|  |
| --- |
| QUEEN’s university |
| ELEC 476 Project Routing in ICN – Queue Model |
| Progress Report #1 |
|  |
| **Doug Marr, Daniel McLean** |
| **2/26/2013** |

|  |
| --- |
|  |

## Introduction

Information-Centric Networking (ICN) is an up and coming technology that can potentially replace the current IP networks in place around the world. ICN works by being centralized around data instead of hosts. By being data centric, a consumer can request data and the closest (or best fit) host of the data can transfer it to the consumer without the consumer being aware of which hosts have this data. This will achieve much higher transfer rates and network utilization.

While this is good in concept, ICN has a few implementation problems. Further simulation is needed to study the feasibility of an ICN. This project focusses on the routing problem. Routing is difficult in ICN because it the network needs to be reactive to open data requests instead of host directed data request **[does that make sense?]**. Our model is an ICN network with a centralized controller which controls the routing. This is perhaps the best way to introduce ICN to existing networks. The controller will be aware of the network and made routing decisions based on how busy the network is and where the data is located.

Our model uses different two-phase routing methods that will analyzed with the M/G/1 queue model. The network itself is randomly generated to try to achieve a realistic model. However some assumptions are used to keep some simplicity. The network will remain static once simulated. There will be no network failures or transfer failures. All the data will be the same size and will take a constant delay to transfer from one node to the queue of the next node.

The objective of these simulations will be to determine which two-phase routing method is best for many different network topologies.

## Background

To achieve a useful simulation a realistic model must be implemented. Our approach is to use random processes where feasible while keeping other elements fixed for simplicity. Our approach is to randomly generate a topology and randomly assign cache sizes, data to producers, and delay between nodes. The model will keep a static network after initial generation. Each producer will have the same storage size which will be initialized full (randomly) and will remain constant after generation. Each switch and producer will have a fixed queue size. The decision of what element is random and which is static is made based off of which is most important for analyzing routing methods. The consumer will randomly produce requests which go directly to the controller’s queue. The controller will then send the request and routing formation to the appropriate node.

The controller, which handles the routing decisions, is most effected by the topology, how busy the producers/switches are (how many elements in their queue) as well as the data request. Having random queue and storage sizes would be more realistic however it would add a large amount of complexity with little change to routing performance as the routing decisions would behave the same or very close to that of a network where these elements are randomized.

The network remains static after it is initially generated to give simplicity to the system. Having a model where the network can change during simulation is more realistic and will effect routing adaptation, however it adds a lot complexity. This decision remains realistic since our simulation model will look at which routing method gives the best performance and not the network adaptation. This is justified because network adaptation in a controller based ICN should be an added feature of a routing method which gives the best performance. **[is that ok to say?]**

