

Homa Transport Protocol

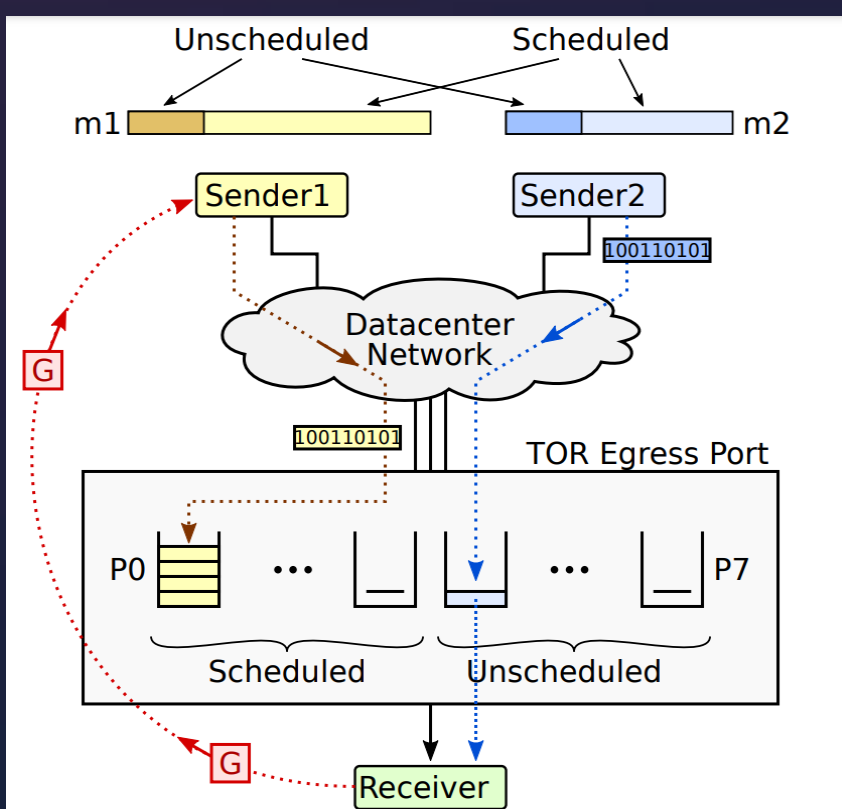
Students: Maria Correa

Product Owner: Wenjia Wang, Florida International University
Instructor/Faculty: Masoud Sadjadi, Florida International University

PROBLEM

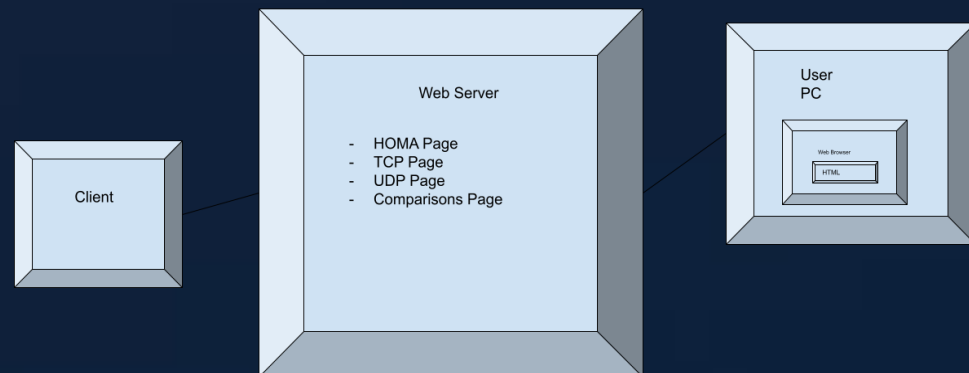
Traditional data transport protocols such as TCP, face the problem of high latency when dealing with high volumes of data transfers. Homa is a data transport protocol that was developed specifically to improve performance with high volumes of messages and large message sizes. It uses priority queues and a connectionless protocol to ensure low latency. Because of its innovative features, implementing Homa in different environments has shown to be difficult. Using ns-3, a network simulator, the Homa protocol was implemented and used for this comparison.

