COMP480 HW2 Report

David Dai (wd16)

1. To run this program, just uncomment the four lines of run function and then execute the python file. The plot will automatically show up. The program may take some time since the Consistent Hashing with Bounded Loads can be slow sometimes.

2. Assumptions:

- To handle cycles, I have two solutions. Firstly, I pass a random seed range from [1, 2147483647] along with the value into the murmur-hash function. In this way, it will be more random and will be less likely to have cycles. Secondly, if I found out the value has been hashed over a certain number of times (In my case, it is 80), I will perturb the original value by a random number from [1, 100]. Therefore, it becomes very unlikely to have a cycle during hashing.

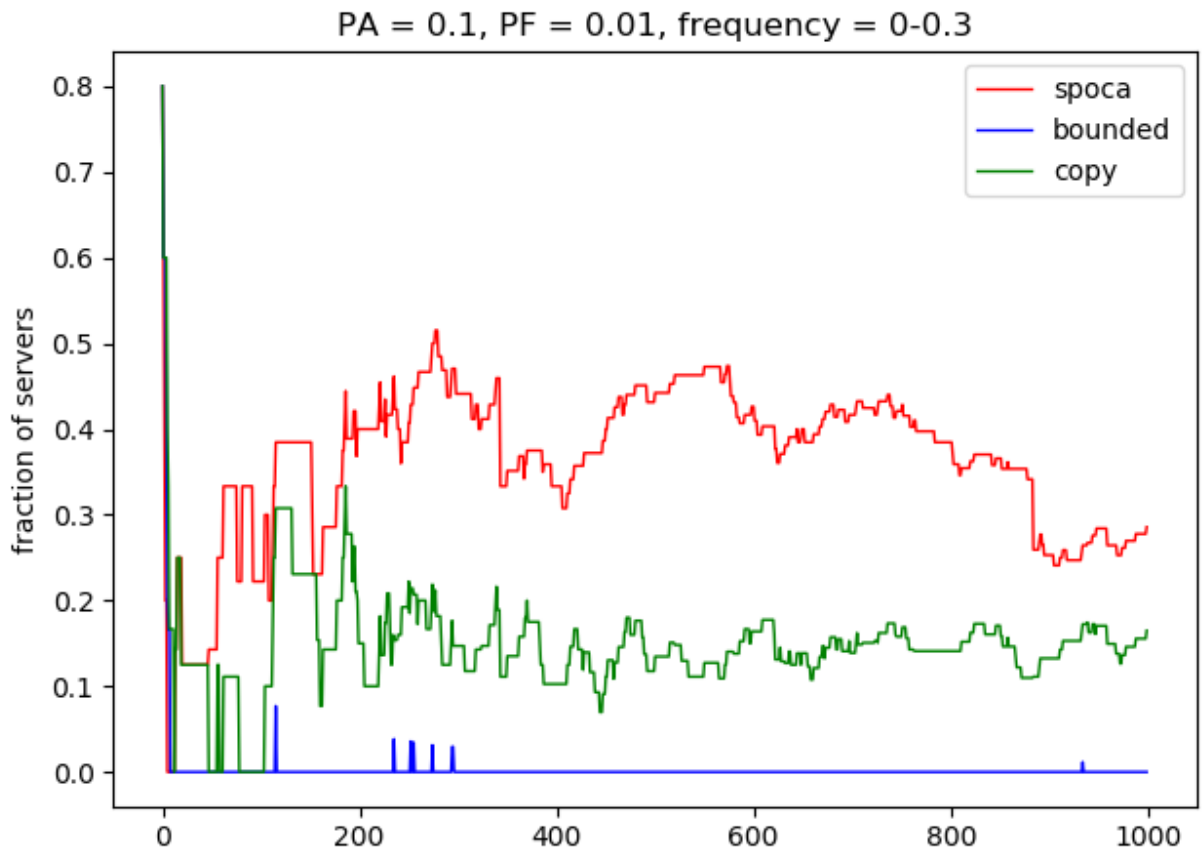
- For SPOCA and Consistent Hashing with multiple copies, I just directly add the job to the server regardless the server's loads. For Consistent Hashing with Bounded Loads, I first check if all the servers are full. If it is the case, I will add the element to the server that has the smallest overhead as required. If it is not the case, I will only add the task into the servers that are not overloaded.

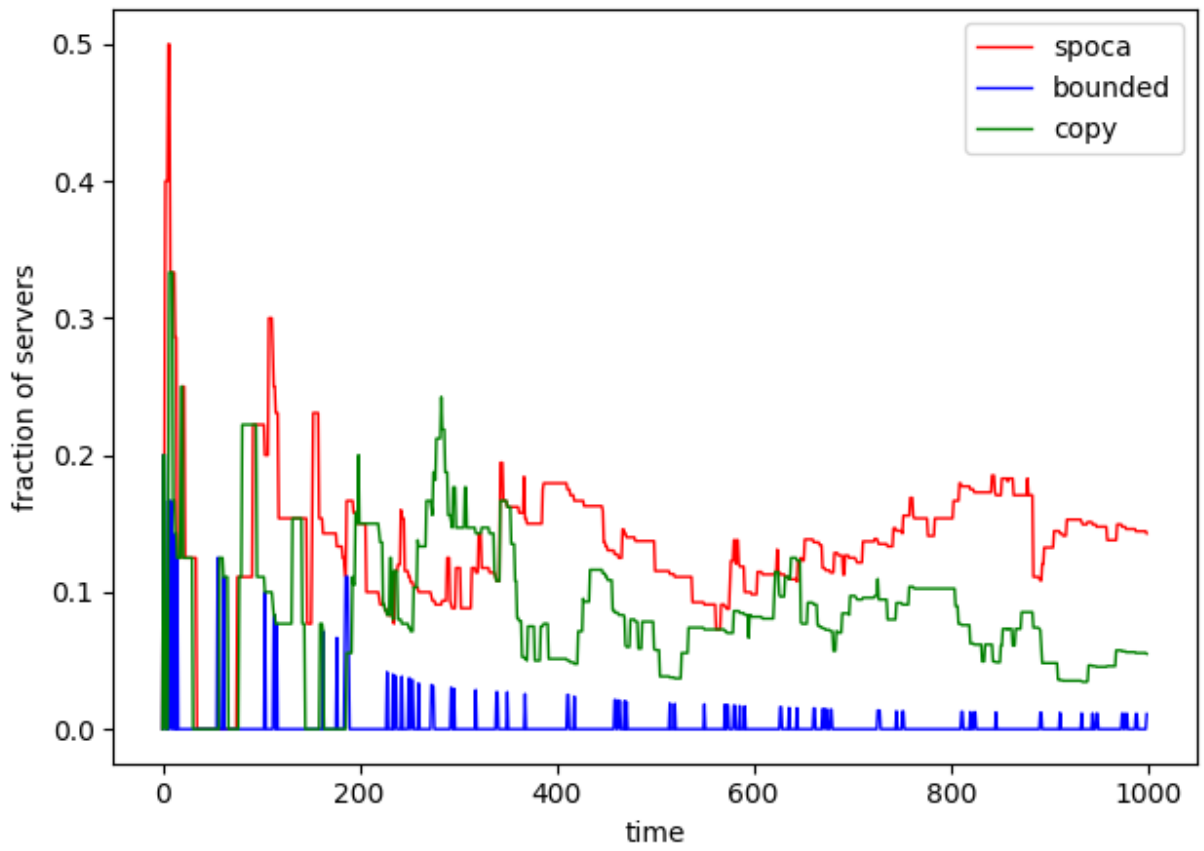
- The expected total capacity of the servers are 2*(n + (pa - pf) * 1000) where 2 is the expected size of the servers and (n + (pa - pf) * 1000) is the expected number of servers after 1000 iterations.

