# Video streaming code documentation

Richard G. Clegg, Raul Landa, David Griffin, Miguel Rio

This document describes the inputs, outputs and code structure for the code to simulate video streaming. It should be read in conjunction with the mathematical model description. The basis of the code is to split the simulation task into separate models:

1. Demand model: responsible for working out when and where users arrive and depart streams.
2. Network model: responsible for tracking where servers are.
3. Stream model: responsible for assigning users arriving to individual streams (users who are communicating with each other).
4. Routing model: responsible for working out how traffic between two users travels via available servers.
5. Server model – given a number of data centres and a set of user demand split into “streams” this model chooses which data centres are available to a stream.
6. Routing model – given a number of communicating users (from the stream model) and allowed data centres (from the server model), how will traffic between the users be routed.
7. Quality of Experience (QoE) model – given set of users (from the stream model) and a set of routes (from the routing model), what QoE will the users experience.
8. Cost model – given set of users (from the stream model) and a set of routes (from the routing model), what will the infrastructure provider charge.

## Model inputs

The model takes inputs which specify the demand profile, the nature of the simulation to be performed and the nature of the output required. The main input file is a structured JSON file which specifies all the parameters and is user editable. Inputs can be compulsory or optional. For model components the "type" parameter refers to a python class. This enables new classes for each model to be written according to a base class template making the model highly extensible.

### An example JSON configuration file is shown below.

### {

### "demand\_model": {

### "type": "demandmodel.poisLogDemand.poisLogDemand",

### "locations": "../geo/VideoDistributionModel/out/poker.out",

### "time\_demand": "../geo/VideoDistributionModel/out/fake\_time.out",

### "mean\_daily\_arrivals": 10000,

### "session\_mean": 7200.0,

### "session\_sigma": 1.24

### },

### "stream\_model": {

### "type": "sessionmodel.pokerModel.pokerModel",

### "max\_room": 10,

### "min\_room": 4

### },

### "network\_model": {

### "type": "networkmodel.simpleNetwork.simpleNetwork",

### "file": "../geo/VideoDistributionModel/out/ec2\_price.out",

### "between\_cost\_dollar\_GB": 0.01,

### "full\_stream\_MBs": 1.0,

### "compressed\_stream\_MBs": 0.1

### },

### "route\_model": {

### "type": "routemodel.hotPotatoRoute.hotPotatoRoute"

### },

### "qoe\_model": {

### "type": "qoemodel.haversine.haversine"

### },

### "server\_model": {

### "type": "servermodel.nRandom.nRandom",

### "number": 2

### },

### "output\_models": [

### {

### "type": "outputmodel.basicDaily.basicDaily",

### "file": "pokerdaysummary.txt"

### },

### {

### "type": "outputmodel.basicDaily.basicDaily"

### },

### {

### "type": "outputmodel.outputCost.outputCost",

### "file": "pokercostsummary.txt"

### },

### {

### "type": "outputmodel.qosPdf.qosPdf",

### "file": "pokerqoesummary.txt",

### "style": "quintile",

### "time": "daily"

### }

### ],

### "simulation\_days": 100

### }

### Notes

The structure has separate sections for each model and for output as well as a general section. Each model begins with a “type” which refers to a python class. This allows the quick addition of new models into the structure. For example "routemodel.hotPotatoRoute.hotPotatoRoute" refers to the file routemodel/hotPotatoRoute.py which contains a class hotPotatoRoute. Each of the classes so referenced must be derived from the appropriate base class for that model type.

### General inputs

"simulation\_days": Compulsory. Gives the number of days to simulate.

### Demand model inputs

This section specifies the demand model (see description in the mathematical overview). The only type developed so far is the Poisson input LogNormal dwell time model. Users arrive as Poisson processes modulated by local time of day and location and stay for a duration given by a lognormal distribution.

#### Demand model demandmodel.poisLogDemand.poisLogDemand

The options are:

"locations": Compulsory. Gives the location of a csv file which gives location lat, long, rate triples in the following format:

50.5,-23.0,12.2

-60.0, 40.0, 14.3

-34.2, 22.0, 15.5

Where each line represents a latitude (negative is south of the equator), a longitude (negative is west of Greenwich) and a rate multiplier. Optional: A subcontinental zone identifier.

"time\_demand": Compulsory. Gives the location of a csv file which gives a list of the time period multipliers for the model in the following format:

1.5

2.5

2.4

6.7

7.8

These are given as separate files so they are “pluggable” – that is the location from one model can be easily combined with the time profile from another.

"session\_mu","session\_sigma": Optional, compulsory. Gives the parameters for the lognormal session length distribution. If the session\_mu is not specified then session\_mean must be.

"session\_mean": Optional. Gives the mean length in seconds of a user’s session in a stream. If this is omitted then session\_mu must be specified.

