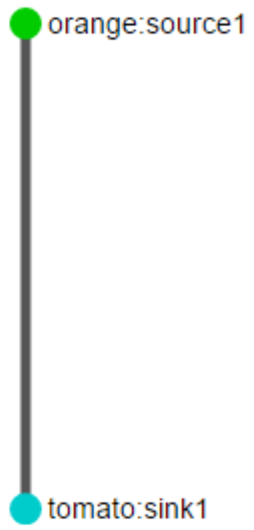
Stream	Status	Avg Packet Loss	Current Packet Loss	Avg Transfer Rate	Current Transfer Rate
demo-stream-1	RECEIVE_START	0.10	0.10	914.44	927.83

Controlling Simulation

- Work Specification
  - Start Simulation
  - Stop Simulation
- Reset

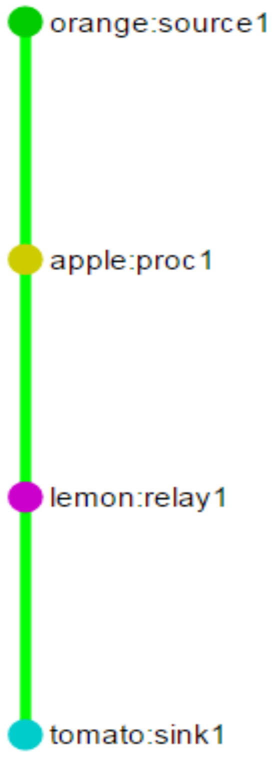
Work Specification Format - Simple

```
{
   "SimId":"demo",
   "StreamList":
   [
      {
         "StreamId":"Stream1",
         "DataSize":"20000000",
         "KiloBitRate":"1000",
         "FlowList":
          [
            {
               "NodeList":
                [
                  {
                     "NodeType":"SinkNode",
                     "NodeId":"tomato:sink1",
                     "UpstreamId":"orange:source1"
                  },
                  {
                     "NodeType":"SourceNode",
                     "NodeId":"orange:source1",
                     "UpstreamId":"NULL"
                  }
                ]
            }
         ]
      }
   ]
}
```



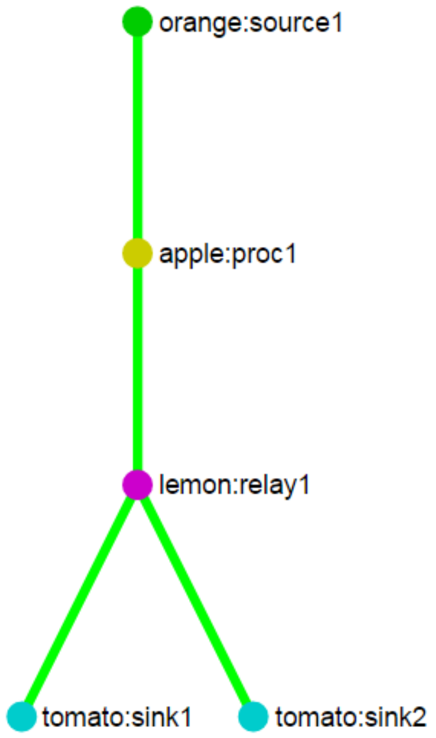
Work Specification - all node types

```
{
   "SimId":"demo",
   "StreamList":
   [
      {
         "StreamId":"Stream1",
         "DataSize":"20000000",
         "KiloBitRate":"1000",
         "FlowList":
          [
             {
                "NodeList":
                 [
                    {
                       "NodeType":"SinkNode",
                       "NodeId":"tomato:sink1",
                       "UpstreamId":"lemon:relay1"
                    },
                    {
                       "NodeType":"RelayNode",
                       "NodeId":"lemon:relay1",
                       "UpstreamId":"apple:proc1"
                    },
.
.
.
.
                    {
                      "NodeType":"ProcessingNode",
                       "NodeId":"apple:proc1",
                       "UpstreamId":"orange:source1",
                       "ProcessingLoop":"100000",
                       "ProcessingMemory":"1000"
                    },
                    {
                       "NodeType":"SourceNode",
                       "NodeId":"orange:source1",
                       "UpstreamId":"NULL"
                    }
                 ]
             }
          ]
      }
   ]
}
```