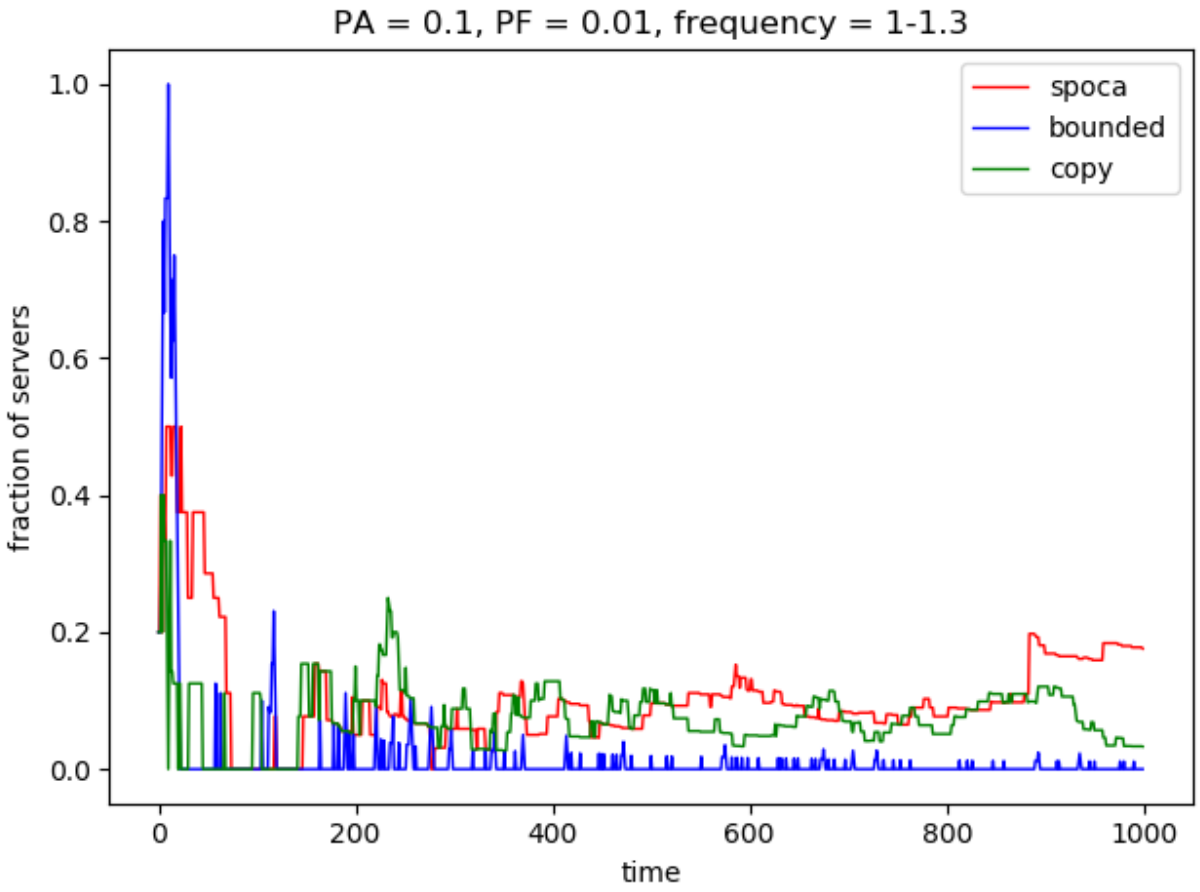
3. Report:

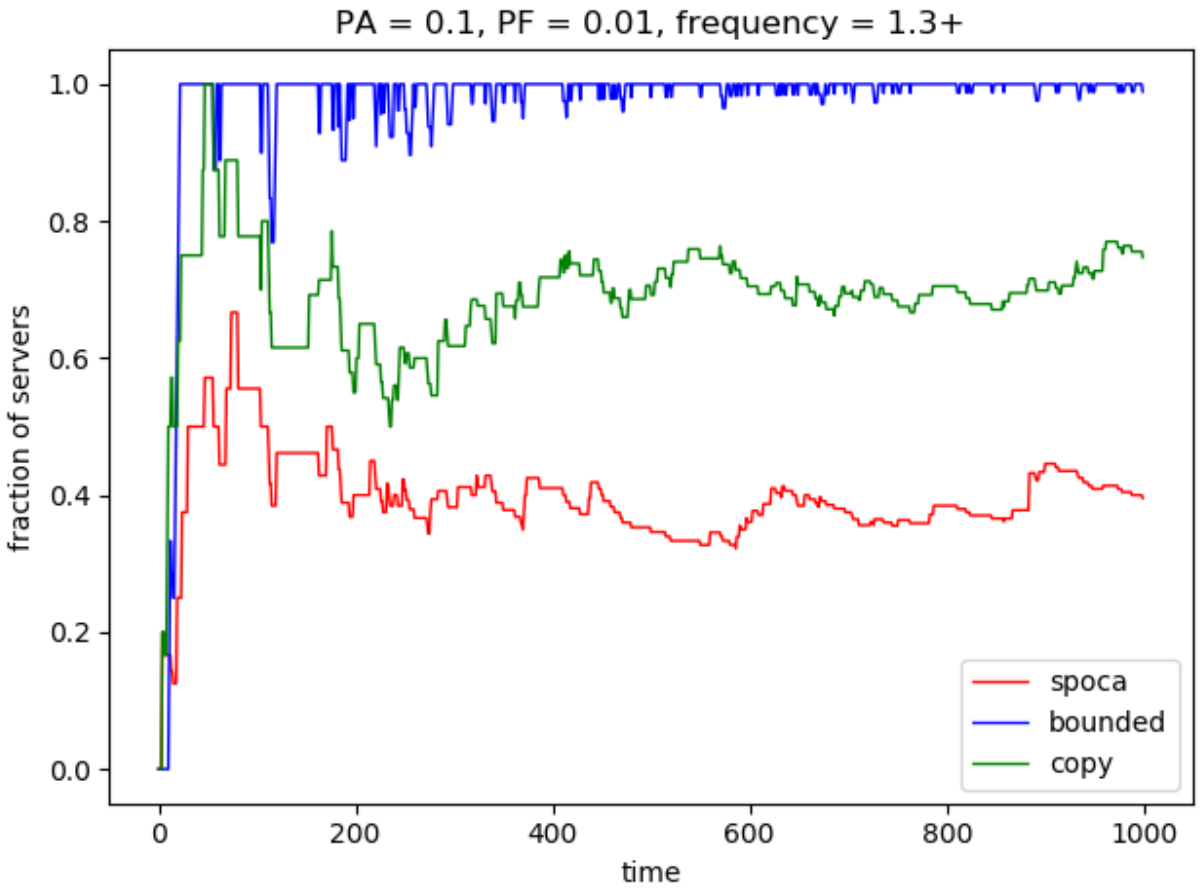
The first run is when pa = 0.1, pf = 0.01. The second run is when pa = 0.1, pf = 0.005. The third run is when pa = 0.05, pf = 0.01.The third run is when pa = 0.05, pf = 0.005.

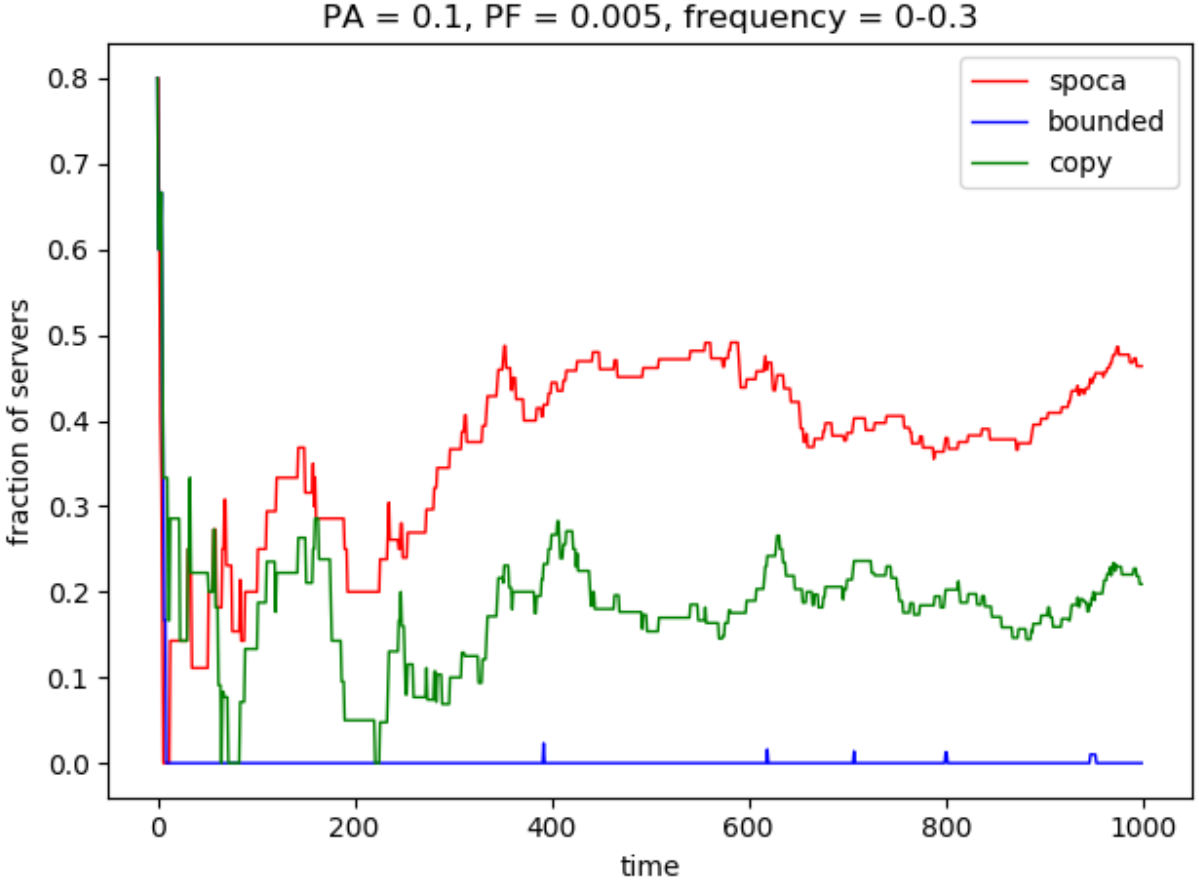
The last four plots are the overheads. Plots are attached on next few pages.

