HALMSTAD UNIVERSITY

Lab 1

DT4015 - Data Communications

Fabian Henrysson & Haron Obaid

Group 10

Part 1: Getting Started with OMNET++ (20%)

In this part of the lab, you will setup and run a simple network simulation using OMNET++. For this purpose, you can use the **virtual machine** that you will find in this **link** (which requires you to have VirtualBox in your computer), or you can follow the instructions in the official website for OMNET++. For simplification, this document follows the virtual machine method.

After the installation, you will perform two main tasks: 1) setting up a simple network to exchange packets, and 2) collect and summarize statistics from the simulation. During the lab session, you will follow a simplified, step-by-step tutorial on how to perform these tasks. For extra information, or for an in-depth tutorial on this exercise, feel free to visit the official OMNET++ TicToc Tutorial in https://doc.omnetpp.org/omnetpp4/tictoc-tutorial/index.html.

Milestones

To be awarded the points for this part, you will show that you were able to perform the following tasks:

1. Running a 2-node TicToc (10%)
2. Statistics for Transmitted and Received Packets (10%)

Checkpoints 1

1. Running a 2-node TicToc

We get a new log each time the message is received by each one of submodules.
We can see a tictocMsg getting sent and received by the transceiver tic and toc. Where the msg is represented graphically by a red ball and that the message takes 100ms as specified in the tictoc1.ned file.

If we give the nodes asymmetrical delay times each one of the transmissions will take different times to complete. We first tried lowering one to 80 ms from 100 ms. Then later swapped it to 200 ms. The program requires you to remake the executable files after the changes.

Conclusion: You can remake the submodules with asymmetric times.

2. Statistics for transmitted and received packets

Each simulation was made to go for 30 simulated seconds.

Tic.out - Tic.in:

100 ms – 100 ms
Tic Sent: 151
Tic Received: 150
Toc Sent: 150
Toc Received: 150
Elapsed time: 0.001636 s

100ms – 200ms
Tic Sent: 101
Tic Received: 100
Toc Sent: 100
Toc Received: 100
Elapsed time: 0.000658 s

10 ms – 1000 ms
Tic Sent: 30
Tic Received: 29
Toc Sent: 30
Toc Received: 30
Elapsed time: 0.000539 s

150 ms - 453 ms
Tic Sent: 50
Tic Received: 49
Toc Sent: 50
Toc Received: 50
Elapsed time: 0.000953 s

1000 ms - 10 ms
Tic Sent: 30
Tic Received: 29
Toc Sent: 29
Toc Received: 29
Elapsed time: 0.000457

Results make sense compared to result data.

Conclusion: We picked different times for the module delay time both symmetrical and different asymmetrical version to simulate 30 seconds of time.

Part 2: Simulating Data, Acknowledgements, and Losses (20%)

In this part of the lab, you will model a network where "data" and "acknowledgements" are exchanged between nodes. Then, you will model a lossy channel where packets are lost with a certain probability. We will start from the network you set up in the first part, so make sure yours is up and running.

During the lab, you will perform two main tasks: 1) simulate processing times and schedule transmissions after a delay, and 2) setting up a network where packets are lost with a probability,

Milestones

To be awarded the points for this part, you will show that you were able to perform the following tasks:

1. Adding Randomness (10%)
2. Setting up a network with a "lossy" channel and show transmission/reception statistics (10%)

Checkpoints 2

1. Randomness, Inter-generation Gaps, Inter-packet Gaps.

Towards accomplishing this goal, we followed the code set out by the instructors and we collected a sample from the network with a Tic delay at 1 s – 0.1 s, 2 s – 5 s, 0.75 s – 0.91 s. These were all simulated on 100s simulated time. The IGG and IPG come in the order of first reception.

