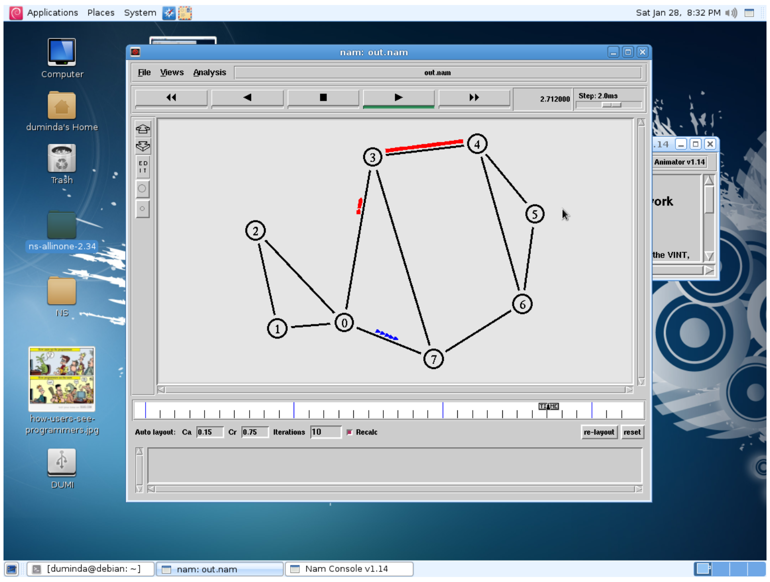


Project Plan Network Simulator

Arponen Jani   Ogenda Dancun
Ruley Brian   Varis Leo

17.07.2020

Aalto University

ELEC-A7151 - Object oriented programming with C++ – Summer 2020

1 Information page

Students

Arponen Jani
Ogenda Dancun
Ruley Brian
Varis Leo

Official Instructor

Sarolahti Pasi

Changelog

Version	Date	Author	Description
0.1	2020-07-08	All	Template

Table 1: Document changelog.

Contents

2 Introduction

The purpose of this document is to outline the project plan for the project portion of the *ELEC-A7151 - Object oriented programming with C++* course. The project topic is to build a network simulator which can model nodes, links and traffic. Each node can be an end-host or a router and they are interconnected using links, which carry packets of data over them.

3 Scope

The scope of the project is defined by the functionality it must implement based on the project description given by the course. The functionality is split roughly into modules and individual system and functional requirements within these modules. The requirements are presented below in Table 2. The numbering of requirements follows the logic: `(A|B|S).(#)[.(#)...(#)]`, where the first character describes the type of requirement: Advanced, Basic, System; and the subsequent numbers are the ID or optional sub-ID.

Module	**Req#**	**Requirement**
Compatibility	S1	It shall be possible to compile and run the program on Ubuntu 18.04.
Network model	B1	The network shall be modeled by nodes and links between nodes. Communication between nodes shall be done by (data) packets over the links.
	B1.1	Links shall be defined by a transmission speed and a propagation delay, which shall govern: 1. how fast new packets can be sent; and 2. how fast they propagate over the link. There shall be a way to queue packets at the node before the link.
	B1.2	Nodes shall be defined by an address and are of a type: router or end-host.
	B1.2.1	Routers shall be able to route packets between other nodes.
	B1.2.2	End-hosts shall be able to run applications that can send and/or receive packets to/from other end-hosts for a specified length of time.
	B2	The model code shall be written in such a way that it is easy to extend with e.g. new kinds of links or applications.
Program	B3	Running simulations shall be "easily configurable" for different network scenarios, through e.g. configuration files.
	B4	It shall be possible to collect statistics on the simulated network, e.g. packet to destination times, link utilization, queue lengths, etc.
	B5	From the applications user interface, it shall be possible to follow the progress of simulation, including statistics and states for links, queues and packets.
GUI	A1	There shall be a graphical user interface (GUI) for the program to interact with all other functionalities.
	A1.1	B5 shall be expanded to an animation on the GUI.
Expanded functionality	A2	B1.1 shall be expanded to create different queue behaviours, including limited queues and as a result, dropped packets.
	A3	B1.1 and B1.2 shall be expanded to include mobile hosts, i.e. wireless links.
	A3.1	Communication parameters of wireless links shall be defined by signal strength.
	A3.2	Mobile hosts shall be able to move around in a 2D map with obstacles that reduce the signal strength of the wireless link.