## Simulation Setups

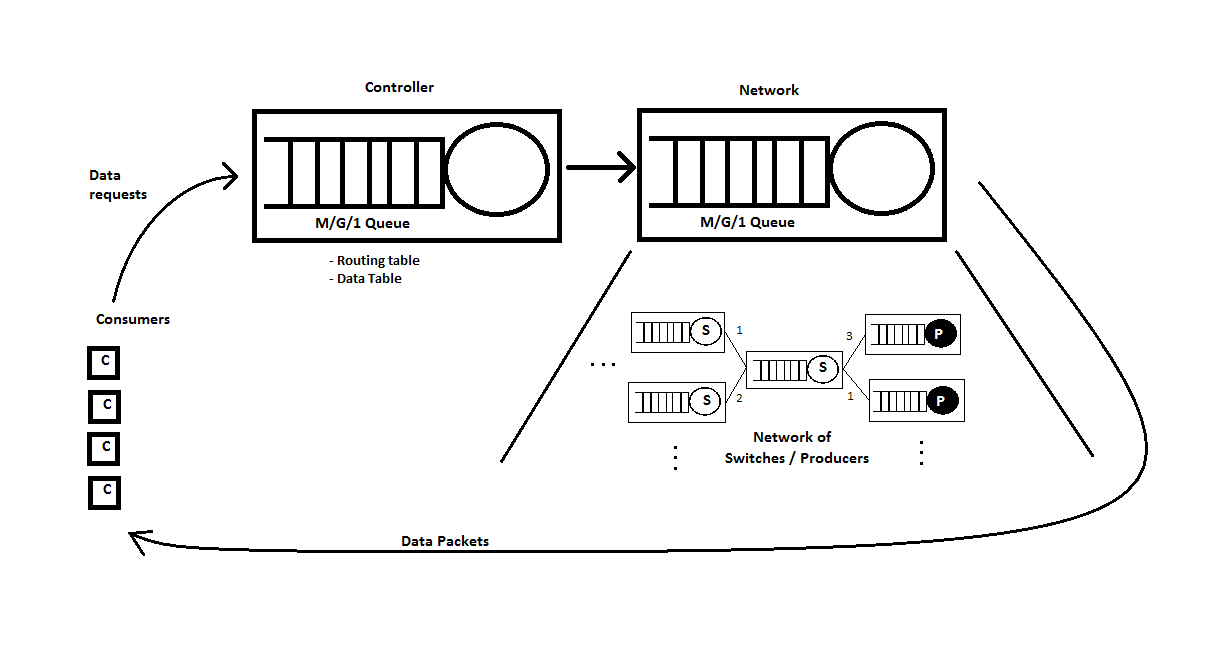
The system is simplified by separating the nodes based on type. The consumer nodes function as system inputs as they are data sinks, have no queues or caches, and only send data requests to the controller for processing. All controller inputs come from the consumers and all outputs flow to either the switches or producers. The controller is modelled using an M/G/1 queue. All data originates at the producers, which have static caches. All network jobs flow from the controller (after routing has occurred) to either the producers or the switches, which have dynamic caches. Since the switches and producers all have queues and form a Jackson’s network, they too can be modelled using an M/G/1 queue. Therefore the result is two queues and servers in series with one another.

Performance Metrics – As both sides of the system are to be represented using a queue model, the primary consideration will be the performance of the queues. That is, the server utilization of the controller and producers as well as the service time of jobs in the controller and in the network.

Parameters – Within the controller, routing is to be the primary metric. Possible routing schemes include shortest path, cost-based shortest path (Dijkstra’s Algorithm), or multipath routing. For the switches and producers, queue size, cache size, as well as the overall network size, node type ratio, and distribution will play a large role in system performance. Queue size of the controller, switches, and producers will also be factor.

Validation – The system will have three stages: Initialization, stabilization, and equilibrium. During initialization, the network topology will be generated, the data will be distributed among the producers, and the controller routing table will be built. During stabilization, data requests will begin, routing will occur, and data will propagate throughout the switch nodes. Collection of system simulation data will begin. During equilibrium, system performance is expected to plateau as the switch caches will fill with the most popular data. Validation will require identification of the transition between stabilization and equilibrium so the collected data can be separated and analyzed. Calibration will likely be required in order to mitigate issues (ex. excessive service times, jobs dropped, etc.) and produce a reasonable network representation.

Verification – Network topology will be verified by generating a 2D graphical representation of the network. The routing table and data distribution will also be examined to verify logical initial conditions. While running, the simulation will output graphs of the current system state. This will then be saved in table format so it can be exported to excel and analyzed and compared to actual system performance data.



## Overall Progress

Complete Working Pending

- System conclusions

- Performance recommendations

- Structure recommendations

- Extract relevant data/remove outliers

- Output data analysis

- Output data visualization

- Validate system performance

- Calibrate parameters

- Match to real system performance data

- Confirm parameters

- System Component Integration

- Exercise

- Collect Data

- Verify Initial conditions

- Validate component performance

- Small scale testing (ie. 2-3 components)

- Create World: network topology generation, routing table, data distribution table, RV generation method

- Build Node Classes

- Implement node communication

- Research

- Define Problem

- Define Assumptions (Data/Structure)

- Verify Model

- Set Objectives