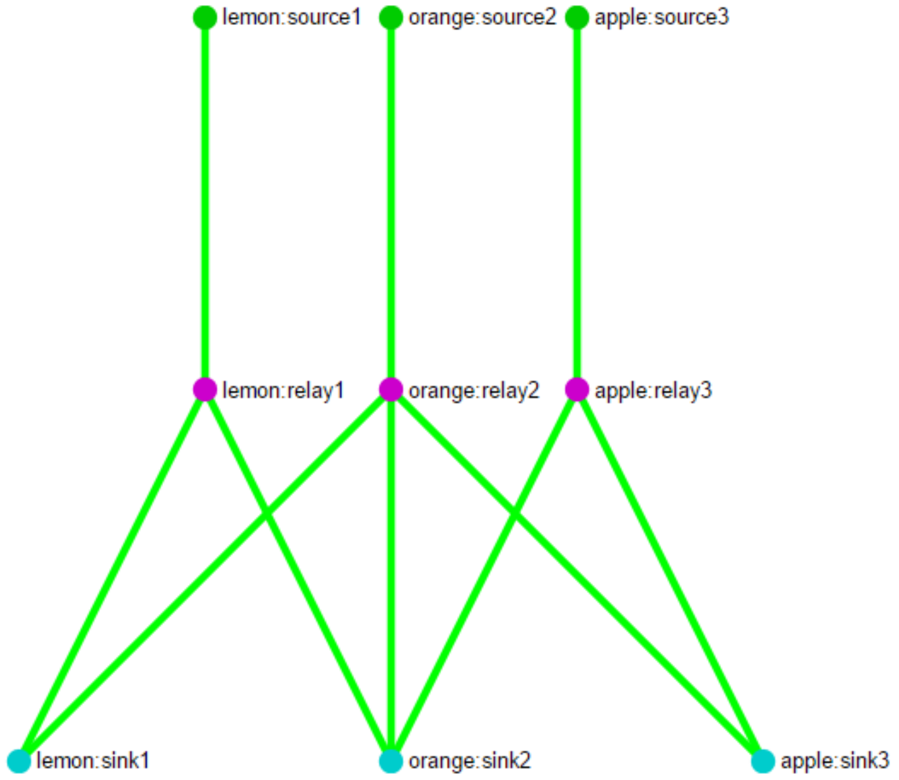
Work Specification - Multiple Flows

```
{
   "SimId":"demo",
   "StreamList":
   [
      {
         "StreamId":"Stream1",
         "DataSize":"20000000",
         "KiloBitRate":"1000",
         "FlowList":
          [
            {
               Flow 1 (same as previous slide)
            },
            {
               "NodeList":
                [
                  {
                     "NodeType":"SinkNode",
                     "NodeId":"tomato:sink2",
                    "UpstreamId":"lemon:relay1"
                  },
                  {
                     "NodeType":"RelayNode",
                     "NodeId":"lemon:relay1",
                     "UpstreamId":"apple:proc1",
                  },
                  {
                     "NodeType":"ProcessingNode",
                     "NodeId":"apple:proc1",
                     "UpstreamId":"orange:source1",
                     "ProcessingLoop":"100000",
                     "ProcessingMemory":"1000"
                  },
                  {
                     "NodeType":"SourceNode",
                     "NodeId":"orange:source1",
                     "UpstreamId":"NULL"
                  }
                ]
            }
          ]
      }
   ]
}
```



