Research project: **Design and Analysis Tool for Queuing Models**
Keywords: CAD, Javascript, modelling, simulations, queueing theory
Supervisor: Prof. Esa Hyytiä (esa@hi.is)

Background: Many software packages are nowadays available as "web-applications" (cf. Overleaf, MS Word, Google's tools). The main benefit is that anyone can immediately use them without installations. Moreover, the files are typically stored in cloud and thus accessible from anywhere (in the world).

The field of queueing theory studies *server systems* such as a pool of web-servers or a router in a network. The basic system consists of a queue (buffer) and a processor. A more complicated models involve several processors (cf. multicore CPUs and computing clusters). A complete model defines also a job arrival process and service order (e.g., the first-come-first-served, FCFS), and performance metrics.

Some queueing models are amenable for analytical efforts, often yielding closed form mathematical expressions for important performance metrics (such as the mean response time). How, some models are beyond analytical means and often evaluated by means of simulations.

Project Description:

You can find an elementary simulator written in Javascript at http://hi.is/∼esa/dispatching/rr

The general idea is to implement a design and analysis tool (cf. CAD) for such networks. The project consists of three phases as detailed below.

I. GUI: Design and build a working GUI:

- User can insert and move around different queueing elements on the drawing board:
  1. Sources of jobs (e.g., define a pattern for web-server requests)
  2. Job routing policies (e.g. Round-robin)
  3. Servers with different scheduling disciplines (FCFS, LCFS, SRPT, . . .)
- Modify parameters (e.g., the processing speed), and connect elements in arbitrary fashion with mouse
- User must also be able to load/save designs

II. Simulations: Once a functional GUI works, the next phase is to facilitate simulations:

- When prototyping and demonstrating, it is good to be able to simulate the system in browser (as in the web page above).
- For accurate results, one often needs a faster simulator. This could be implemented either in Python, C or C++. The simulator would read the model description first, and then carry out simulations are specified.

III. AI and analytical results: (optional) the system would be equipped with analytical capabilities:

- Possibility to analyze a model and try to map it to some known system for which closed-form results exist. This is obviously challenging, but some simple reductions could be demonstrated at minimum.

Prerequisites::

- Some knowledge on web-programming and ambition for elegant clean code
- Basic understanding of the probability theory and simulations
- REI503M *Performance Analysis of Computer Systems* (recommended, lectured this fall)