Gossip-based dissemination, Peer Sampling Service

Markus Roth

October 24, 2016

Contents

1 Solution Overview

1.1 Introduction

I created a single program that can be configured using constants at the top of the file. By changing these constants, it is possible to run all of the experiments in the assignment with the same piece of code. For each task, I have reproduced the configuration constants section of the program at the appropriate chapter in this document. To run the experiment, one needs to copy the configuration section into the correct section in gossip.lua and run it on the cluster. This section is marked with START CONFIG SECTION and END CONFIG SECTION. I chose this approach to reduce code duplication and guarantee a consistent reproducability of all results presented.

1.2 gossip.lua

This is the program that runs the experiements. If follows the provided skeleton. It contains the implementation of the peer samplingn service as well as the implementation of the rumor mongering and anti-entropy protocols.

1.3 parselog.lua

This LUA-script reads a log file created by running an experiment and aggragates the log entries to be able to parse them. It reads from the file named in its first parameter and writes to the same filename with a prefix of aggregated_. The aggregated log file contains the following columns:

1. relative_time: The number of seconds passed from the start of the experiment
2. cycles: The number of cycles passed frmo the start of the experiment
3. absolute_infected_nodes: The absolute number of nodes that are infected
4. relative_infected_nodes: The relative number of nodes that are infected
5. nodes_infected_by_anti_entropy: The cumulative number of nodes infected by the anti-entropy prodocol.
6. nodes_infected_by_rumor_mongering: The cumulative number of nodes infected by the rumor mongering prodocol.

1.4 parselog.lua

The same as parselog.lua, but sorts the entries by their relative time values rather than their cycles. Used for the second-plots.

1.5 plot_*.gp*

These gnuplot scripts read from aggregated log files produced by parselog.lua. They produces a .tex files of the graphs in the /plots directory. These .tex files are then embedded in this report via latex includes.

There are many adaptions of the script used for this report. For each exercise, the script exists named after the exercise number and the parameters used.

Note that I plotted the graphs with cycles in the x axis. This is because we were asked to do so in the lab session. Since I think plots with seconds are more legible, I added them at some places as well.

2 Task 2.1.1

Running gossip.lua with this configuration does what is required:

```
---START CONFIG SECTION---
do_anti_entropy = true
gossip_interval = 5
max_cycles = 10
--END CONFIG SECTION---
```

3 Task 2.1.2

Running the code from section 2.1.1 on 40 peers on the splay cluster gives the following log:

```
2016-10-24 13:55:02.972688 (92) 0 i_am_infected_as_patient_zero
2016-10-24 13:55:07.981443 (19) 0 i_am_infected_by_anti_entropy
2016-10-24 13:55:09.988781 (130) 1 i_am_infected_by_anti_entropy
2016-10-24 13:55:11.952454 (3) 1 i_am_infected_by_anti_entropy
2016-10-24 13:55:12.984608 (76) 1 i_am_infected_by_anti_entropy
2016-10-24 13:55:13.344219 (166) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:13.456906 (56) 1 i_am_infected_by_anti_entropy
2016-10-24 13:55:13.546528 (41) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:14.211685 (150) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:14.968156 (168) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:15.438669 (171) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:15.577630 (96) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:15.781953 (58) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:16.358811 (115) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:16.738897 (167) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:16.984977 (112) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.197953 (113) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.365741 (38) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.605150 (77) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.682568 (40) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.738057 (151) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:17.986358 (78) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:18.346416 (20) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:18.458532 (18) 2 i_am_infected_by_anti_entropy
```

```
2016-10-24 13:55:18.550194 (55) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:19.214494 (22) 2 i_am_infected_by_anti_entropy
2016-10-24 13:55:19.366339 (129) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:19.593522 (132) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:20.190240 (1) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:20.559921 (57) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:20.579709 (114) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:20.768034 (74) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:20.796862 (149) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:21.185359 (111) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:21.331993 (169) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:21.552212 (75) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:21.887625 (39) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:21.911745 (95) 3 i_am_infected_by_anti_gntropy
2016-10-24 13:55:22.369056 (133) 3 i_am_infected_by_anti_entropy
2016-10-24 13:55:25.245777 (148) 4 i_am_infected_by_anti_entropy
```

As we can see, dissemination is complete after around 4 cycles. Not included: Final messages by all nodes saying that they are indeed infected.

4 Task 2.1.3

The script for parsing logs can be found in the file parselog.lua.

5 Task 2.1.4

The gnuplot script can be found in the file plot.gp. It might be modied in details to produce the various graphs in this report, but in general it remains the same.

Applying the script to the data gained in task 2.1.2, we get plot show in figure 1.

6 Task 2.2.1

```
---START CONFIG SECTION---
do_rumor_mongering = true
gossip_interval = 5
initial_hops_to_live = 3
distribution_count = 2
max_cycles = 5
---END CONFIG SECTION---
```

The results of the experiment can be found in 2. As we can see, 12 nodes were infected and two duplicates received, leading to the expected number of 14 recipients of a rumor. Since the cycles are not aligned, calls tend to move from cycle n to cyle n-1, because cycle n-1 runs first. This is the reason we see all infection shappen within cycles 0 to 3, rather than the last infections only happening in cycle 3.