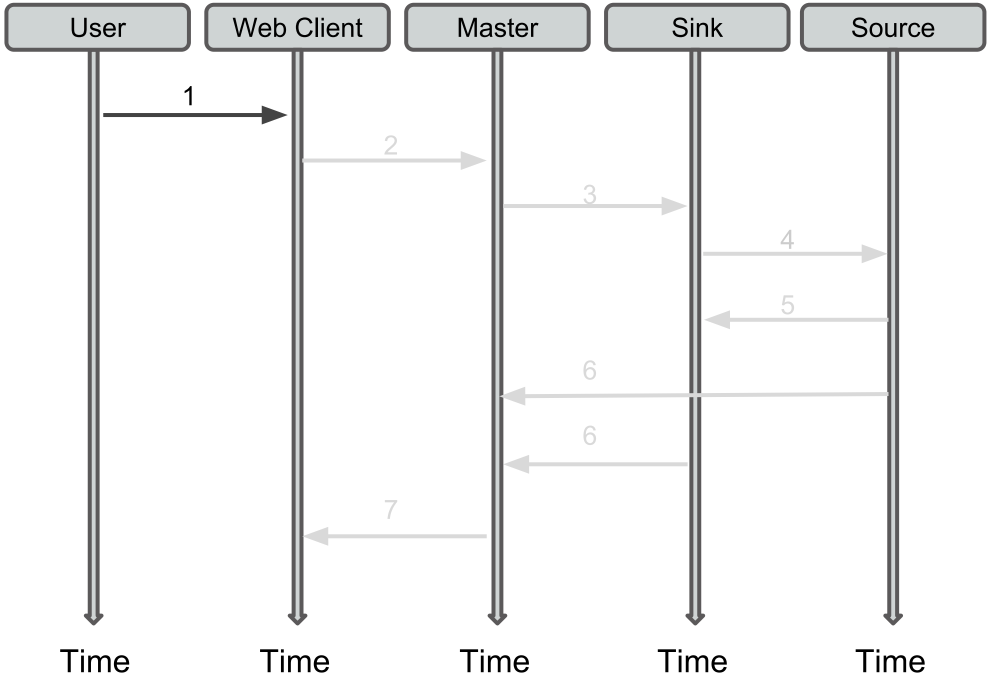
Peer 2 Peer Topology

Each Node Container
is considered as one peer
node of P2P network



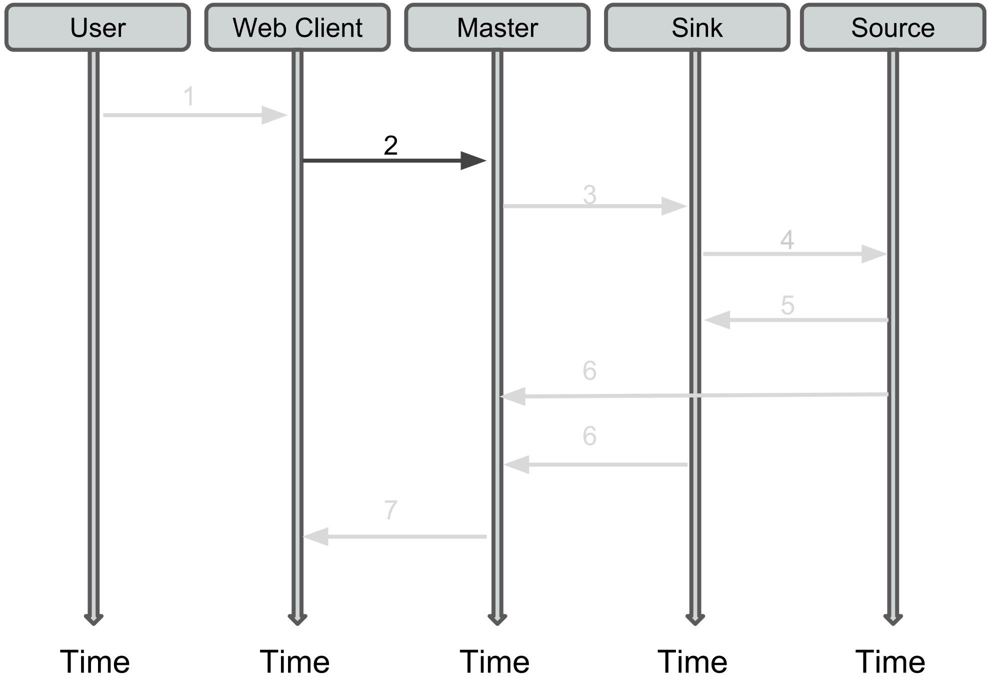
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



