OMNeT++ Projects

Traffic Theory

Ilario Filippini

General Rules

- Projects are not mandatory, one student can decide not to take any project
- Projects must be submitted **by the last exam session of the Academic Year**. Projects submitted by **July 12th 2020** will receive **100%** of the **points, after** this date only **50%** of **points** will be given
- Projects must be developed in OMNeT++
- Personal projects can be proposed, the acceptance and the number of assigned points are at the instructor's discretion

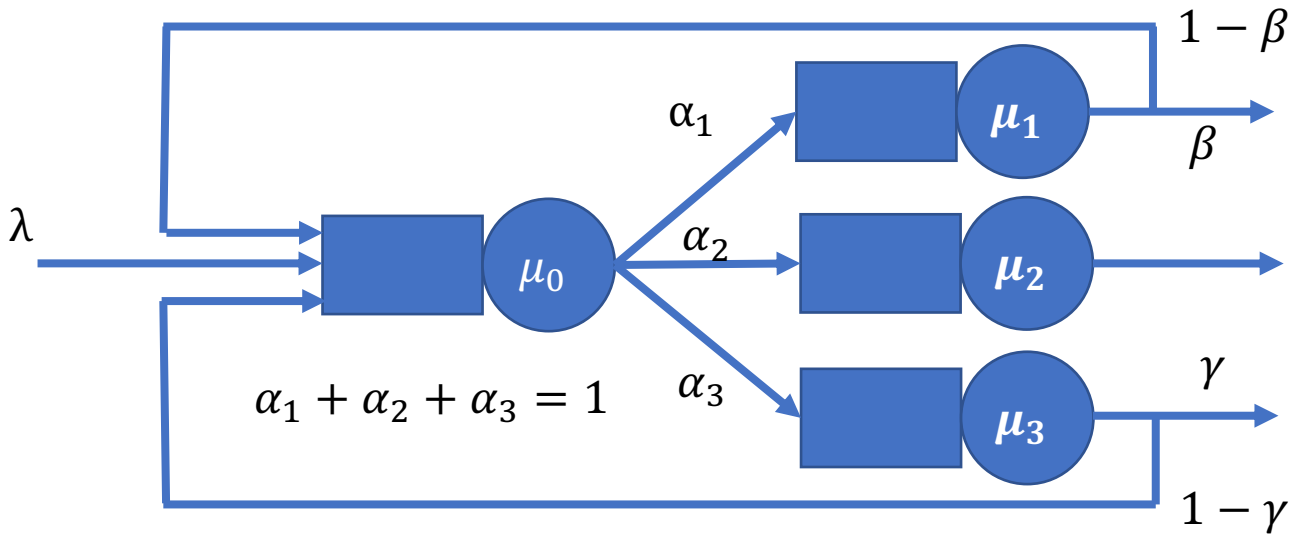
Project Registration

- The students willing to take the project assignment must choose among the project proposals using the **online form** available at https://goo.gl/forms/2ww0Hf1IIUZchdxQ2 **by Jan 31st, 2020**.
- **IMPORTANT 1:** ONLY THE PROJECTS REGISTERED ON THE ONLINE FORM WILL BE CONSIDERED. LATE REGISTRATION WILL NOT BE ACCEPTED.
- **IMPORTANT 2:** REGISTRATION IS NOT BINDING. IF YOU REGISTER FOR A PROJECT AND THEN DECIDE AFTERWARDS YOU DON'T WANT TO DELIVER IT, THAT'S OK.
- For personal projects, write to the instructor describing the project you intend to implement. The deadline of **Jan 31st, 2020** still holds.

Project evaluation

- The following projects are divided in 3 groups
  - Projects **1.x** provide at most **1 point**
  - Projects **2.x** provide at most **2 points**
  - Projects **3.x** provides at most **3 points**
- Projects of classes **1.x** and **2.x** can be assigned to at most **1 student**, projects of class **3.x** can be assigned to groups of at most **2 students**
- In order to start the assessment of the project, you must **submit your OMNeT files via email** to the instructor. You'll be asked to set a **face-to-face meeting** to discuss your project
- Further instructions and clarifications on the project can be requested via email before the submission
- Projects will be evaluated based on the rationale of the design choices, the correctness and organization of the source code (It is not a Software Engineering course !!!)