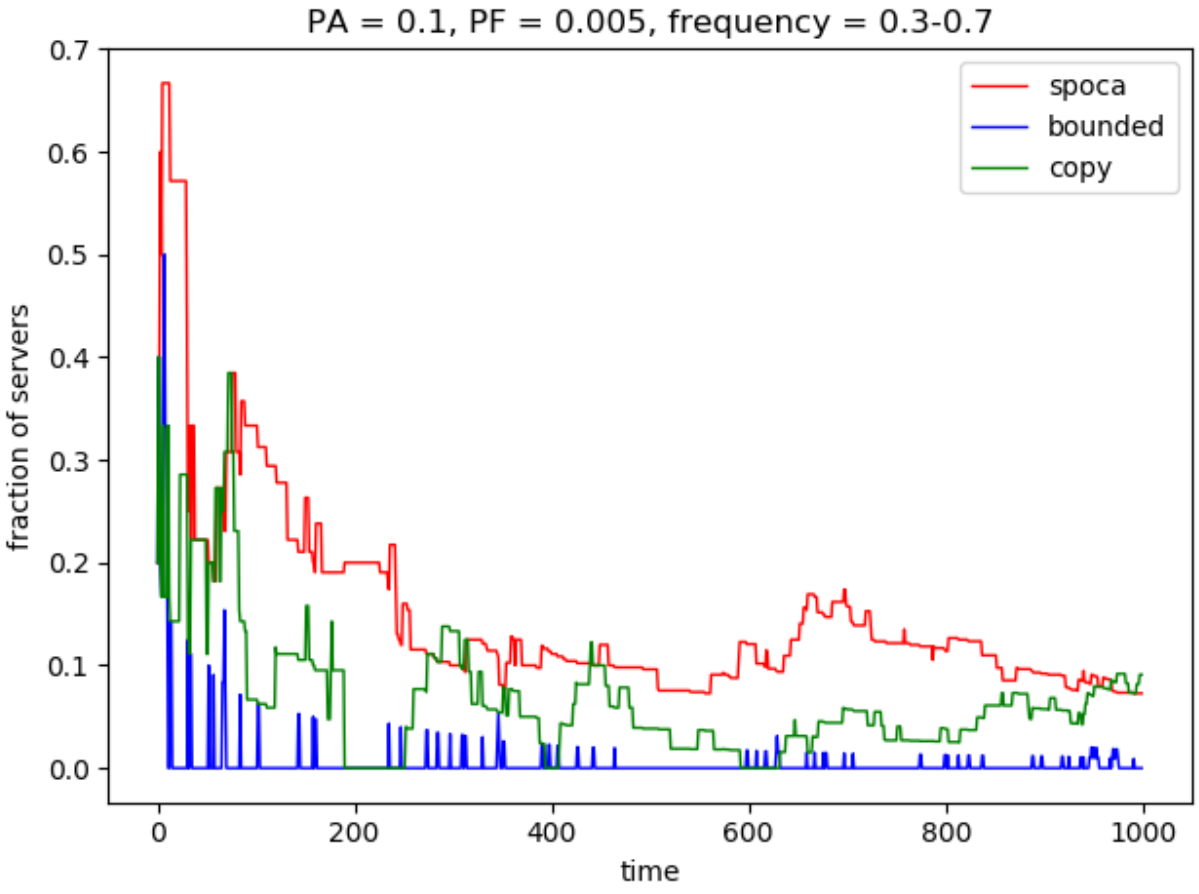

PA = 0.1, PF = 0.01, frequency = 0-0.3
spoca
bounded
copy
fraction of servers
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

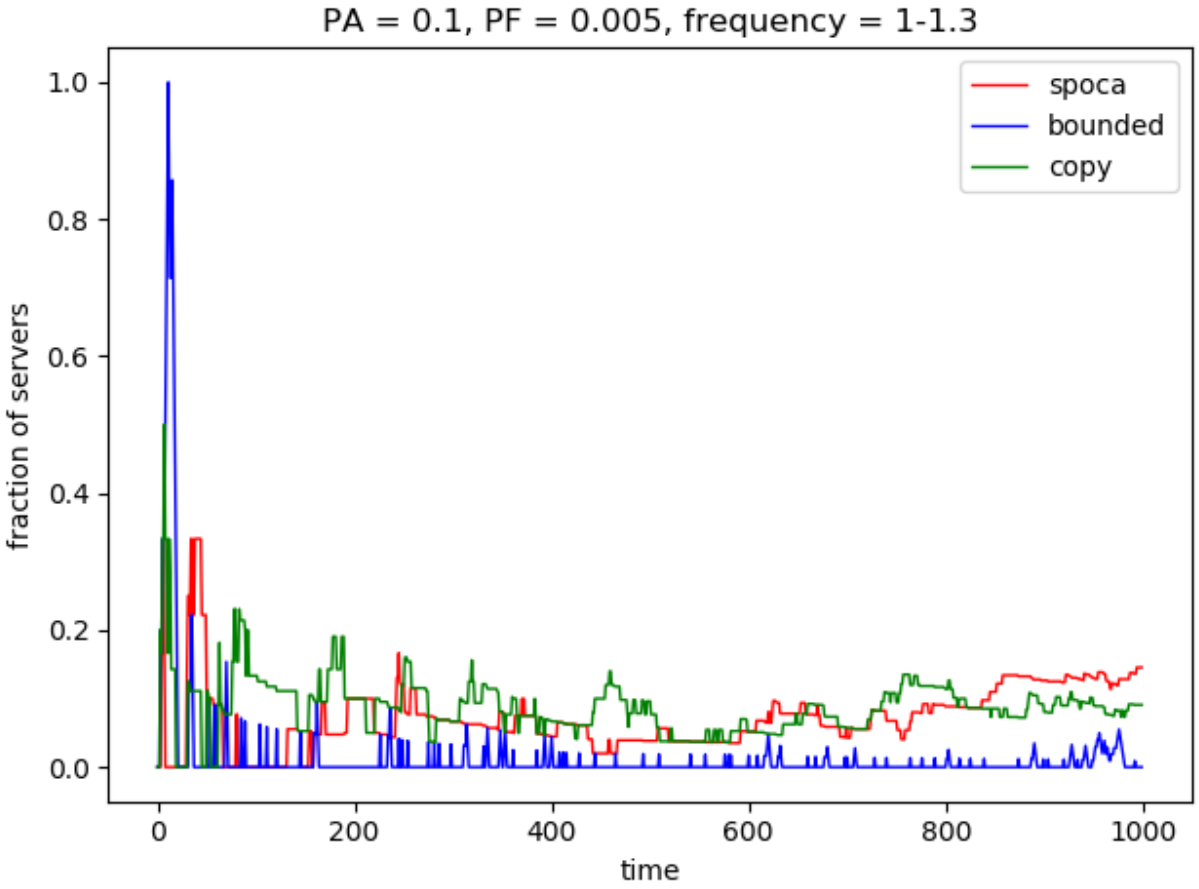

PA = 0.1, PF = 0.01, frequency = 0.3-0.7
spoca
bounded
copy
fraction of servers
0.0
0.1
0.2
0.3
0.4
0.5
0
200
400
600
800
1000
time