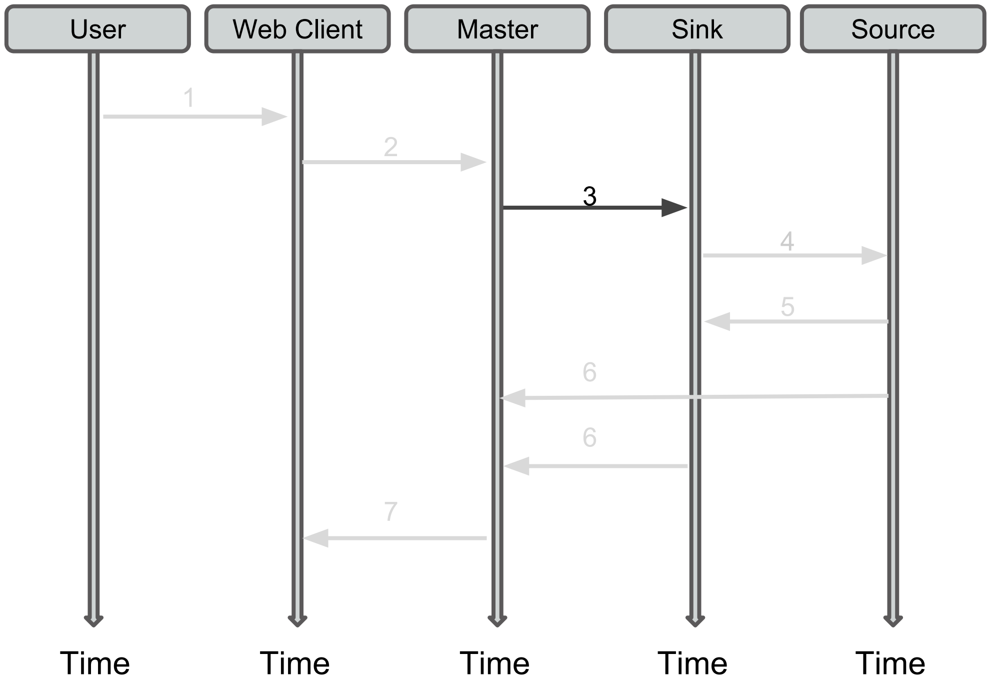
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



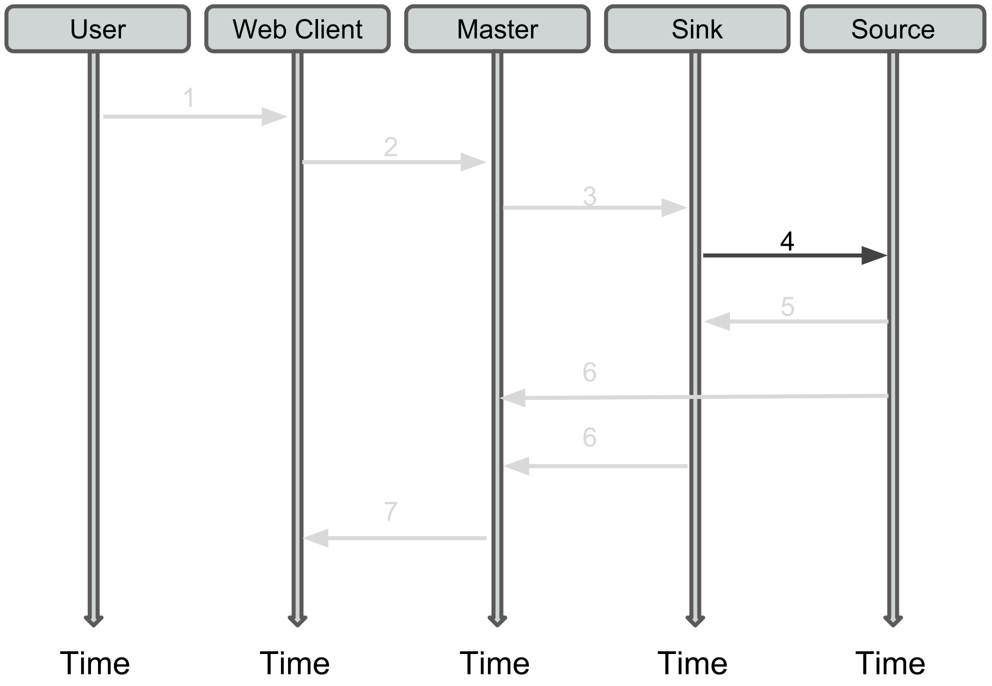
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