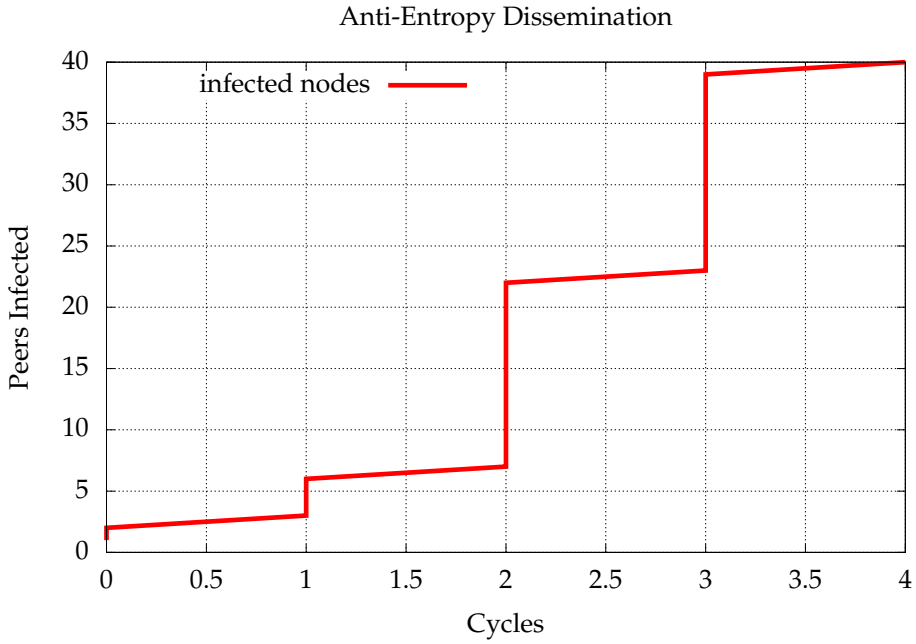

Figure 1: Task 2.1.4: Anti-Entropy dissemination with 40 peers

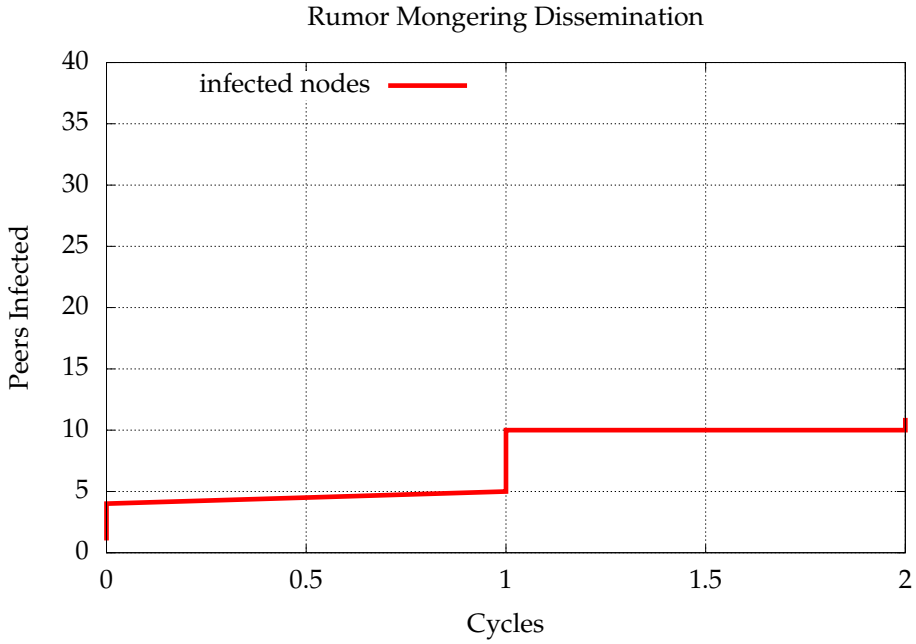

Figure 2: Task 2.2.1: Rumor Mongering dissemination with 40 peers (HTL=3, f=2)

7 Task 2.2.2

The log of Task 2.2.1 lists the incoming messages by rumor mongering:

```
2016-10-24 16:06:16.338689 (18)  0 i_am_infected_as_patient_zero
2016-10-24 16:06:21.348536 (39)  0 i_am_infected_by_rumor_mongering
2016-10-24 16:06:21.353064 (170)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:22.965381 (113)  0 i_am_infected_by_rumor_mongering
2016-10-24 16:06:22.970365 (150)  0 i_am_infected_by_rumor_mongering
2016-10-24 16:06:24.672263 (41)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:24.677613 (22)  2 i_am_infected_by_rumor_mongering
2016-10-24 16:06:24.972311 (37)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:24.976962 (22)  2 duplicate_received
2016-10-24 16:06:25.14612 (168)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:25.19327 (167)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:26.768355 (168)  1 duplicate_received
2016-10-24 16:06:26.771005 (55)  1 i_am_infected_by_rumor_mongering
2016-10-24 16:06:29.112913 (39)  2 duplicate_received
2016-10-24 16:06:29.115037 (55)  2 duplicate_received
```

Again we note that the cycles are not aligned, so we see the infections happing in an earlier cycle than they have been issued.

8 Task 2.2.3

Modyfing the initial_hops_to_live and distribution_count variables in the config section above, I examined some values of f and HTL. The results can be seen in figure 3.