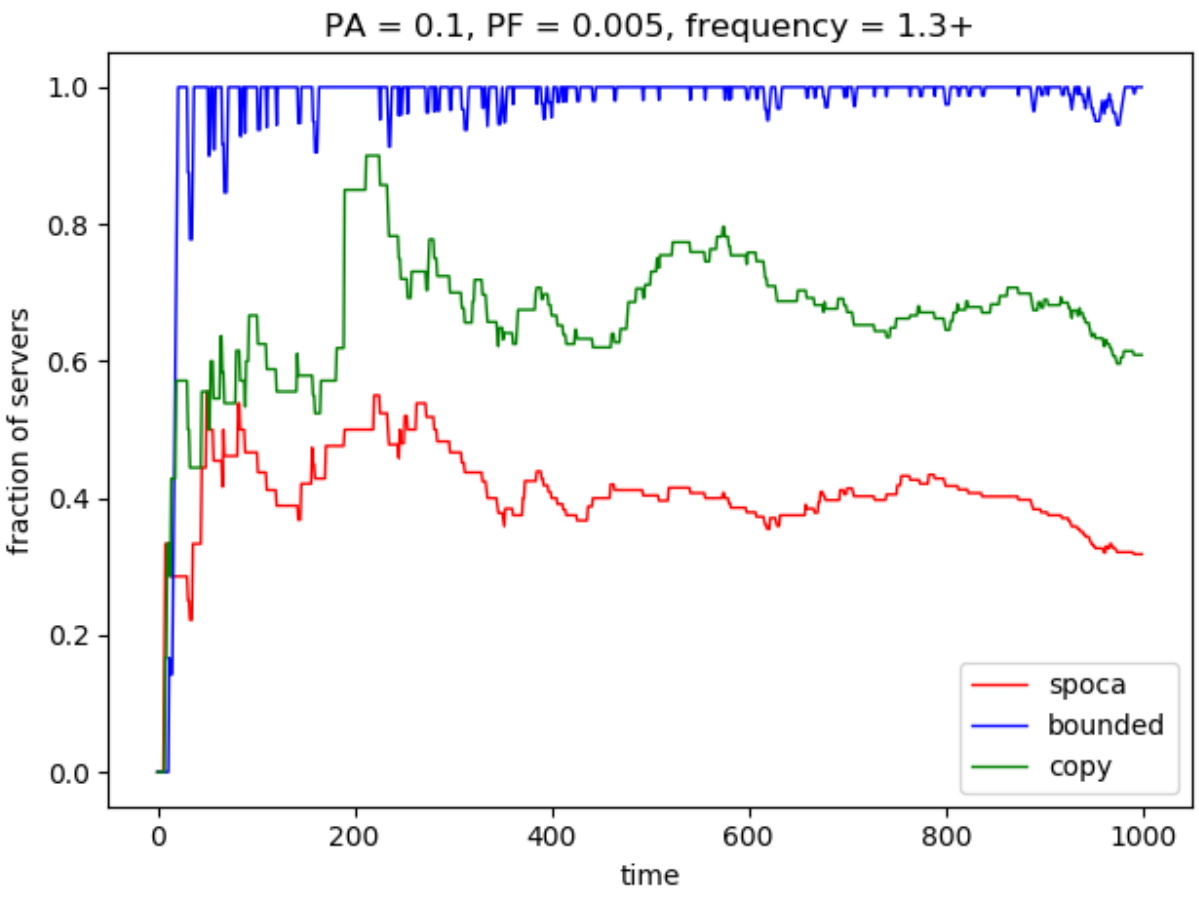

PA = 0.1, PF = 0.01, frequency = 1-1.3
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