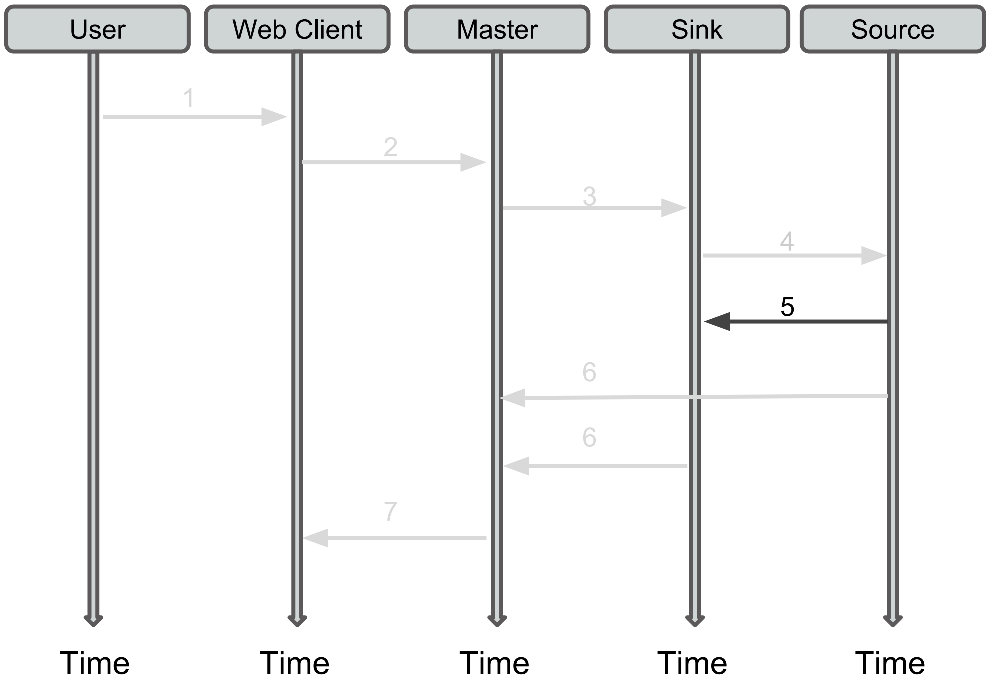
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



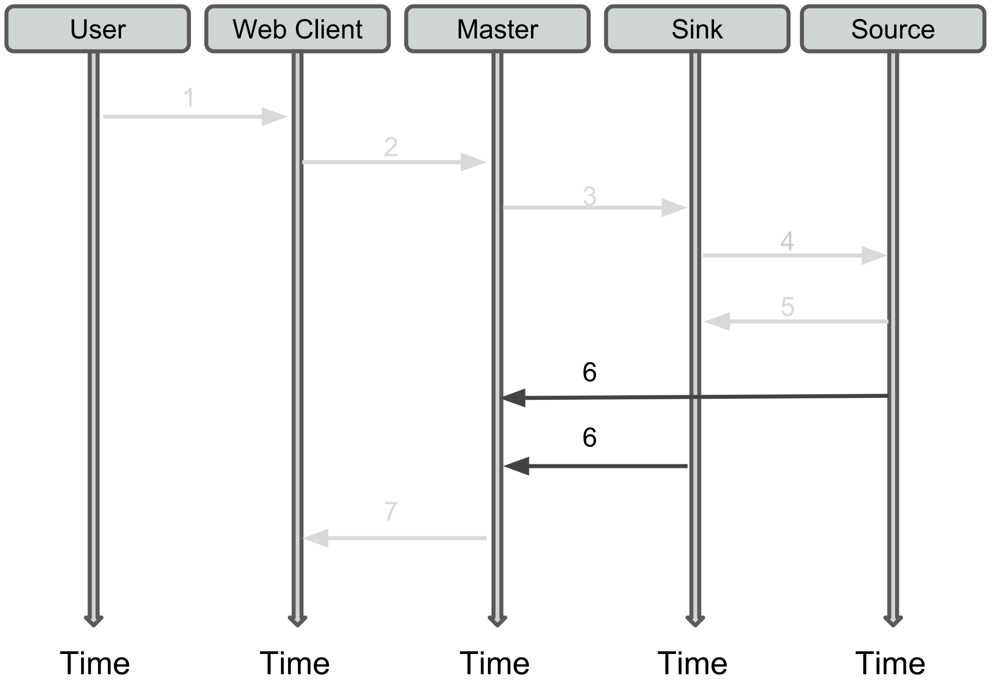
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