It seems to reach all my nodes, I need the parameter f to be at least 4 with HTL around 5.

For better readability, I also plotted the graph using seconds in the x axis. This plit is in figure 4.

9 Task 2.2.4

The contained scripts parselog.lua and the various plot scrips are able to plot total infections and infections by type of infection. The log outputs both values, and the plot script decides which to plot.

10 Task 2.3

To show a comparison of the different infection methods, I use the following configuration:

```
---START CONFIG SECTION---
do_rumor_mongering = true
do_anti_entropy = true
gossip_interval = 5
initial_hops_to_live = 3
distribution_count = 2
```

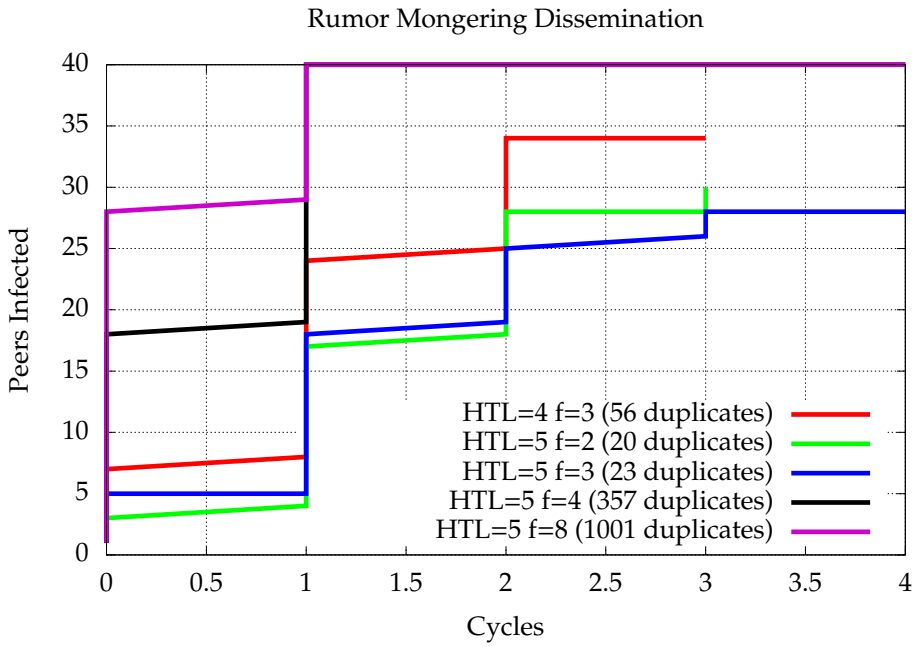

Figure 3: Task 2.2.3: Comparison of f and HTL values for Rumor Mongering (40 peers)

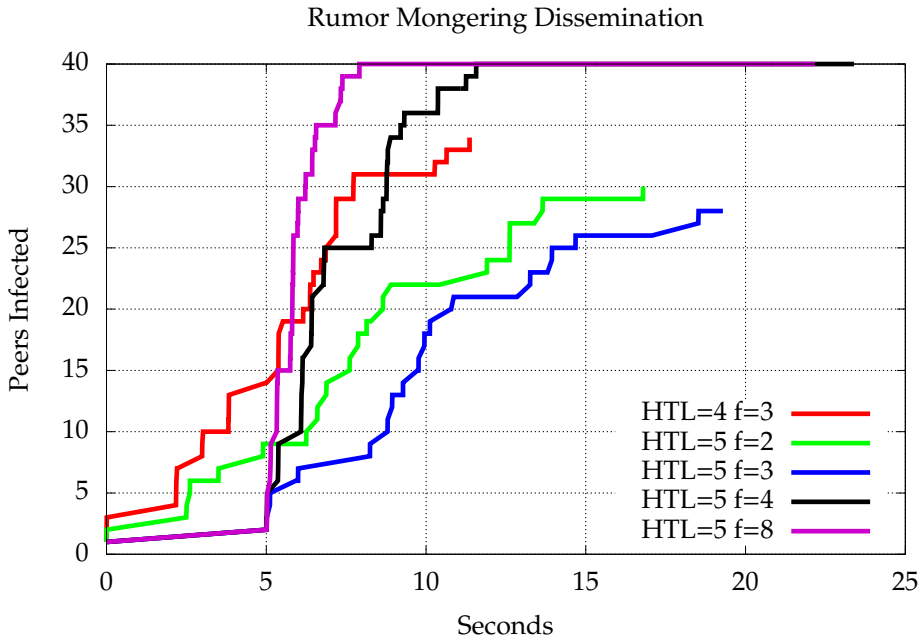

Figure 4: Task 2.2.3: Comparison of f and HTL values for Rumor Mongering (40 peers)