1 - 0,1 (s)	Tic out	Toc in	Toc out	Tic in
IGG/IPG	1,33872951	1,33872951	1,327094791	1,334683321
STDEV	1,038503977	1,02495771	1,018254606	1,031530686

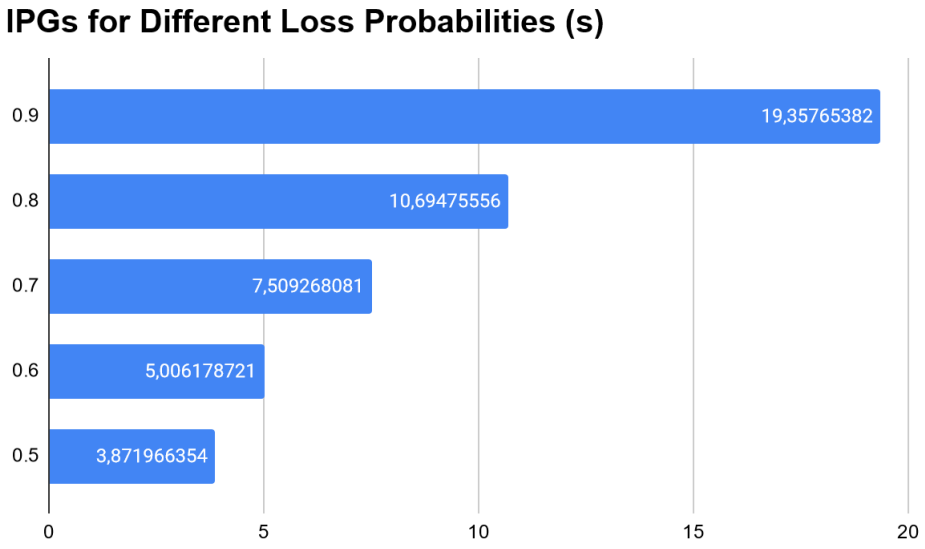
2 – 5 (s)	Tic out	Toc in	Toc out	Tic in
IGG/IPG	2,539606954	2,539606954	2,538498837	2,538498837
STDEV	1,844809411	1,844809411	1,866116417	1,866116417

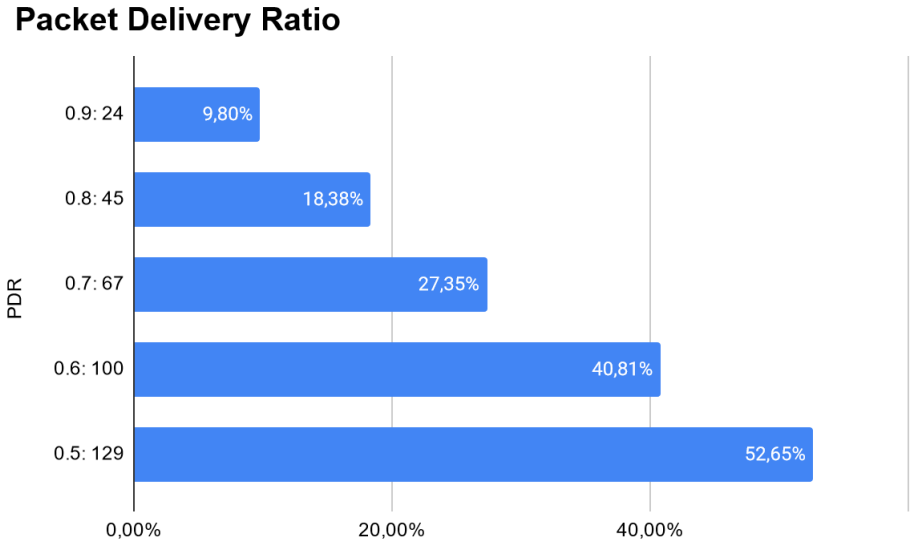
0,75 - 0,91 (s)	Tic out	Toc in	Toc out	Tic in
IGG/IPG	0,9821011124	0,9979457999	0,9971981041	0,9971981041
STDEV	0,7173812199	0,7173812199	0,7227248909	0,7227248909

Conclusion: The IGG/IPG & the Standard Deviation are most impacted by the delay times inputed and could be seen from our data the IGG/IPG are for the most part the same and only minor differences are noticeable for the same delay times.

2. Adding probability of losses and effects on IPG and Packet-delivery Ratio

We made a simulation for the different loss probabilities 0.9, 0.8, 0.7, 0.6 and 0.5. These all ran for a simulated 100 seconds where Tic had a delay of 2s and Toc on 0 s as it will not be transmitting. Giving us a total of 245 packages per run. Out of those the different plots are extracted on the basis of the different IPGs and PDRs.