Project 1.1



- Implementing the above network of M/M/1 queues
- All input parameters must be NED parameters
- Collecting statistics on
  - Average utilization of each server
  - Number of users in each service center over time
  - Average number of times a user enters each service center before leaving
  - Overall network delay
- Hint: use message fields to carry information

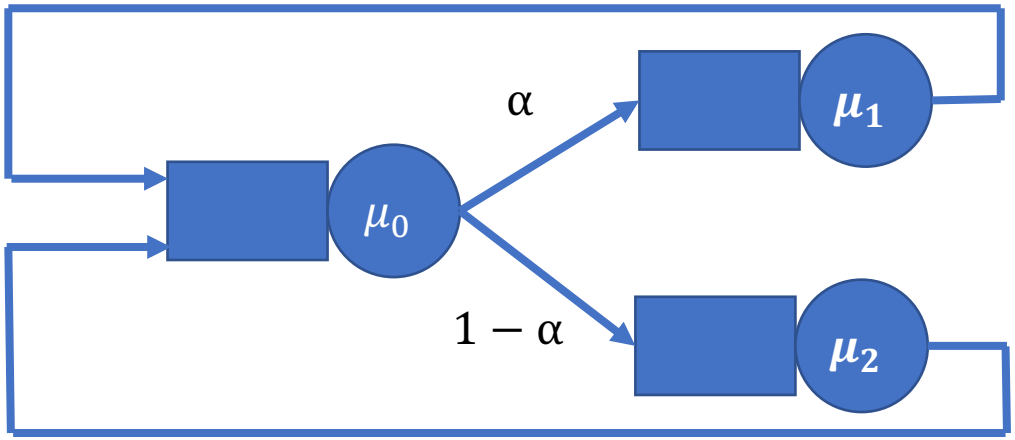
Project 1.2

- Implementing a M/M/2 queueing system with a SIRO scheduling policy
- Collecting statistics on:
  - Average queueing and response time
  - Average utilization of each server
  - Queue length over time
  - Average queue length

Project 1.3

- Implementing a M/M/∞ queueing system
- Collecting statistics on:
  - Average response time
  - Number of used servers over time
  - Average number of used servers

Project 2.1



- Implementing the above closed network of M/M/1 queues
- The network has *N* users and when it starts all but one users are in the queue of center 0, one user starts his service at the server of center 0
- All input parameters must be NED parameters
- Collecting statistics on
  - Average utilization of each server
  - Number of users in each service center over time
  - Average in and out rate at each server
  - Average waiting and response time at each server

Project 2.2

- Implementing a M/M/1 system with a SPTF discipline
- Collecting statistics on:
  - Average queueing and response time
  - Average utilization of the server
  - Queue length over time
  - Average queue length

Project 2.3

- Implementing a Processor Sharing System with a limit *k* on the number of simultaneously served users
- All settings must be available as NED parameters
- Collecting statistics on:
  - Average queueing and response time
  - Queue length over time
  - Average queue length
  - Number of used servers over time
  - Average number of used servers

Project 3.1

- Implementing an *n*-class round-robin scheduler for a M/M/1 queue
- Each class is characterized by different arrival rate and service time
- The scheduler must be fair (all classes have the same amount of time *t* at each turn) and the *t* must be an input parameter
- Note that if residual work *w* for a user is less than *t*, the turn for that user lasts *w*
- All settings must be available as NED parameters
- Collecting statistics on:
  - Per-class and generic average queueing and response time
  - Queue length over time
  - Average queue length
  - Average utilization of the server

Project 3.2

- Implementing an *n*-class priority scheduling for a M/M/1 queue with and without preemption
- Each class is characterized by different arrival rate and service time
- All settings must be available as NED parameters
- Collecting statistics on
  - Per-class and generic average queueing and response time
  - Per-class extended average service time
  - Queue length over time
  - Average queue length
  - Average utilization of the server