SYSTEM DESIGN

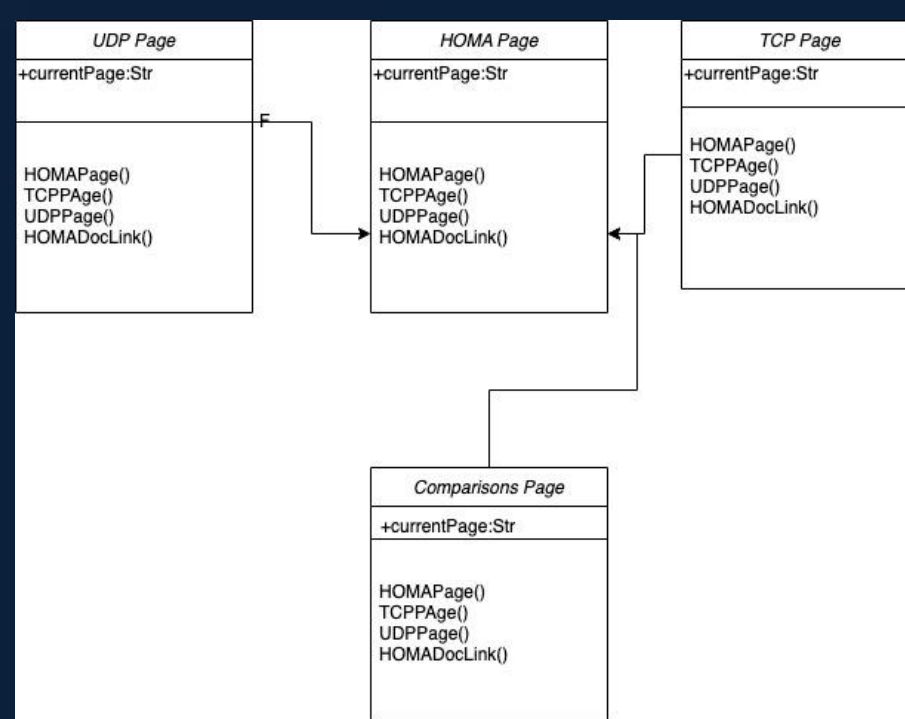


Sender1 is transmitting scheduled packets of message m1, while Sender2 is transmitting unscheduled packets of m2.

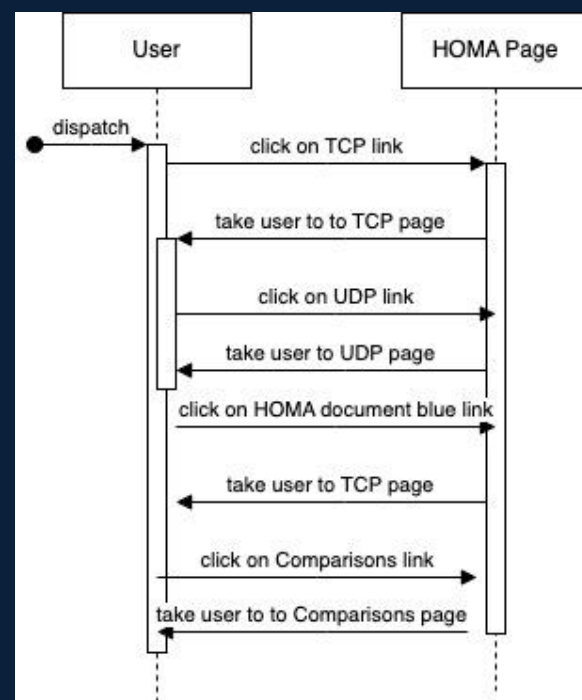
UML DIAGRAMS



Deployment Diagram.



Class Diagram.



Sequence Diagram.