"arrival\_rate": Optional. Gives the overall arrival rate in persons per second.

"mean\_daily\_arrivals": Optional. Gives the average number of users arriving per day

### Session model inputs

This section describes how the demand should be mapped to streaming sessions. Separate models will be developed for the MOOC and the Poker scenario as these different scenarios map users to chat rooms in different ways.

#### Poker model sessionmodel.pokerModel.pokerModel

This takes two parameters. min\_room and max\_room which describe the minimum and maximum people in a poker room. They are both compulsary. If a room has space then a user joins a random room with space. If no rooms have space the user joins a waiting queue. If enough people are in the waiting queue then it becomes a room.

#### Mooc model sessionmodel.moocModel.moocModel

This takes no parameters. All users enter the saem session

### Route model

Separate models for routing are used. hotPotatoRoute means that traffic routes via only a single router to minimise delay. On the other hand stayOnRoute means that traffic stays on the managed network for as long as possible by entering the network at the closest place to the origin and leaving at the closest place to the destination (close in delay terms).

#### Route model routemodel.hotPotatoRoute.hotPotatoRoute

No inputs are necessary.

#### Route model routemodel.stayOnRoute.stayOnRoute

No inputs are necessary.

### Network model inputs

This model is responsible for keeping track of the locations of servers where video routers may be implemented.

#### Network model inputs networkmodel.simpleNetwork.simpleNetwork

"file" (compulsory) a csv file in which each line is a 4-tuple, the lat, long of a server, the cost of putting 1GB of traffic from the Internet to that server and the cost of sending 1GB of traffic from that server to the Internet.

### Server model inputs

The server model chooses which data centres from the network model are used to serve a given set of streams.

#### Server model servermodel.nRandom.nRandom

This model chooses *n* datacentres at random to serve traffic. As soon as a session starts n of the servers are chosen at random.

"file" (compulsory) defines the number of servers.

#### Server model servermodel.nStatic.nStatic

This model chooses *n* datacentres to serve traffic. They are chosen to be the best n servers to meet demand which has the average profile of traffic.

"file" (compulsory) defines the number of servers.

#### Server model servermodel.nDynamic.nDynamic

This model chooses *n* datacentres to serve traffic using a clustering model based upon the users present

"file" (compulsory) defines the number of servers.

"update\_time" (optional) defines the time in seconds between which servers should be chosen. E.g. if it is set to 60 then new servers will be selected at most once per minute.

### QoE models

The QoE model section gives the models which return proxy measures for the user experience. A route is specified as *(u1,d1,...,dn,u2)* where *ui* are the locations of the two users and *di* are the locations of the data centres (there may be only one of these). The route is broken down into hops and the delay on each hop is calculated according to a formula from the lat and long of that location. The delay over managed network (between datacentres) and unmanaged network (between datacentre and user) is tracked separately.

#### Haversine model qoemodel.haversine.haversine

This model uses only the haversine distance to get delay. It takes no parameters.

#### Landa model qoemodel.landa.landa

This model uses published work about how subcontinental zones affect delay. The work is described in [NETWORKING]. The model takes no parameters.

### Output models

The output models are designed to send output to a file or to the screen (if a file is not specified). All have the option "file" (optional). If specified output is to file, if not, output is to screen. Most output modules produce output with a comment file as the first line. Note that several output models are specified hence the JSON file requires square brackets and then curly brackets even if only one output model is used.

### *outputmodel.basicDaily.basicDaily*

Simply prints the number of arrivals in a given day. Note in the example file this one is used twice, deliberately, once to screen and once to file.

***outputmodel.outputCost.outputCost***

This provides output showing the costs experienced every day. The costs are in rows for each day with each row containing:

cost network->server, cost server->server, cost server->network, cost CPU

***outputmodel.shiftStats.shiftStats***

Options: "time": "daily"|"end"

This provides output showing how users shift within sessions. Each row shows the day number, the number of sessions in total within that period (including both new sessions and sessions which existed at the start of the period) and how many times that the choice of servers for the session changed. (Choosing the same servers in a different order does not count as a change).

***outputmodel.qosPdf.qosPdf***

Options: "style": "basic"|"quintile"|"decile"|"pdf"

"time": "daily"|"end"

This provides output about the qos experience of users. There is one row per day if daily is selected or just one for the whole simulation period if end is selected. The row contains the mean delay on unmanaged network, the standard deviation of the delay on manged network, the same two quantities for unmanaged network. The next lines depend on the option selected.

1. For basic no further output is given
2. For quintile, the output continues with the quintiles for managed then unmanaged network.
3. For decile the same but with deciles not quintiles
4. For pdf the whole PDF (with a given granularity) for managed then unmanaged network is printed on each line.