Performance Analysis and Simulation of Communication Systems: Project

May 18, 2021

# Project A

## Part I: PRNG and Random Variables

1. Create a function that implements a linear congruential generator (LCG), accepting as input the parameters: *seed*, *m*, a and *c*.

Hint: It is better if you do not attempt to modify the rng module of ns-3 instead create a function in your simulation file (e.g. mysimulation.cc) and call the function from the main.

1. Generate 1000 values uniformly distributed in the range [0,1] using your PRNG. For this case use *m*=100, *a*=13 c=1 and seed =1;
2. Compare the distribution of your values with the distribution of values generated using the

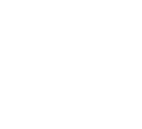
UniformRandomVariable() of ns-3.

1. Comment on the difference in the results and propose values of *m*, *a* and *c* which gives you better results.
2. What PRNG does ns-3 use? What method does ns-3 use to generate a normal random variable?
3. Using the time system command of linux compare the execution time for the generation of the uniform distribution using your function and ns-3 function.
4. Write a second function that generates an exponential distribution with mean 𝛽 > 0 from a uniform distribution generated using the LCG; Choose one of the methods for generating RV covered in the course and motivate your choice with respect to the specific task.
5. Compare your exponential distribution with ns-3 ExponentialRandomVariable()and the theoretical expression of the probability density function.

## Part II: Mathematical Modelling of a system of Queues

The system we model represents a typical network infrastructure interconnecting a subnet with the Internet. In the scenario, shown in figure 1, there are four sources of traffic, named *A*, *B*, *C* and *D*.

**A** *5Mbps*



*Server*

e

*5Mbps*

*10Mbps*

*Router*

g

*8Mbps*

Internet

f

*8Mbps*

*5Mbps*

*5Mbps*

**D**

**C**



 *5Mbps*



**B**

*Figure 1: Scenario for Project A*

Assume that each of the four devices (*A*, *B*, *C* and *D*) generates request packets with a Poisson distribution (i.e. an exponential distributed time between packets). The size of each packet is also exponentially distributed. See **Table I** for the exact values.

*Table 1 Traffic parameter for Project A*

|  |  |  |
| --- | --- | --- |
| Source | Mean packet interarrival  time | Mean packet size |
| A | 2 ms | 100 Bytes |
| B | 2 ms | 100 Bytes |
| C | 0.5 ms | 100 Bytes |
| D | 1 ms | 100 Bytes |

The server with probability *p=0.7* will immediately reply to a received packet with a response packet addressed to the source of the request packet.

With probability *1-p* the server will forward the request through the router to the internet.

All the communication links are full duplex and Point-to-Point, you can assume the signal propagation delay to be negligible.

Before transmission each packet is buffered in a queue, each device has a separate queue for each P2P link. For example, device *f* has three queues one for the packets going to *B*, one for the packet going to *C* and one for the packet going to *g*.

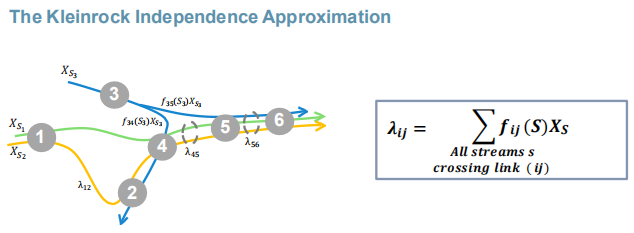
Using queuing theory and the network of queues theory answer the following questions:

1. What is the average number of request packets in the system that are traveling to the server?
2. What is the average number of request packets in each link that are traveling to the server?

**Solutions**

A duplex communication system is a point-to-point system composed of two or more connected parties or devices that can communicate with one another in both direction. In a full-duplex system, both parties can communication with each other simultaneously. There is a two-way communication channel between them, or more strictly speaking, there are two communication channels between them.