CURRENT SYSTEM

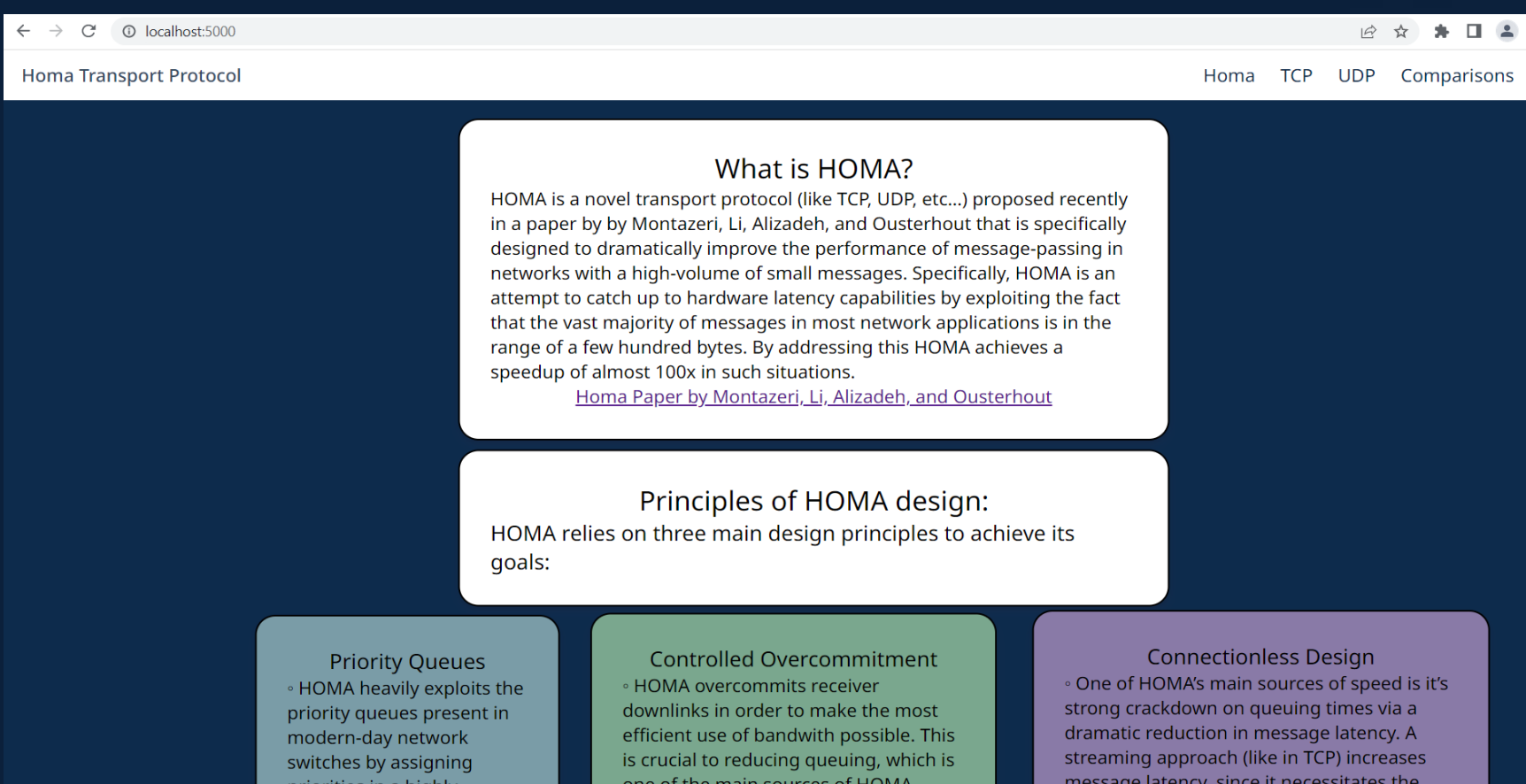
The current system of which was utilized for the comparisons uses ns-3, a free open-source network simulator. It provides different network models and examples to work with, such as a TPC and UDP model, which was used to minimize the variables in comparing them to the Homa protocol. The Homa protocol runs with pre-designed tests and hard-coded volume of messages with set sizes to replicate the low latency feature talked about in the original document. The way the tests and outputs of such tests are set up are also not user-friendly making it difficult to understand.

```
ncorr023@UbuntuHoma:~$ cd Homa/HOMA-project
ncorr023@UbuntuHoma:~/Homa/HOMA-project$ ./waf --run scratch/HomaL4Protocol-simple-test
Waf: Entering directory '/home/ncorr023/Homa/HOMA-project/build'
Waf: Leaving directory '/home/ncorr023/Homa/HOMA-project/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.837s)
Time it took: 233090 ns Message size: 532480 b
ncorr023@UbuntuHoma:~/Homa/HOMA-project$
```

Example output of a Homa test: Unclear on what the elements signify.