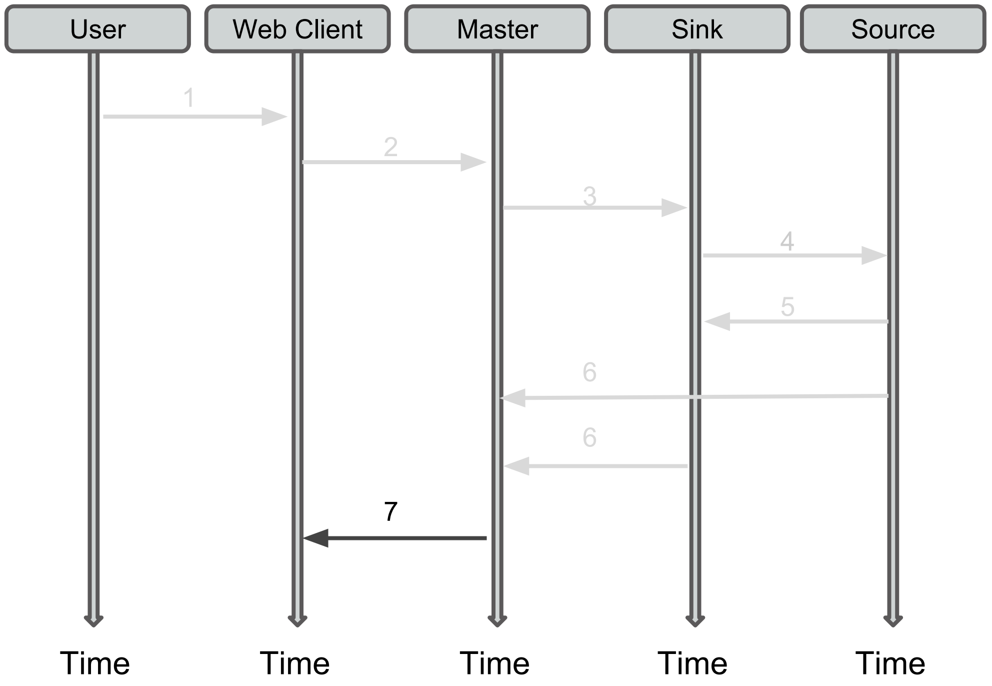
Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



Work Flow of the simulator

1. User uploads Work Specification
2. Master receives Work Specification
3. Updated Flow sent to Sink (for every flow in Work Specification)
4. Sink sets up listening port and forwards Flow to upstream
5. Source starts sending data
6. Periodic metric updates to master
7. Periodic update of Web Client



Demo

Contributions

- Different elements(nodes) of the MDN
- Extensible nodes
- Actual data loads
- Configurable traffic loads
- Immediate feedback
- Centralized control

Limitations

- Manual deployment of node containers
- Number of nodes in a node container
- Manually coded Work Specification
- Clock drift

Future Work

- Node containers Auto-deployment
- Maximum node number enhancement
- Dynamic marker insertion in data traffic
- Improvements in visualization part
- UI based Work Specification generator

Tried and discarded

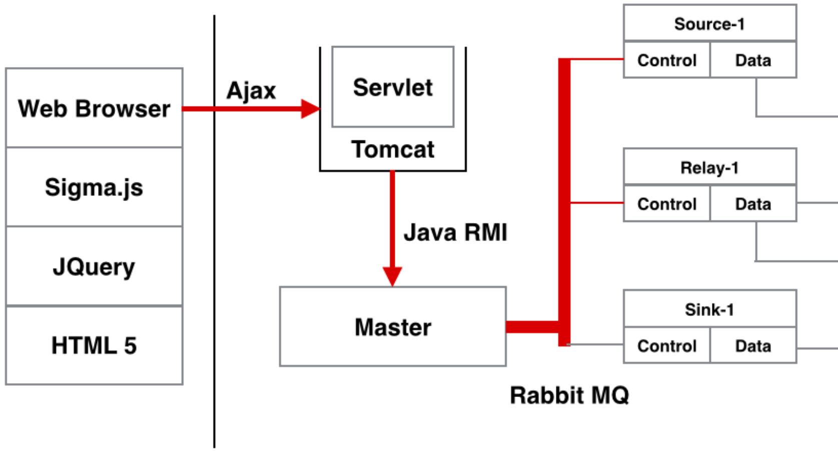
- **System Design & Development Progress**