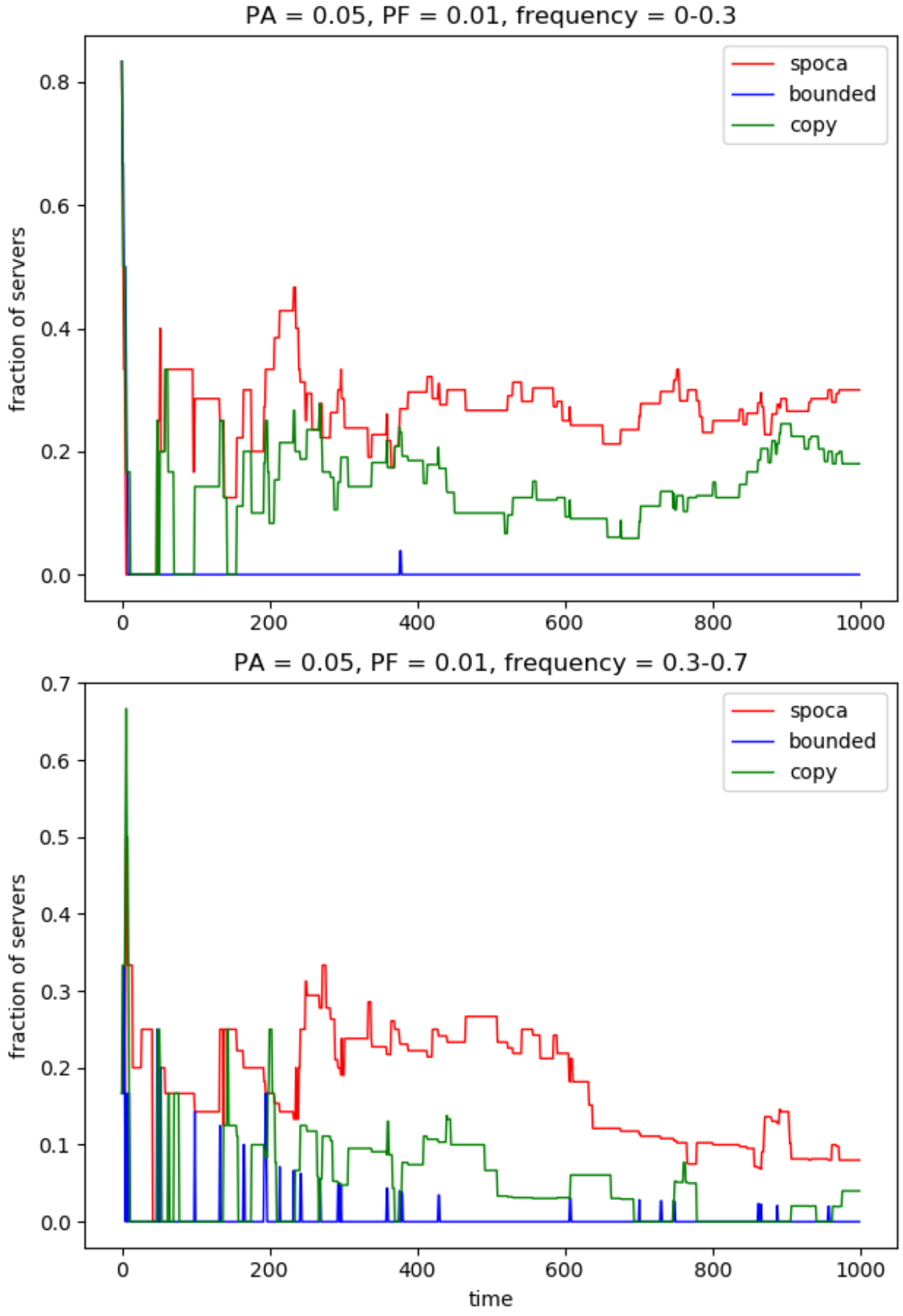

PA = 0.1, PF = 0.01, frequency = 1.3+
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

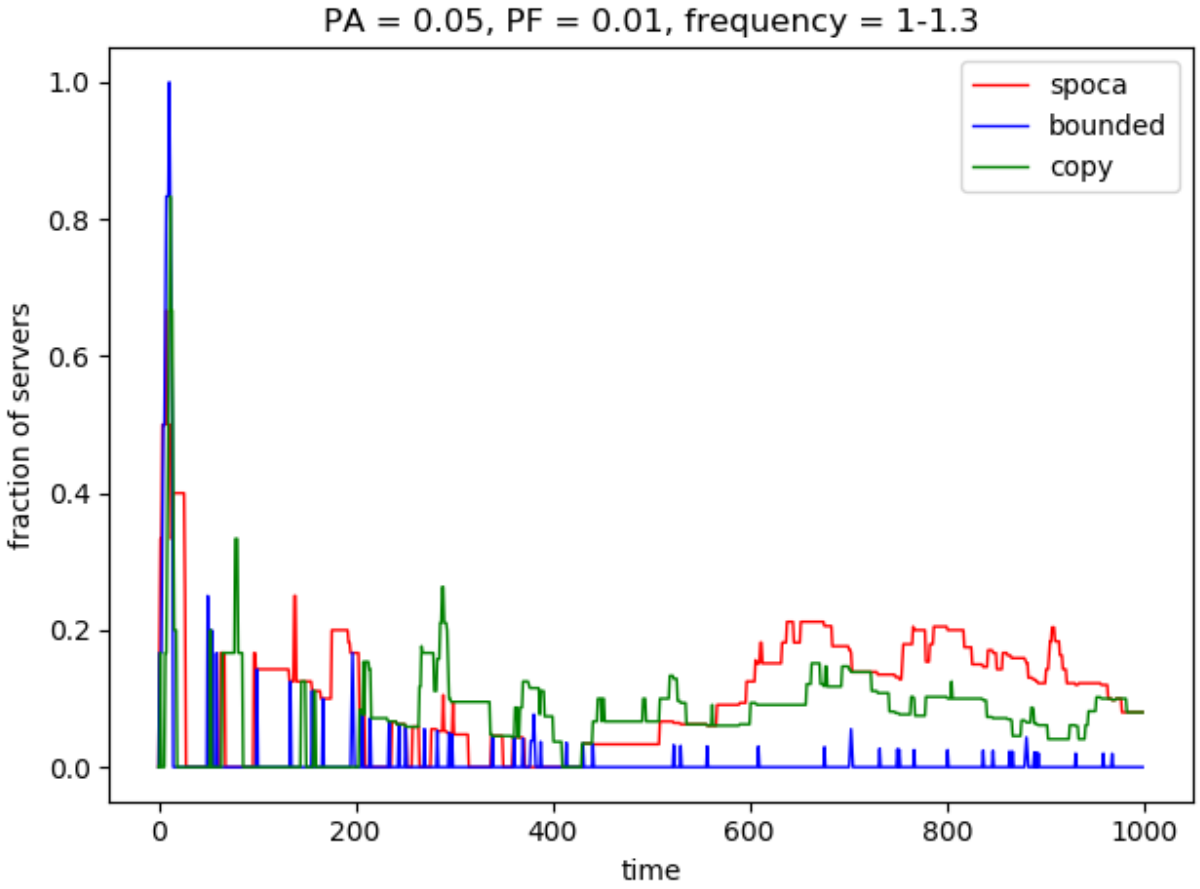

PA = 0.1, PF = 0.005, frequency = 0-0.3
spoca
bounded
copy
fraction of servers
time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