Conclusion: The different plots have an inverse relation which makes sense. A larger probability of loss would give you a higher time per successful package.

Part 3: Automated Repeat reQuest and Packet Transmission Delay (60%)

In this part, you will 1) implement Stop-and-Wait Automated Repeat reQuest (ARQ) between two nodes in a lossy data channel and a perfect feedback channel, and 2) measure in a simulation if the delay analysis from the lectures holds in an experimental setup.

Milestones

To be awarded the points for this part, you will show that you were able to perform the following tasks:

1. Implement Stop-and-Wait ARQ (30%)
2. Simulation of Transmission Delay with Retransmissions (30%)

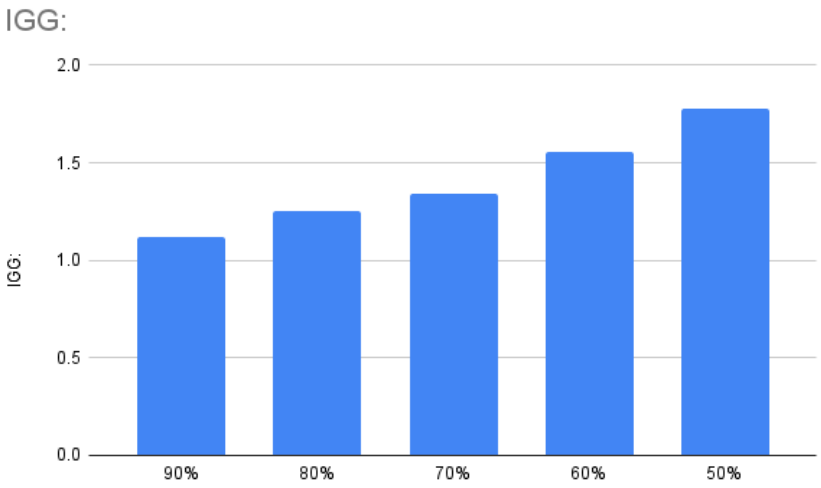
Checkpoints 3

1. IGGs, IPGs, and PDR for ARQ.

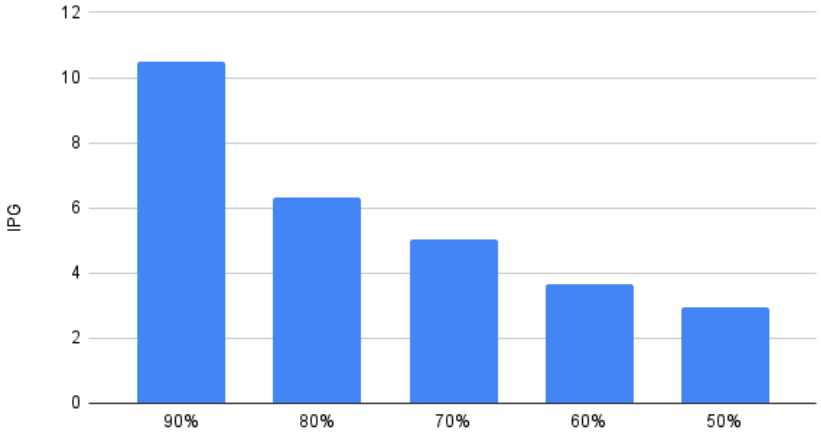
We started by implementing the ARQ by adding the given code snippets to our Part:2 program and changing the toc-receiver behavior as described in the assignment. Where it will resend the message if the message is not received. The loss implementation is made using a random uniform between 0-1 where the message will be lost if the random number is less than the loss probability.

The fixed transmission rate is set at every 2 seconds, or 0.5 Hz and the random time is set with a random exponential with an average of 2 seconds as described in the assignment. We also increased the simulation time to 200 seconds for both checkpoints in part 3.