Table 2: Program functional requirements dissected from the project description.

There are no specific requirements imposed on the applications (B1.2.2) and what they should do, rather, their data and functionality is arbitrary. In order to keep the applications small yet still relevant, below are some examples of applications whose functionality could be implemented or emulated in the case it's part of a larger, real protocol or technology (e.g. ping and tracert – implementing all of ICMP is out of the scope of this project). Examples of such applications are listed in the below Table 3.

Application	**Description**
TCP	Not necessarily an application, but a protocol on top of which other applications can be built. E.g. if A3 is implemented, robust data transfer requires acknowledging packet receival in situations where packet loss is possible.
Ping	Request a reply from the destination address. Can be used to time round trip times.
Traceroute	Requests a reply from the destination address using sequentially increasing time-to-live parameter on sent packets. Can be used to trace the path the packets took to the destination.
"File" transfer	Send a file to, or request a file from the destination. The file can be any data split into multiple packets, e.g. randomly generated string or number sequence.
DHCP	In case a new node is added to the network and connected to a router, yet the node doesn't have an assigned address, the router should be able to provide one for the node. This is an example of an application running on the routers instead of end-hosts.

Table 3: Examples of applications that could be implemented for the network simulator.

4 Structure

The program structure can be described using the UML diagram below in Figure 1. The network model contains links and nodes. End host and router expand the node class, which contains general data about them. End host can run applications, for example Ping. Routers are used to route packets between end hosts. Links are used to handle packet transfers between two different nodes.