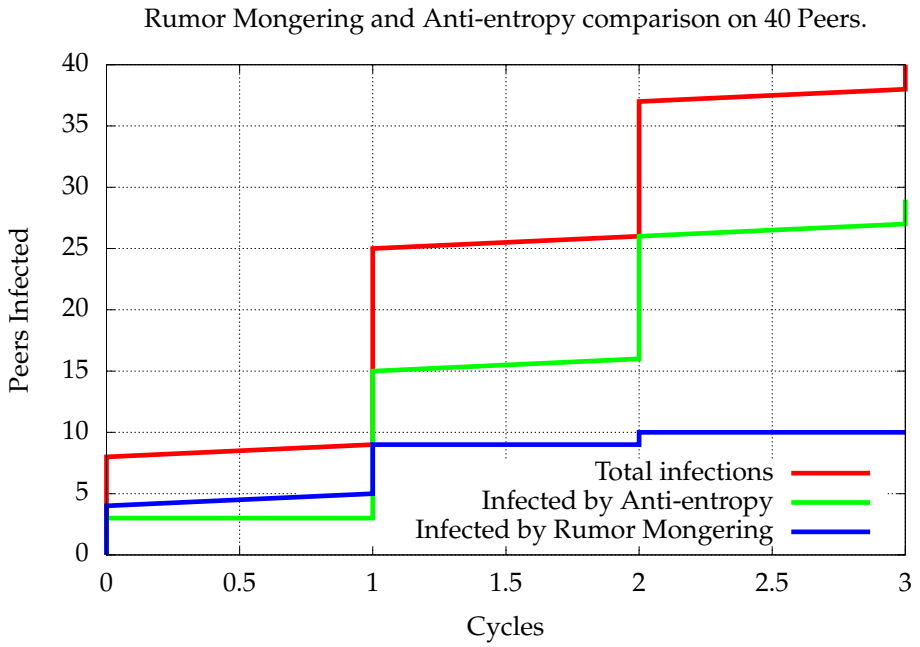

Figure 5: Task 2.3: Comparison of Anti-entropy and Rumor Mongering on 40 peers

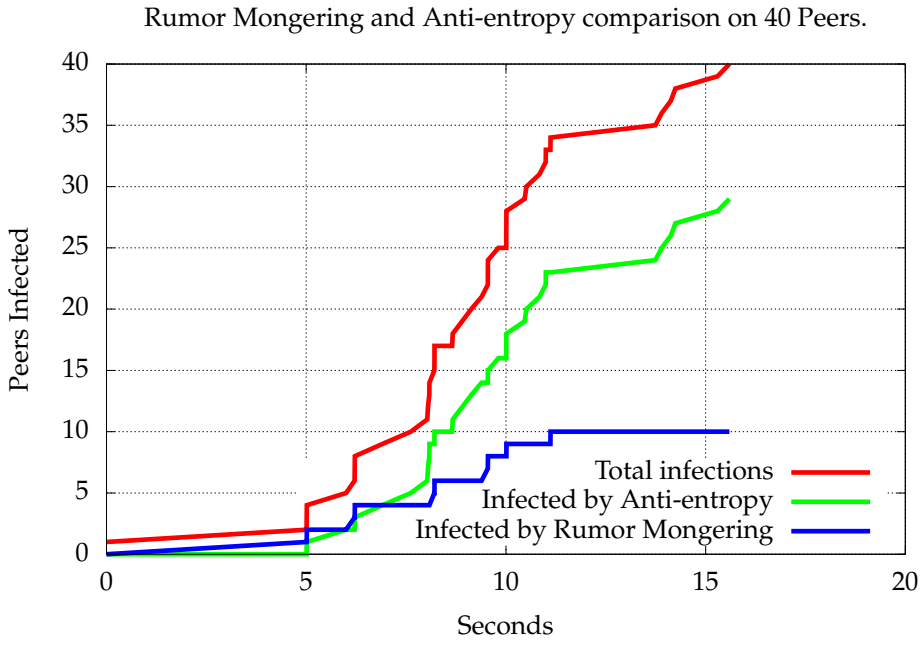

Figure 6: Task 2.3: Comparison of Anti-entropy and Rumor Mongering on 40 peers

```
max_cycles = 20
start_gossipping_after_cycles = 0
---END CONFIG SECTION---
```

The resulting plot can be seen in figure 5. There were 4 duplicates in the Rumor Mongering. I also plotted the graph with seconds in the x axis in figure 6.

11 Task 3.2

I implemented the Peer Sampling Service in the gossip.lua file. I also wrote some unit tests for the array operations. To start gossip.lua in unit test mode, set the unit_test_mode constant to true. It will the run all the tests and end, and not actually start the simulation.

To use the Peer Sampling Service without any dissemination, I can use the following configuration:

```
---START CONFIG SECTION---
do_peer_sampling = true
peer_sampling_interval = 5
peer_sampling_view_size = 8
peer_sampling_exchange_rate = 4
peer_sampling_healer_parameter = 1
peer_sampling_shuffler_parameter = 1
max_cycles = 10
---END CONFIG SECTION---
```

According to the check partion script, the graph is connected:

```
The network is connected
```

The In-degree and clustering graphs can be seen in figures 8 and 7, respectively.

Varying the H and S parameters as recommended gives the results listed in graphs 9 to 14.

12 Task 4.1

This switch is implemented in gossip.lua. if the do_peer _sampling flag is set, the methods for returning peers for the dissemination protocols take their peers from the local view. Otherwise, they take them from the total list of available nodes.