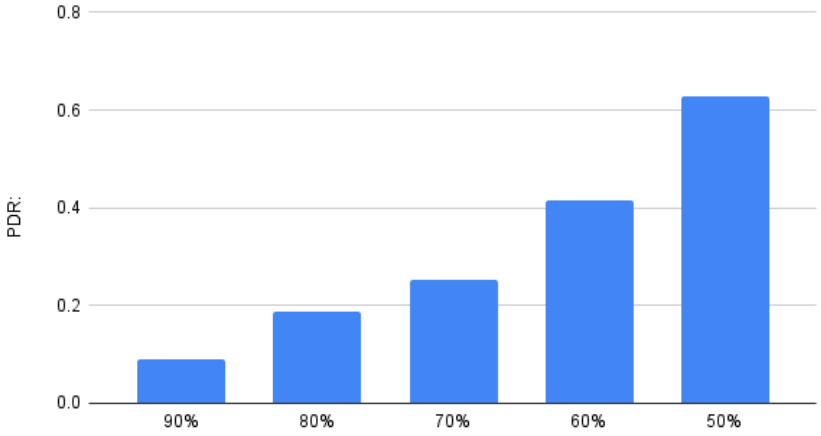
Fixed transmission interval results:



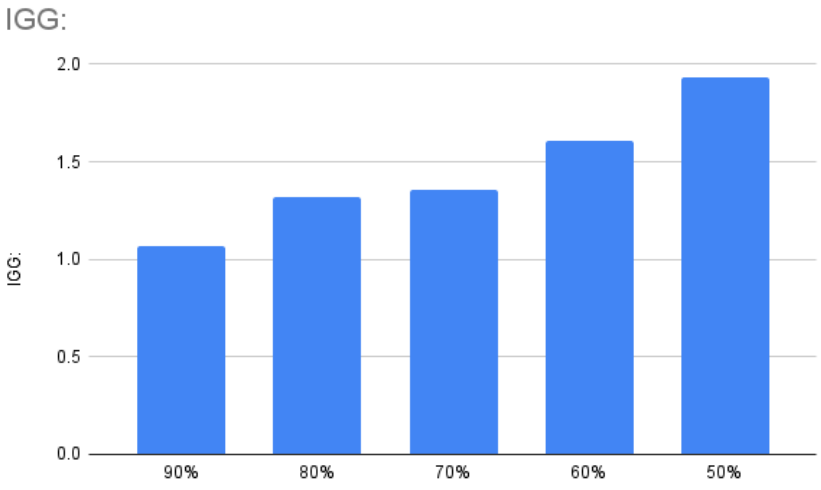
IPG:

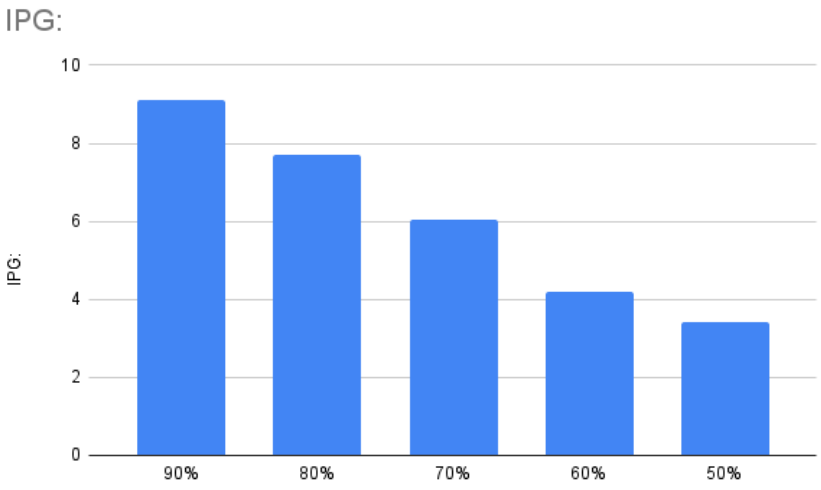


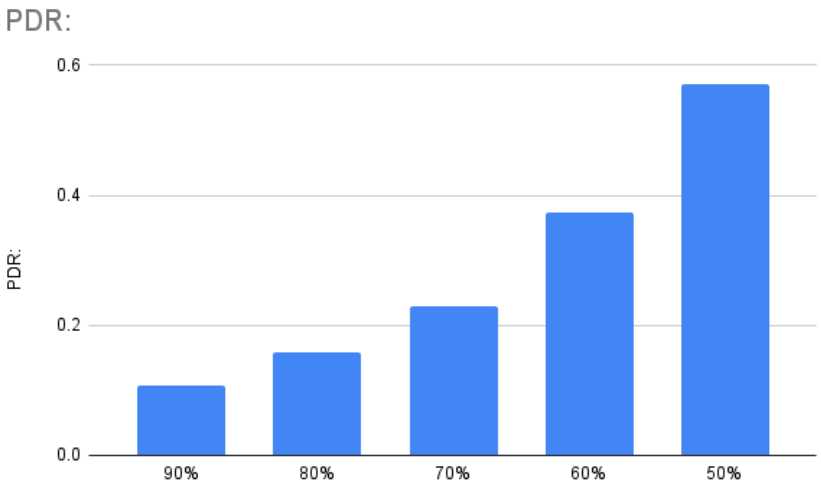
PDR:



Random transmission interval results:





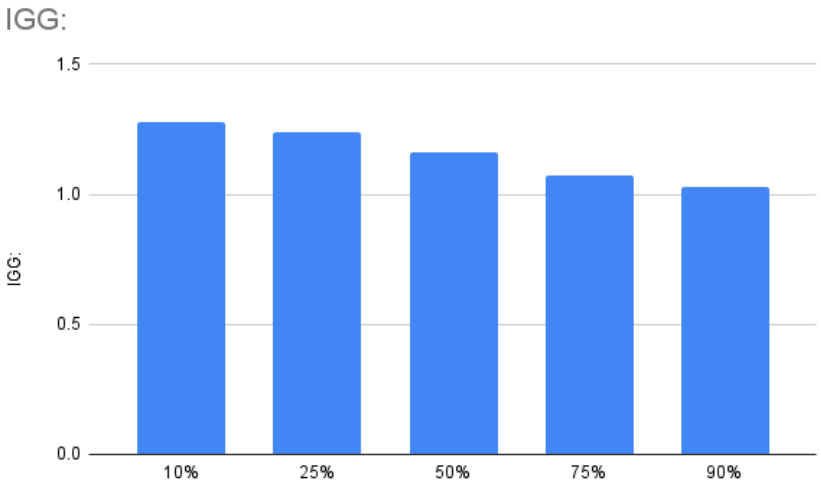


Conclusion:

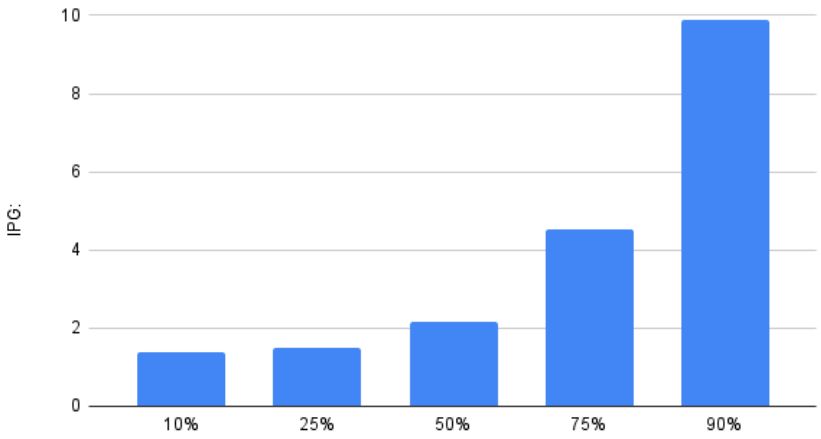
We can see the difference/similarities randomness has in a lossy receiver. We can also observe that the ARQ is working in the graphs. The IGGs show that the closer we get to loss ratio goes to 100% we go toward 1. Which makes sense as we transmit every second. We can see that the IPGs corelate to the probability exponentially as the probability of loss increase. In the PDR graph you can see that the transmissions are lost at a rate close to the set loss probabilities.