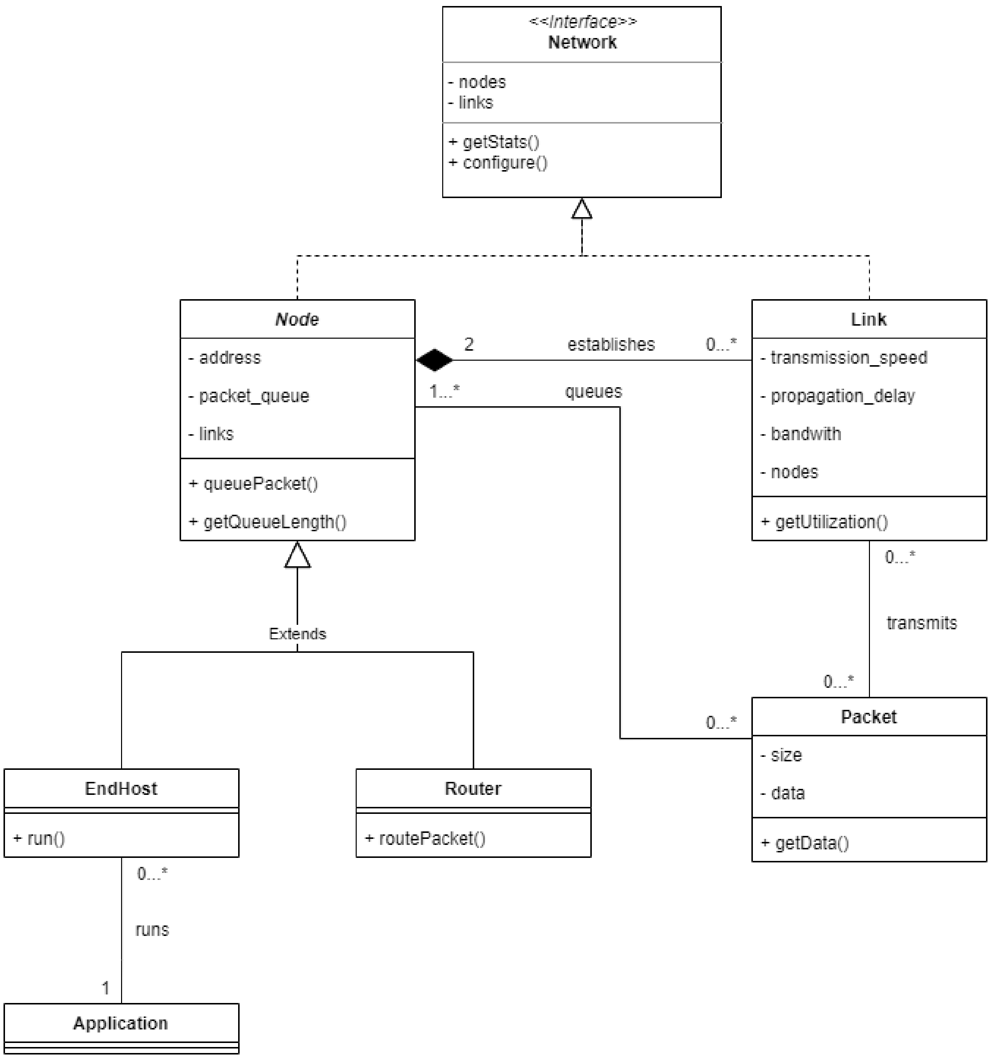


Figure 1: UML diagram of the basic components

The advanced functionality expands the UML chart with new elements, as seen in Figure 2 and describes how the program can be expanded to include mobile hosts and wireless links. With a 2D map, each node gains a location on a plane and additionally, mobile hosts have the ability to move around in the map. Wireless links are established between a mobile host and any other Node in reach. This link works much like it's base class, but with the addition of the signal strength attribute, which defines the bandwidth and success of packet transfer. Obstacles are objects located on the 2D map that affect the propagation of wireless signals by reducing the signal strength.