13 Task 4.2

```
---START CONFIG SECTION---
do_rumor_mongering = true
do_anti_entropy = true
```

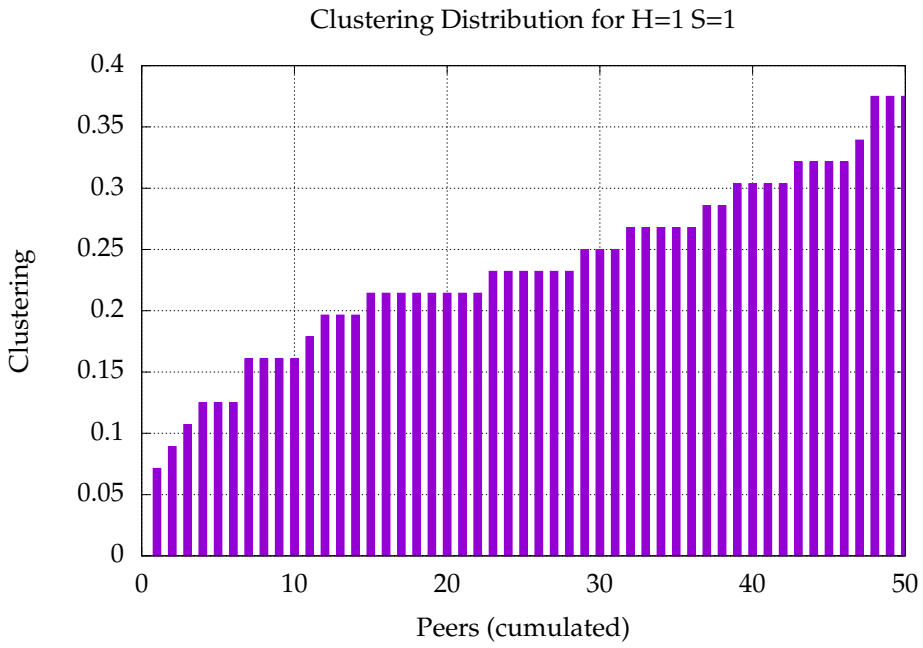

Figure 7: Task 3.2: Clustering (50 Peers, H=1, S=1)

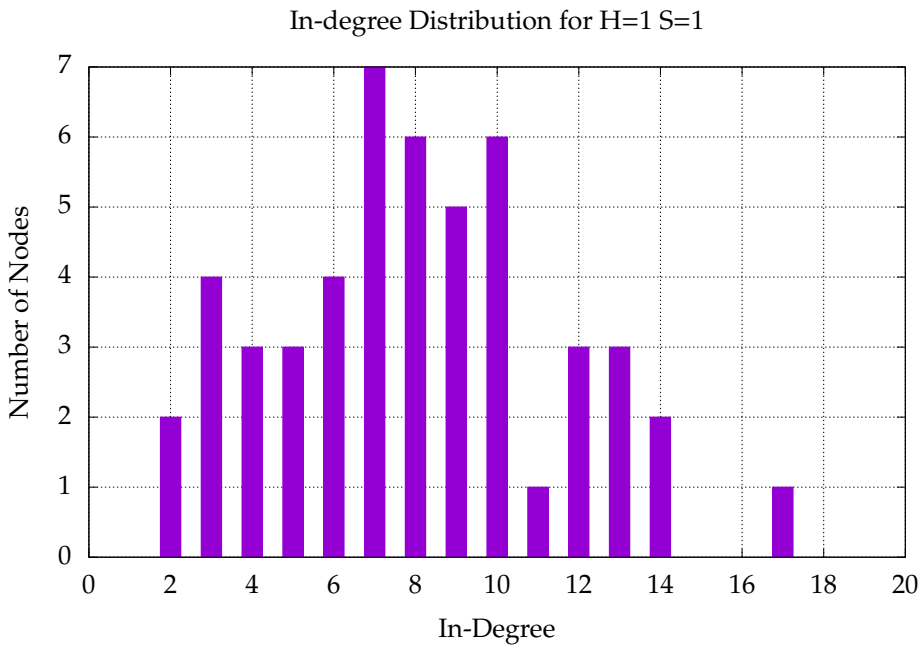

Figure 8: Task 3.2: In-degrees (50 Peers, H=1, S=1)

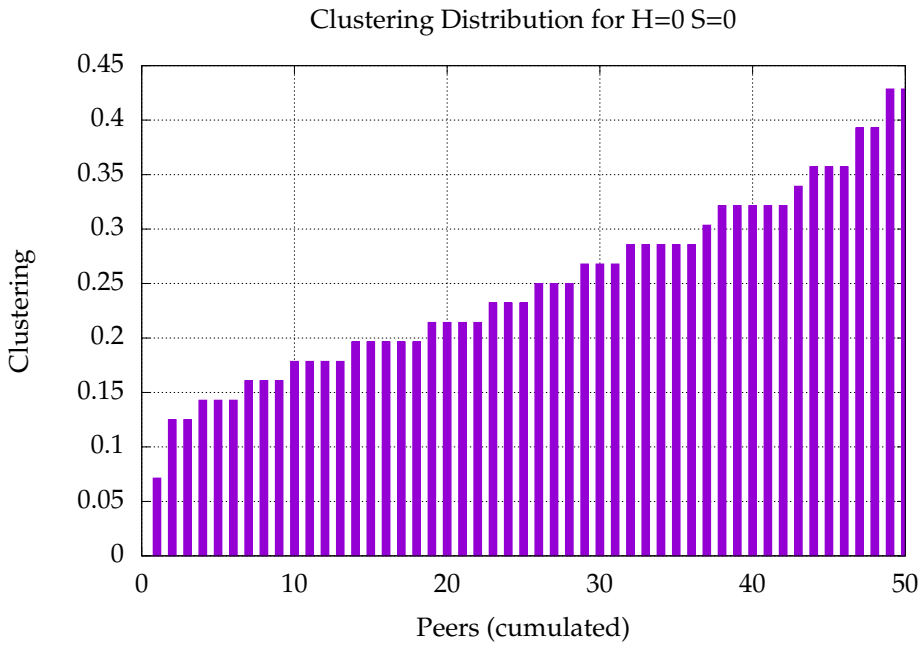

Figure 9: Task 3.2: Clustering (50 Peers, H=0, S=0)