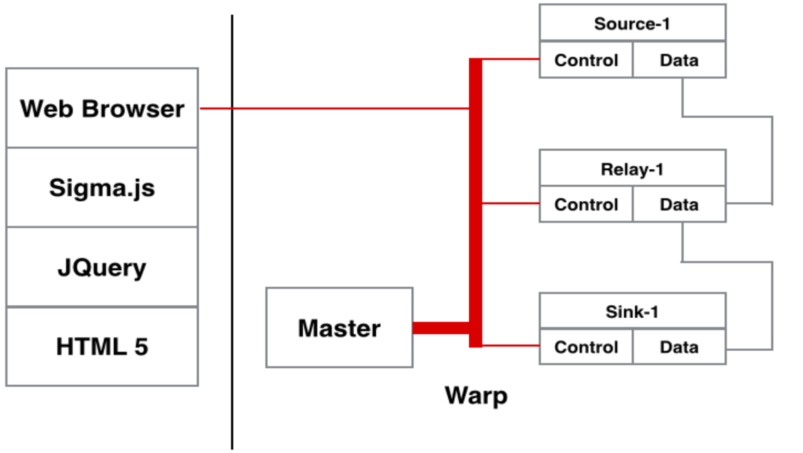
  Initial Design & Implementation

  - Front-end: HTML5, JQuery, Sigma.js
  - Back-end: Tomcat, Java Servlet
  - Inter-process call:
    - Ajax
    - Java RMI
    - RabbitMQ

  Evaluation of Initial Design

  - Multiple communication mechanisms
  - Too many dependencies
  - No unified interfaces for control messages





Tried and discarded

- Ns2, Gns3 / NetSim, Mininet, Omnet++
- Not Scalable: Simulations run on a single machine.
- Nodes simulate L2 to L4 of the network stack and don't have functionality to simulate L7 processing.

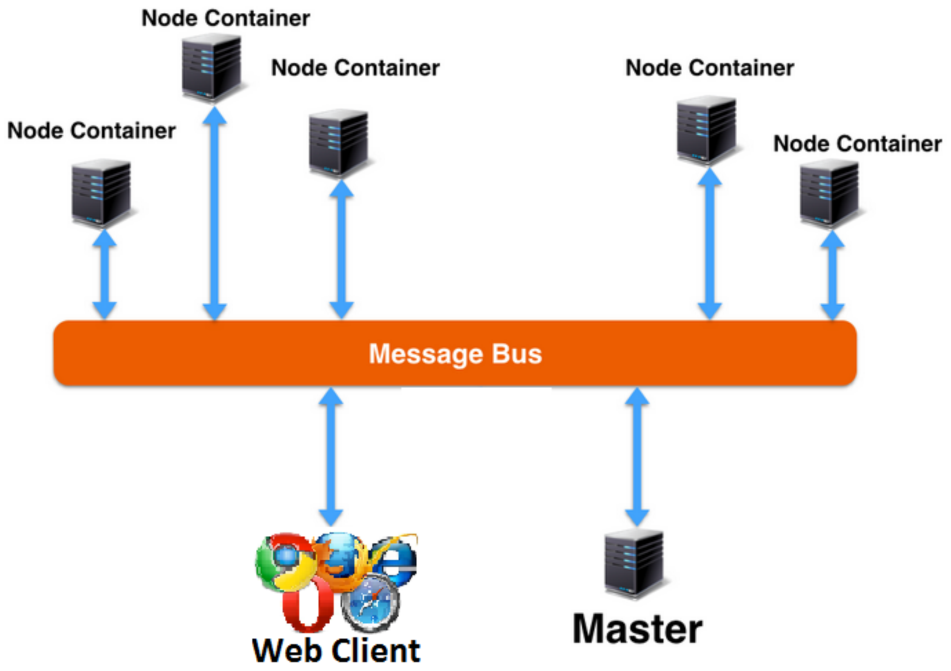
Acknowledgement

- We would like to thank Ericsson Research for sponsoring this project and providing us with this opportunity. We would like to specially thank Vlad, Jia and Alvin for guiding us through this project providing us with valuable feedback on design of the system.