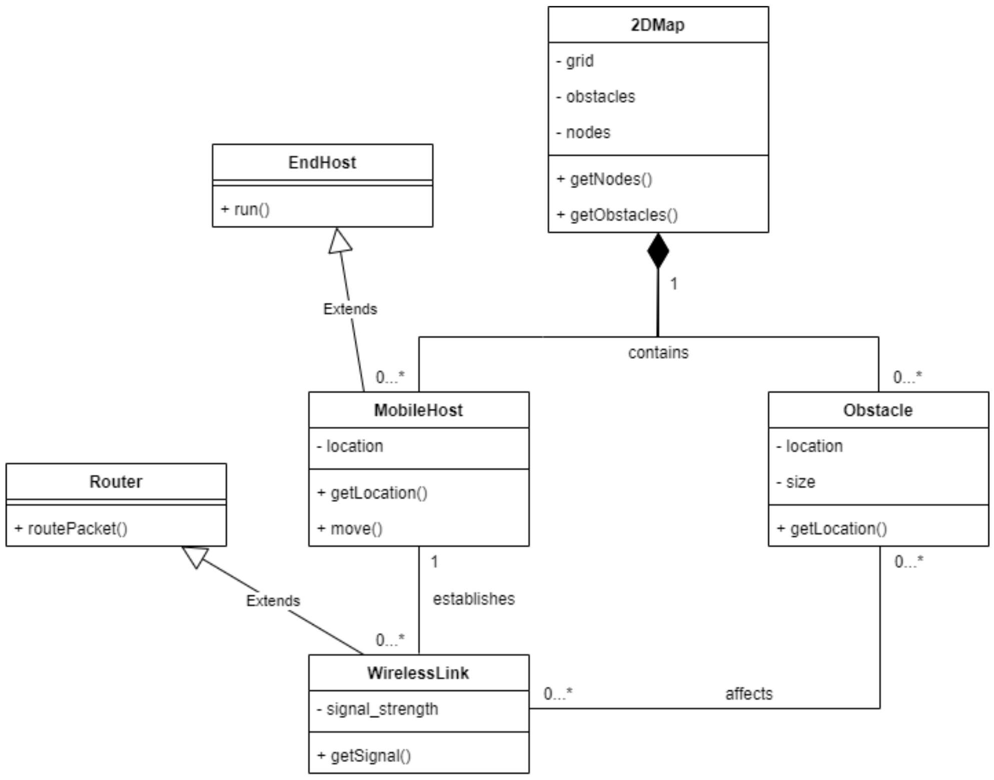


Figure 2: UML diagram of the expanded functionality

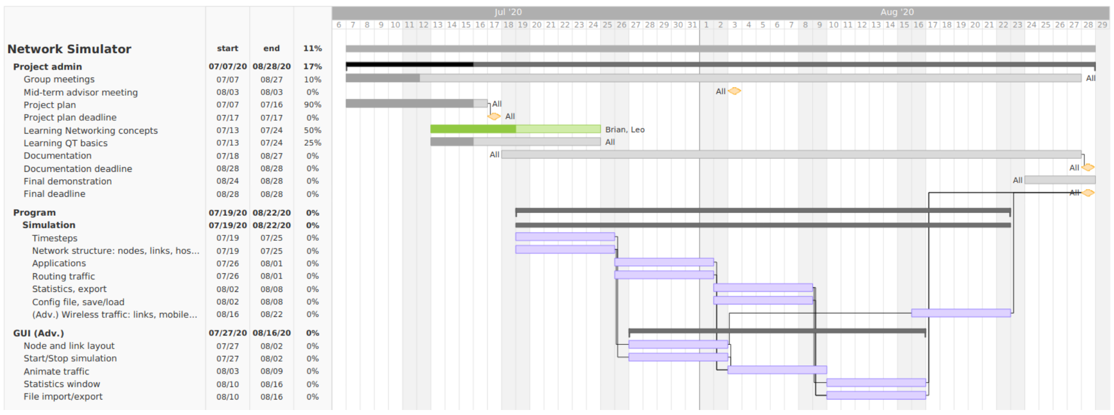
5 External libraries

The only external library needed for the project is Qt for GUI – as recommended in the project description. Some other needs for external libraries may rise during the project and will be considered when necessary. Examples of additional functionality that may require external libraries include using specific filetype(s) for configuration and/or statistics data files, such as JSON.

We will update the external libraries we might use in case of need. So far we believe that since we are developing a GUI application, that Qt is enough for the task.

6 Schedule

The initial scheduling can be seen in Figure 3, note that the tasks have not been distributed among the team members yet. The schedule is broken into rather simple features or tasks and will no doubt become more fleshed out as we progress in the project further. Each of the tasks consists of planning, programming and testing of the code. The major dependencies are outlined as links between tasks, e.g. the network structure must be built before applications or traffic routing can be built.



Network Simulator	**start**	**end**	**11%**
Project admin	**07/07/20**	**08/28/20**	**17%**
Group meetings	07/07	08/27	10%
Mid-term advisor meeting	08/03	08/03	0%
Project plan	07/07	07/16	90%
Project plan deadline	07/17	07/17	0%
Learning Networking concepts	07/13	07/24	50%
Learning QT basics	07/13	07/24	25%
Documentation	07/18	08/27	0%
Documentation deadline	08/28	08/28	0%
Final demonstration	08/24	08/28	0%
Final deadline	08/28	08/28	0%
Program	**07/19/20**	**08/22/20**	**0%**
Simulation	**07/19/20**	**08/22/20**	**0%**
Timesteps	07/19	07/25	0%
Network structure: nodes, links, hos...	07/19	07/25	0%
Applications	07/26	08/01	0%
Routing traffic	07/26	08/01	0%
Statistics, export	08/02	08/08	0%
Config file, save/load	08/02	08/08	0%
(Adv.) Wireless traffic: links, mobile...	08/16	08/22	0%
GUI (Adv.)	**07/27/20**	**08/16/20**	**0%**
Node and link layout	07/27	08/02	0%
Start/Stop simulation	07/27	08/02	0%
Animate traffic	08/03	08/09	0%
Statistics window	08/10	08/16	0%
File import/export	08/10	08/16	0%

Figure 3: Gantt chart of the project schedule.

7 Communication

The project team has settled on using Microsoft Teams as the communication tool for the project. Weekly progress meetings will be held and the chat used to communicate outside of the meetings. Major tasks and features will be tracked using the built in Tasks functionality in Teams. Bugs and incidents will still be handled through Git, if it is deemed appropriate.