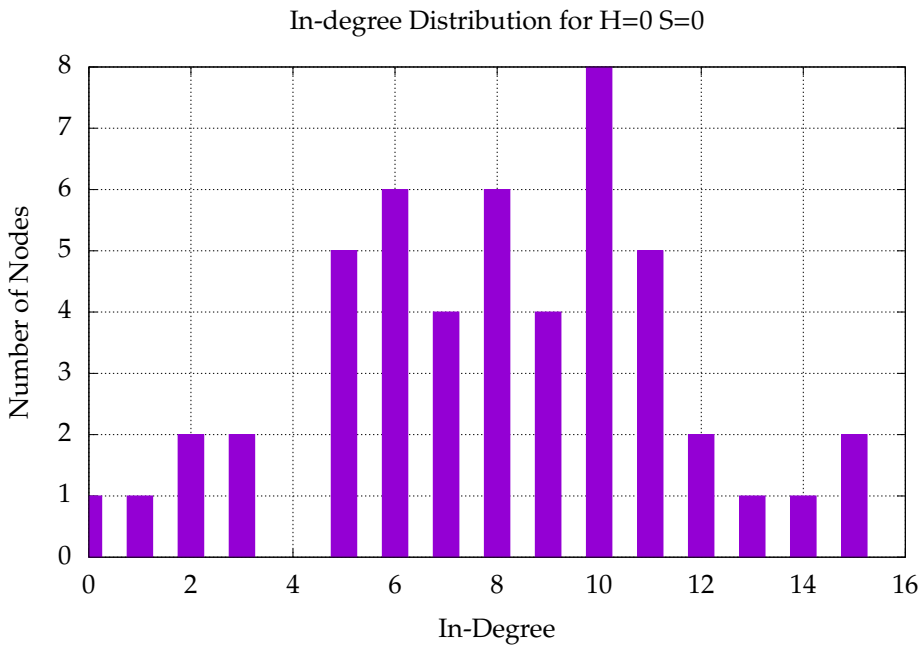

Figure 10: Task 3.2: In-degrees (50 Peers, H=0, S=0)

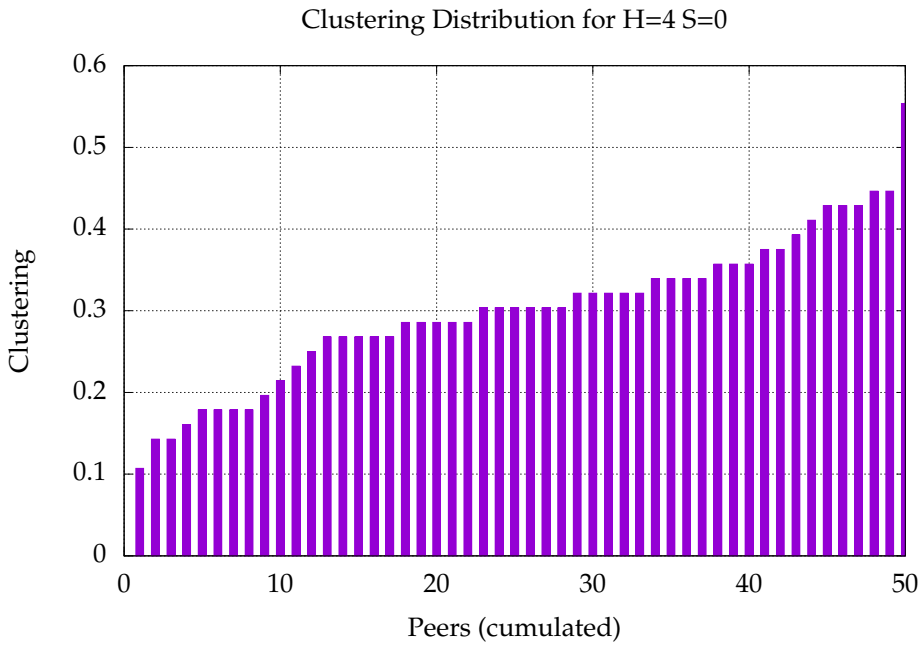

Figure 11: Task 3.2: Clustering (50 Peers, H=4, S=0)