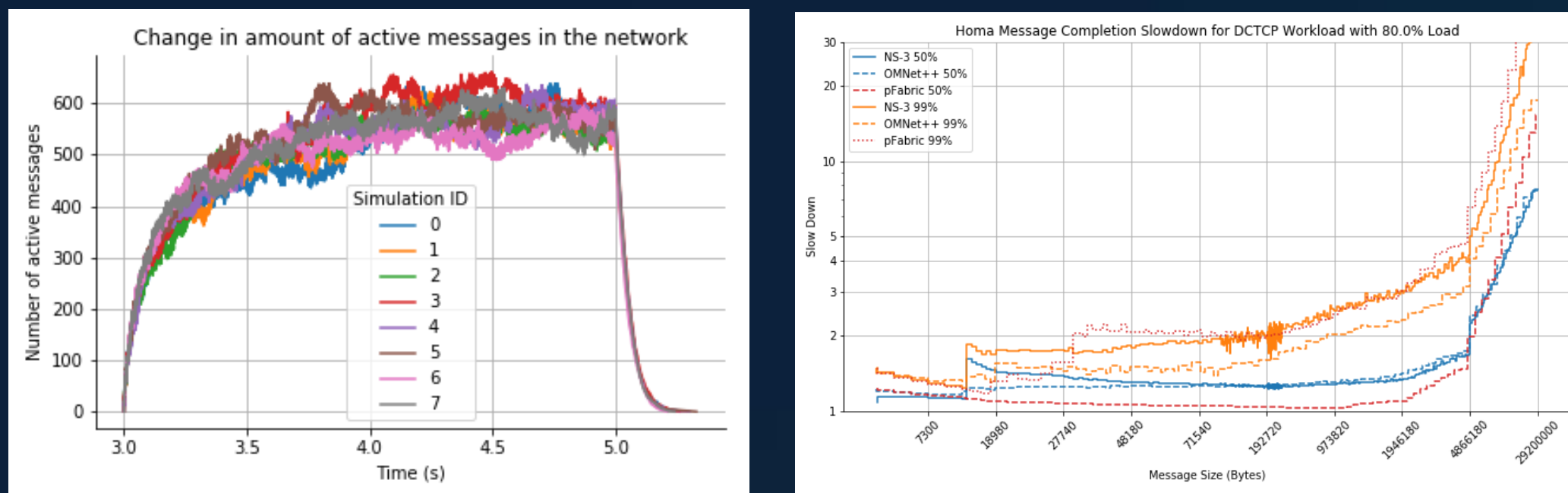
WEBSITE

A website was developed to highlighting the differences between HOMA and other common data transport protocols. JavaScript was used as the main programming language to develop the back-end of the website as well as the open-source server environment, Node JS, and the web application framework, Express. For our database, MySQL was used, and everything was connected locally. To develop the front-end of the first page named "Homa", HTML was for the design and CSS for the style.



Screenshot of the "Homa" page.

PERFORMANCE TESTS



IMPLEMENTATION

The first feature that was implemented was documentation on the test C++ scripts to be able to understand the process of how Homa works and how it gets its message and the size of each message. The second most important feature was to be able to manipulate the size of the message being sent to be able to freely compare Homa with other data transport protocols, such as TCP and UDP. The last implementation was to calculate the total time in nanoseconds it takes for the Homa protocol to send a message given its size.

```
ncorr023@UbuntuHoma:~/HomaCIS4911$ ./waf --run scratch/HomaL4Protocol-simple-test
Waf: Entering directory '/home/ncorr023/HomaCIS4911/build'
Waf: Leaving directory '/home/ncorr023/HomaCIS4911/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.837s)
Time it took: 233090 ns Message size: 532480 b
ncorr023@UbuntuHoma:~/HomaCIS4911$ ./waf --run scratch/TcpL4Protocol-simple-test
Waf: Entering directory '/home/ncorr023/HomaCIS4911/build'
Waf: Leaving directory '/home/ncorr023/HomaCIS4911/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.864s)
Time it took: 752504000 ns Message size: 532480 b
ncorr023@UbuntuHoma:~/HomaCIS4911$
```

Example output of a more readable and user-friendly Homa test.

INSTRUCTIONS

Explicit instructions on how to run the Homa test on a Linux environment: On the command-line interface:

- git clone <https://github.com/ncorr23/HomaCIS4911.git>
- ./waf configure
- ./waf

On the file /scratch/HomaL4Protocol-simple-test.cc:

- Change the size of the message to your liking: line 220 variable name "messageSize" (It accounts for Serialized Size of Homa and ipv4h)

On the Command-line interface:

- ./waf --run scratch/HomaL4Protocol-simple-test

SUMMARY AND SHORTCOMINGS

Unfortunately, there were a lot of issues and shortcoming trying to find working Homa repositories to work from. The first attempt was a Homa implementation as a Linux kernel module which failed due to unsupported versions not allowing the protocol to build or run correctly. The second attempt was a gRPC implementation of Homa which ran into similar issues including missing information due to it being in early stages of developing and getting dropped early on. In theory, the Homa Transport Protocol should be ideal for massive data transport but as of right now, there are not a lot of working implementations which could open the door to future projects and improvements.

REFERENCES

- Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: a receiver-driven low-latency transport protocol using network priorities. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 221–235. DOI:<https://doi-org.stanford.idm.oclc.org/10.1145/3230543.3230564>
- Arslan, Serhat. 2021. An NS-3 Implementation of Homa Transport Protocol, url = <https://github.com/serhatarslan-hub/HomaL4Protocol-ns-3>

ACKNOWLEDGEMENT

The material presented in this poster is based upon the work supported by Florida International University Knight Foundation School of Computing and Information Sciences. I thank Wenjia Wang for his assistance and mentorship that I received throughout the senior design project.