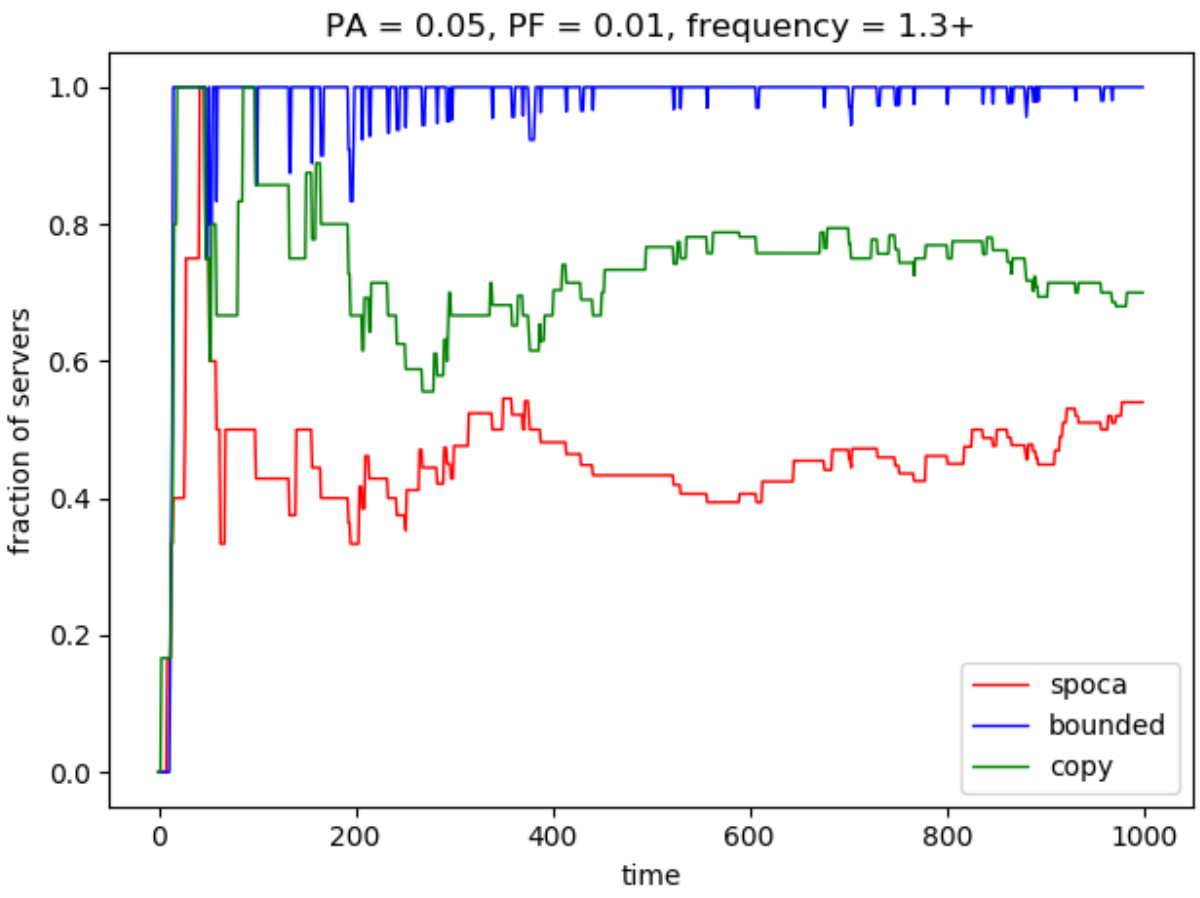

PA = 0.1, PF = 0.005, frequency = 0.3-0.7
spoca
bounded
copy
fraction of servers
time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000

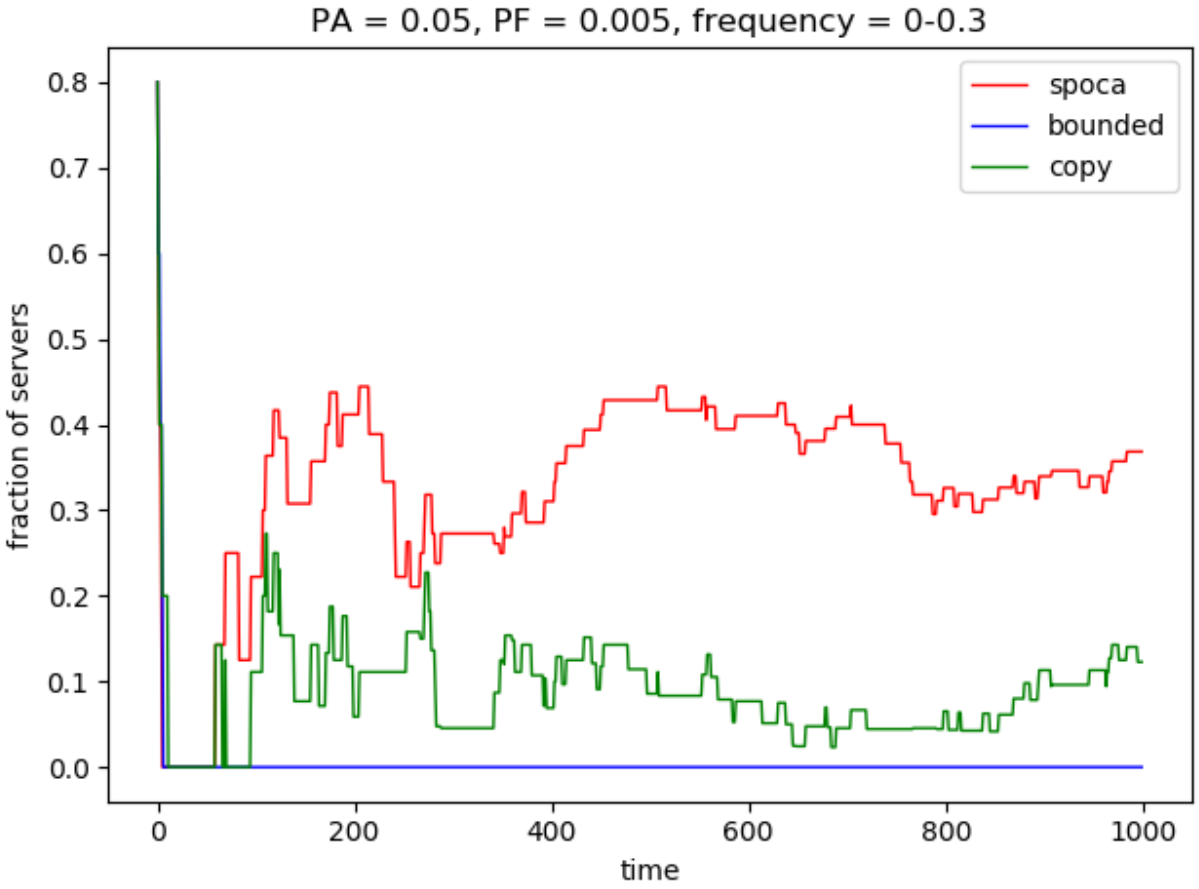

PA = 0.1, PF = 0.005, frequency = 1-1.3
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