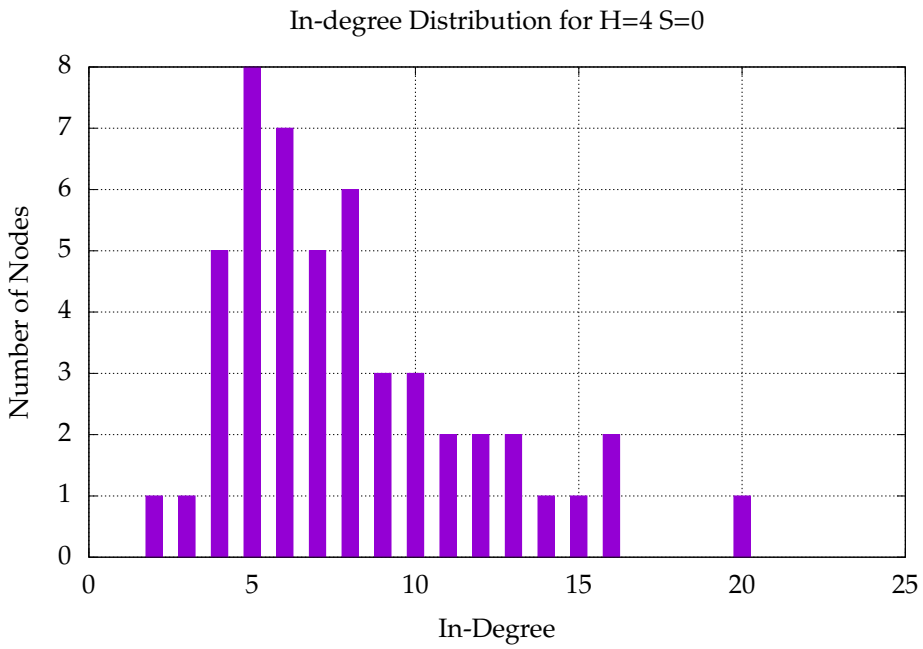

Figure 12: Task 3.2: In-degrees (50 Peers, H=4, S=0)

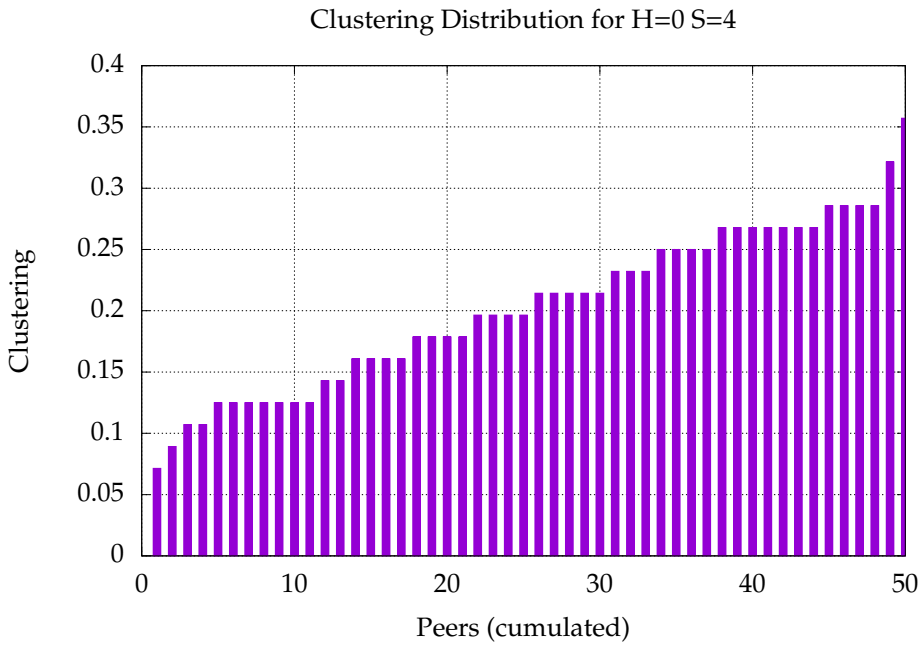

Figure 13: Task 3.2: Clustering (50 Peers, H=0, S=0)

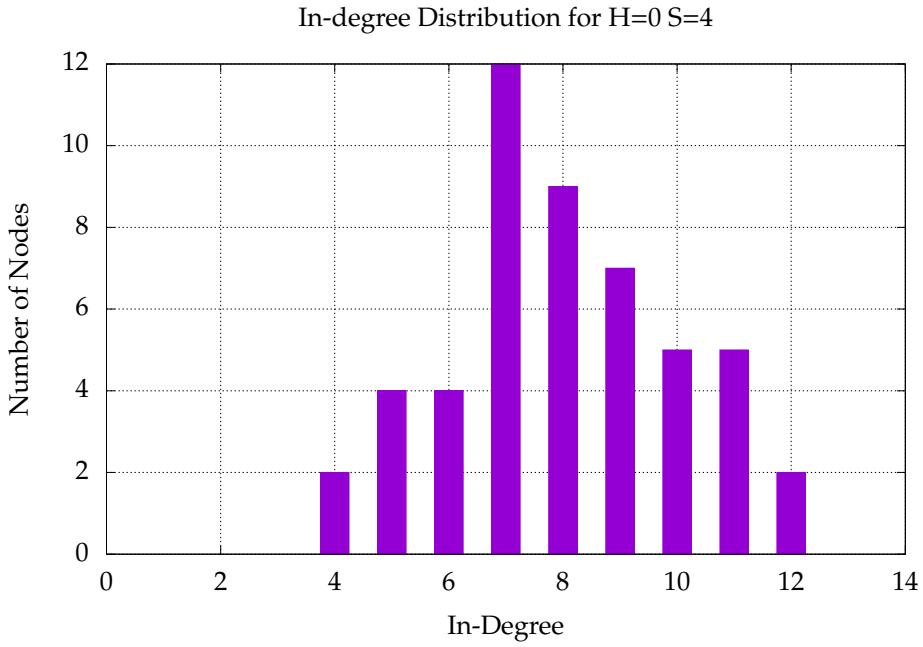

Figure 14: Task 3.2: In-degrees (50 Peers, H=0, S=4)