1. According to the Slow Truck Effect, on a unidirectional link, the arrival process that obeys the Poisson distribution at the beginning will no longer obey the Poisson distribution after a few points. The reason is that long data packets require longer processing time than short data packets. If things go on like this, the distance between data packets will get closer and closer. For a complex network with intersecting links, the departure process after the data packets are converged is very complicated. Therefore, it is reasonable to apply Kleirock independence approximation to simplify the network process. The result of the application is that at the intersection of different links, for each link where the arrival flow is Poisson distributed, the arrival rate of the arrival flow is directly added to form the arrival flow of the new M/M/1 system.



*figure.2 The Kleinrock Independence Approximation*

Based on Kleinrock Independence Approximation, we know how to calculate the for ,,.

*Table2. Arrival rateof each link for Project A*

|  |  |  |
| --- | --- | --- |
|  | Details | Arrival rate(Packets/s) |
|  | 1/2ms | 500 |
|  | 1/2ms | 500 |
|  | 1/0.5ms | 2000 |
|  | 1/1ms | 1000 |
|  | = | 500 |
|  | =+ | 2500 |
|  | =++ | 4000 |

1. Bandwidth describes the maximum data transfer rate of a network or Internet connection. It measures how much data can be sent over a specific connection in a given amount of time. Therefore, given bandwidth, it is easy to calculate service rate because service rate measures how many packets can be sent over a specifica connection in a second.

Expressing bandwidth

K = kilo = 1000 bits

M = mega = 1000 kilo = 1,000,000 bits

G = giga = 1000 mega = 1,000,000,000 bits

T = tera =1000 giga = 1,000,000,000,000 bits

*Table3. Service rate of each link for Project A*

|  |  |  |
| --- | --- | --- |
|  | Calculation Details | Service rate(packets/s) |
|  | 5Mbps/100\*8bits | 6250 |
|  | 5Mbps/100\*8bits | 6250 |
|  | 5Mbps/100\*8bits | 6250 |
|  | 5Mbps/100\*8bits | 6250 |
|  | 5Mbps/100\*8bits | 6250 |
|  | 8Mbps/100\*8bits | 10000 |
|  | 10Mbps/100\*8bits | 12500 |



*Table4. the average number of queueing packets of each link for Project A*

|  |  |  |
| --- | --- | --- |
| Queue Packets in each link | Claculation Details | Avg. Queueing packets of a link |
|  |  | 0.08695 |
|  |  | 0.08695 |
|  |  | 0.4706 |
|  |  | 0.1905 |
|  |  | 0.08695 |
|  |  | 0.33333 |
|  |  | 0.4706\*4000/12500=0.15 |

Finally, back to that two question requested in the project:

1. What is the average number of request packets in the system that are traveling to the server?

=0.08695+0.08695+0.4706+0.1905+0.08695+0.33333+0.4706=1.72585 (requested Packets)

1. What is the average number of request packets in each link that are traveling to the server?

See Table 4

## Part III: Simulations and Results Comparison

Implement the scenario in ns-3. Use the P2P communication for all the links. For the generation of the traffic you can use the UDP Client and Server Application. The Server application will need to be modified so that it reply only to a fraction *p=0.7* of the messages it receives while the rest is forwarded to the router. You can assume that the router will simply drop all the received messages without taking any action.

* Run the simulation and measure:
  + The average number of request packets queued at the link between *g* and the

*Router* (*g* → *Server* )

* + The average queuing delay and the total average delay for the request packets traversing the link f →*g*
* Compare the results you get from the simulation with the one you obtained in Part II.
* Measure the delay between *A* and the *Server* and between the *Server* and *A*.
* Replace each P2P link with a bus using CSMA/CD do not change the datarate or delay.
* Run the simulation and compare the results with the one obtained with the P2P. Try to explain the difference you see.

Carrier-sense multiple access(CSMA) is a media access control(MAC) protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus or a band of the electromagnetic spectrum. A transmitter tries to detect the presence of a carrier signal from another node before attempting to transmit. If a carrier is sensed, the node waits for the transmission in progress to end before initiating its own transmission. Using CSMA, multiple nodes may , in turn, send and receive on the same medium. Transmissions by one node are generally received by all other nodes connected to the medium.