Questions

High Level View of MDN Simulator

- Master
  - Control the system
  - Collect metrics
- Web Client
  - Visualize nodes
  - Present metrics to user
  - Input - Control simulation
- Node Container
  - Host and manage the nodes
- Message Bus
  - Communication link between all components



Components

- **Master Node**: The server that instantiates/deploys the nodes, collects and aggregates statistics and controls the overall simulation
- **Web Client**: The web interface that lets us interact with the system and visualize the state of the system
- **Message Bus**: A decentralized messaging framework that lets us transfer control messages in a reliable manner separately from the data traffic
- **Node Containers**: A remote process that hosts various node types of the simulation

Input

- **Work Specification**: The input file given by the user that has a list of streams and stream parameters
- **Stream**: A unique media entity like a movie or audio clip that has a defined size and bitrate. A stream has a list of flows that have requested this stream
- **Flow**: A unique path that a stream follows from the source to sink

The IP Imperative

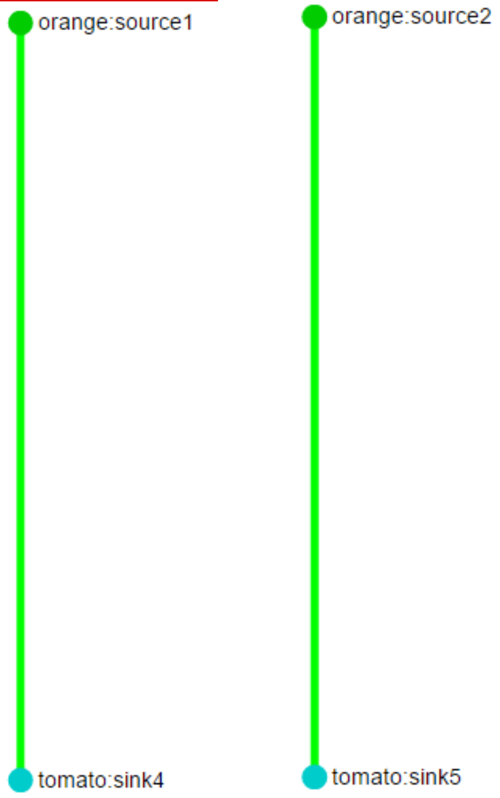
- More than 90%of the world’s population will have access to mobile broadband by 2020
- Global mobile data traffic is expected to grow over 10x by 2020 with video predicted to be 50% of all this traffic
- In 2020, there are expected to be 15 billion video-enabled connected devices globally
- And, in advanced markets, 50% of content watched is on-demand

[Source: http://www.ericsson.com/televisionary/blog/ericsson-media-vision-2020-game-changer-two-ip-imperative/]

Work Specification - Multiple Streams

```
{
"SimId":"sim1",
"StreamList":
 [
   {
         "StreamId":"Stream1",
         "DataSize":"20000000",
         "KiloBitRate":"1000",
         "FlowList":
          [
            {
               "NodeList":
                [
                  {
                     "NodeType":"SinkNode",
                     "NodeId":"tomato:sink4",
                    "UpstreamId":"orange:source1"
                  },
                  {
                     "NodeType":"SourceNode",
                     "NodeId":"orange:source1",
                     "UpstreamId":"NULL"
                  }
                ]
            }
          ]
   },
```

```
    {
          "StreamId":"Stream2",
          "DataSize":"20000",
          "KiloBitRate":"50",
          "FlowList":
           [
             {
                "NodeList":
                 [
                   {
                      "NodeType":"SinkNode",
                      "NodeId":"tomato:sink5",
                      "UpstreamId":"orange:source2"
                   },
                   {
                      "NodeType":"SourceNode",
                      "NodeId":"orange:source2",
                      "UpstreamId":"NULL"
                   }
                 ]
             }
           ]
    }
  ]
}
```



The IP Imperative

- The number of IP connected devices that can view video has grown from 200 million (personal computers) to over 1.6 billion in the period 2000-2013 alone. In 2020, there are expected to be 15 billion video-enabled connected devices globally
- Global data traffic is expected to grow over 10x by 2020 with video predicted to be 50% of all this traffic
- And, in advanced markets, half of content watched is on-demand

[Source: http://www.ericsson.com/televisionary/blog/ericsson-media-vision-2020-game-changer-two-ip-imperative/]