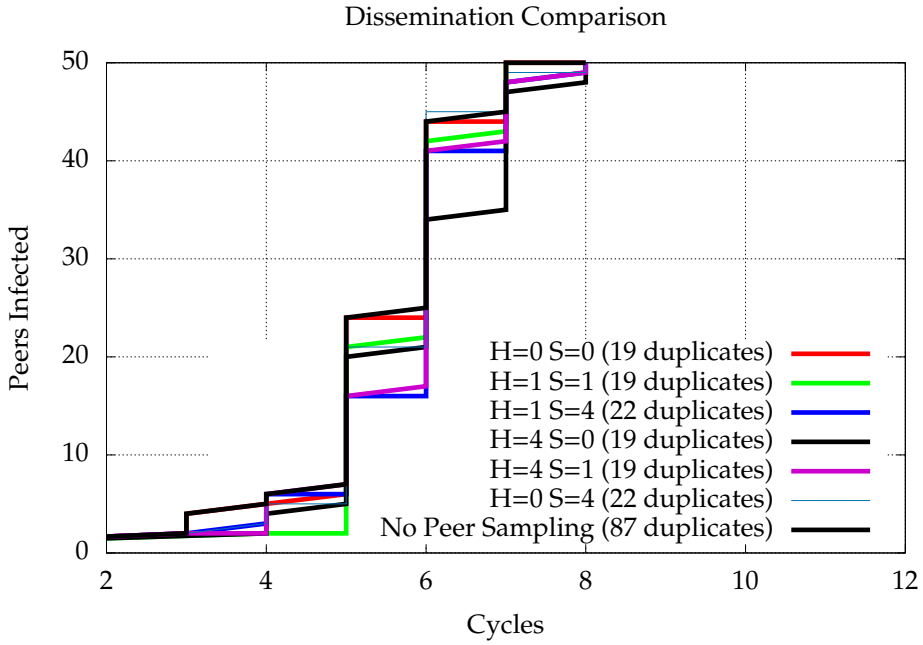

Figure 15: Task 4.2: Dissemination Comparison with Peer Sampling

```
do_peer_sampling = true
gossip_interval = 2
peer_sampling_interval = 5
initial_hops_to_live = 4
distribution_count = 3
peer_sampling_view_size = 8
peer_sampling_exchange_rate = 3
peer_sampling_healer_parameter = 0
peer_sampling_shuffler_parameter = 0
peer_selection_policy = "rand"
max_cycles = 10
start_gossipping_after_cycles = 5
---END CONFIG SECTION---
```

Finally, I compared various values of healer and shuffler parameters and plottet the results. I used the above configuration, with changed values in the peer_sampling_healer_parameter and peer_sampling_shuffler_parameter. The results can be found in figure 15. In this figure you can also see the duplicate rates. For this particular graph, I wanted to see the results in seconds too, so I added an additional plot with seconds on the x axis in figure 16.

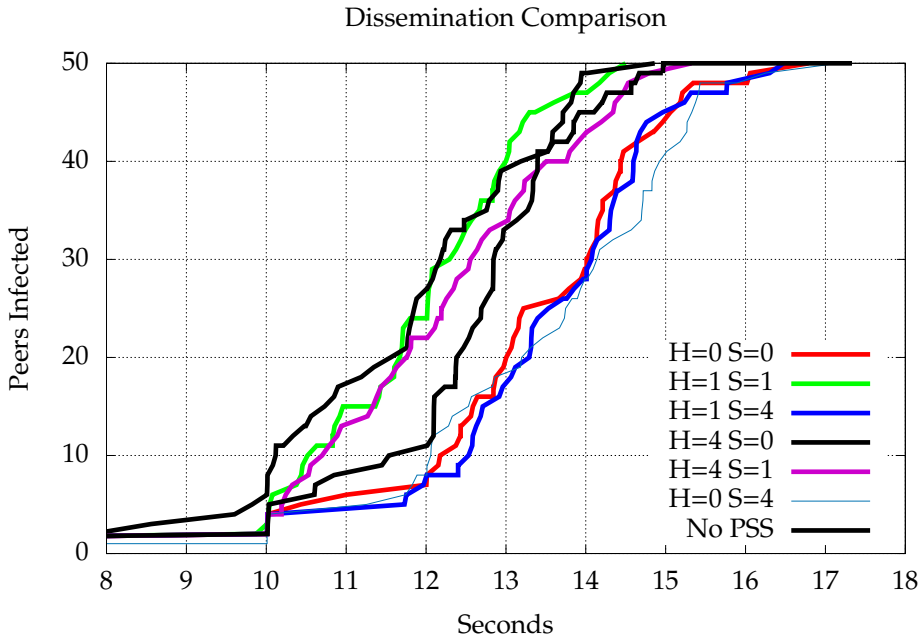

Figure 16: Task 4.2: Dissemination Comparison with Peer Sampling