P2P link refer to a wire or other connection that links only two computers or circuits, as opposed to other [network topologies](https://en.wikipedia.org/wiki/Network_topologies" \o "Network topologies) such as [buses](https://en.wikipedia.org/wiki/Computer_bus" \o "Computer bus) or [crossbar switches](https://en.wikipedia.org/wiki/Crossbar_switch" \o "Crossbar switch) which can connect many communications devices.

The difference we saw is that there are many more lost packets in CSMA/CD link than that in the P2P link when the parameters are the same. This is because when the mean inter-transmission time is too small, there will be a lot of packets in the medium while the medium only allows one packet transmitted one time. Therefore, the packets with collision will be back off( or lost) without retransmission. This has been verified that the lost packets reduce to 0 when we increase the mean inter-transmission time to 1 for node A, B, C, and D to 1 second in CSMA/CD link.

* Run the simulation but this time use the custom PRNG that you implemented in Part I to generate the exponential packet size and the exponential time between packets. Which differences do you observe?

Seed:

When simulating any random numbers it is essential to set the random number seed. Setting the random number seed with set.seed() ensures reproducibility of the sequence of random numbers.

By setting a seed for random variable generator to tell it to start from a fixed seed instead of starting from a seed taken randomly. That will ensure that for the same data, it will output the same result(e.g. [3 84 12 21 43 6]) when someone reruns the codes.

1. What seeds you used
2. What RNG you used if not the default
3. How were independent runs performed,
4. For large simulations, how did you check that you did not cycle.
5. Decide how you are going to manage independent replications, if applicable.
6. CObvince yourself that you are not drawing more random values than the cycle length, if you are running a very long simulation.

Queue:

PointToPointHelper.setQueue() Each point to point net device must have a queue to pass packets through. This method allows one to set the type of the queue that is automatically created when the device is created and attached to a node.

p2p\_1.SetQueue ("ns3::DropTailQueue", "MaxSize", StringValue ("1p"));

<https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_helper.html>

TrafficControlHelper

QueueDisc is an abstract base class providing the interface and implementing the operations common to all the queueing disciplines.

Child classes need to implement the methods used to enqueue a packet (DoEnqueue), dequeue a single packet (DoDequeue), get a copy of the next packet to extract (DoPeek), check whether the current configuration is correct (CheckConfig).

As in Linux, a queue disc may contain distinct elements:

* queues, which actually store the packets waiting for transmission
* classes, which allow to reserve a different treatment to different packets
* filters, which determine the queue or class which a packet is destined to

In ns-3 every queue disc including multiple queues or multiple classes needs an external filter to classify packets (this is to avoid having the traffic-control module depend on other modules such as internet).

The traffic control layer interacts with a queue disc in a simple manner: after requesting to enqueue a packet, the traffic control layer requests the qdisc to "run", i.e., to dequeue a set of packets, until a predefined number ("quota") of packets is dequeued or the netdevice stops the queue disc. A netdevice shall stop the queue disc when its transmission queue does not have room for another packet. Also, a netdevice shall wake the queue disc when it detects that there is room for another packet in its transmission queue, but the transmission queue is stopped. Waking a queue disc is equivalent to make it run.

Every queue disc collects statistics about the total number of packets/bytes received from the upper layers (in case of root queue disc) or from the parent queue disc (in case of child queue disc), enqueued, dequeued, requeued, dropped, dropped before enqueue, dropped after dequeue, queued in the queue disc and sent to the netdevice or to the parent queue disc. Note that packets that are dequeued may be requeued, i.e., retained by the traffic control infrastructure, if the netdevice is not ready to receive them. Requeued packets are not part of the queue disc. The following identities hold:

* dropped = dropped before enqueue + dropped after dequeue
* received = dropped before enqueue + enqueued
* queued = enqueued - dequeued
* sent = dequeued - dropped after dequeue (- 1 if there is a requeued packet)

TrafficCOntrolHelper.SetRootQueueDisc:This class can help to create [QueueDisc](https://www.nsnam.org/doxygen/classns3_1_1_queue_disc.html" \o "QueueDisc is an abstract base class providing the interface and implementing the operations common to...) objects and map them to the corresponding devices.