2. Statistics (average, standard deviation) for transmission delay (i.e., number of attempts)

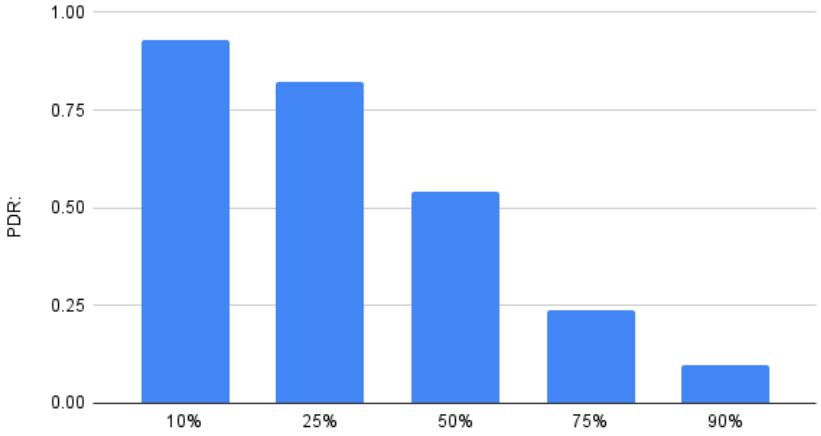
We changed the transmission scheduling according to the specification (1 Hz). From that we got the IGG and IPG as shown below. After that we wrote a python script to read .csv files of the results to get the standard deviation of the number of attempts.



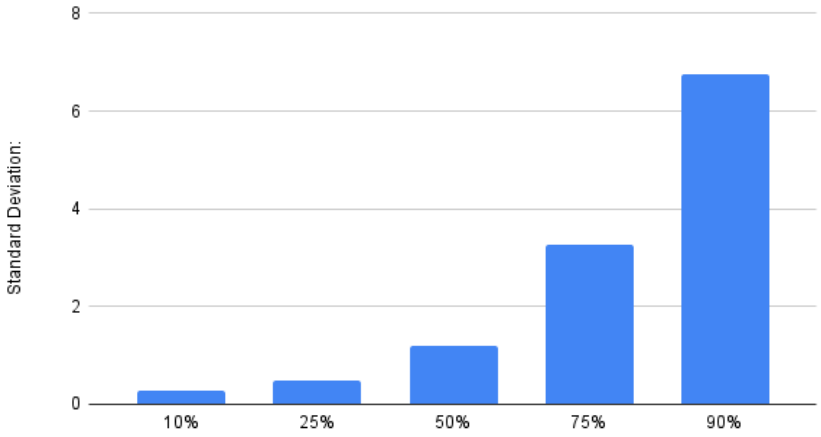
IPG:



PDR:



Standard Deviation:



Conclusion:

From the standard deviation graph, we can visually see that it closely resembles the inverse to the probability of a successful packet delivery as it grows in a exponential manner.

Bonus Exercise: Full Scale Simulation (up to 4 bonus points)

If you are reading this, congratulations. It means you were able to complete successfully the 3 parts of this lab and can now perform a full-scale simulation. The intention of this final part is for you to exhibit your command of statistics and simulations.

The task at hand is quite simple. You will rely on your simulation setup to perform a very similar experiment and obtain reliable results. For that, your simulation will use the repetition feature in OMNET++, where random values change between runs. If you run your simulation twice, you will notice that all the values are the exact same, so the repetition feature in OMNET++ helps you by changing the seed in the random-number generator. This way, two experiments will get two different values.

The setup is the next one:

- Sender transmits at a variable rate (exponential with average 1s)
- Receiver will fail to get the packet successfully with a probability *p*. Upon transmission, it sends an ACK after a short random "processing interval".
- Sender retransmits the "same" packet until success.
- Five probabilities (0.10, 0.25, 0.50, 0.75, 0.90)

Perform five repetitions and obtain:

- Average IPG and IPG for the receiver.
- Average PDR and IPG for messages from the sender
- Show the values in plots with error bars (i.e., confidence intervals of 95%)

Bonus Checkpoint

1. Statistics, plots, and explanations for IGGs, IPGs, and PDR.

Add your evidence here, writing a concise explanation of what you did.

Group members and work performed.

[Fabian Henrysson]

- Programming Part 1 - 3
- Part 3 activity 2 standard deviation script
- Part 3 Spread sheets/graphs

[Haron Obaid]

- Programming Part 1 - 3
- Remade Part 2
- Part 2 Spread sheets/graphs