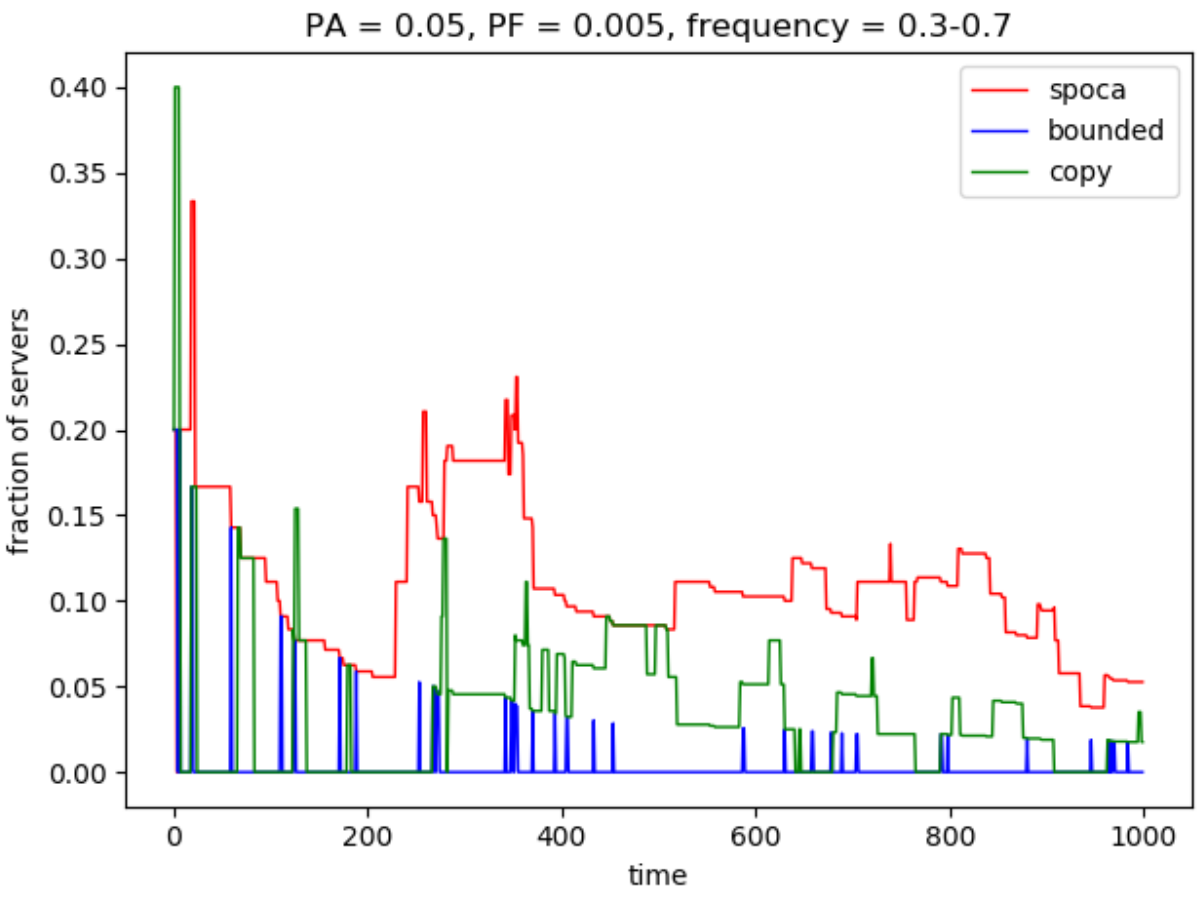

PA = 0.1, PF = 0.005, frequency = 1.3+
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

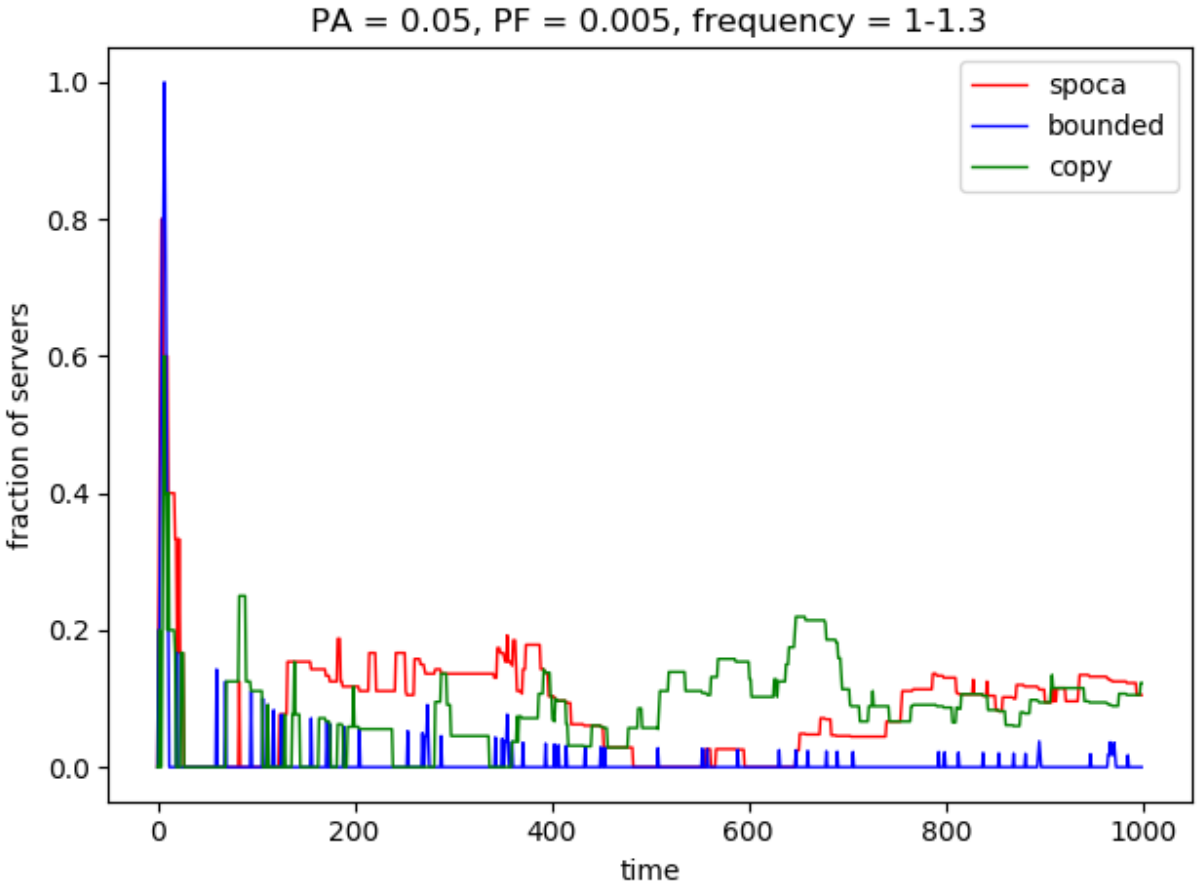

PA = 0.05, PF = 0.01, frequency = 0-0.3
spoca
bounded
copy
fraction of servers
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000
PA = 0.05, PF = 0.01, frequency = 0.3-0.7
spoca
bounded
copy
fraction of servers
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000
time

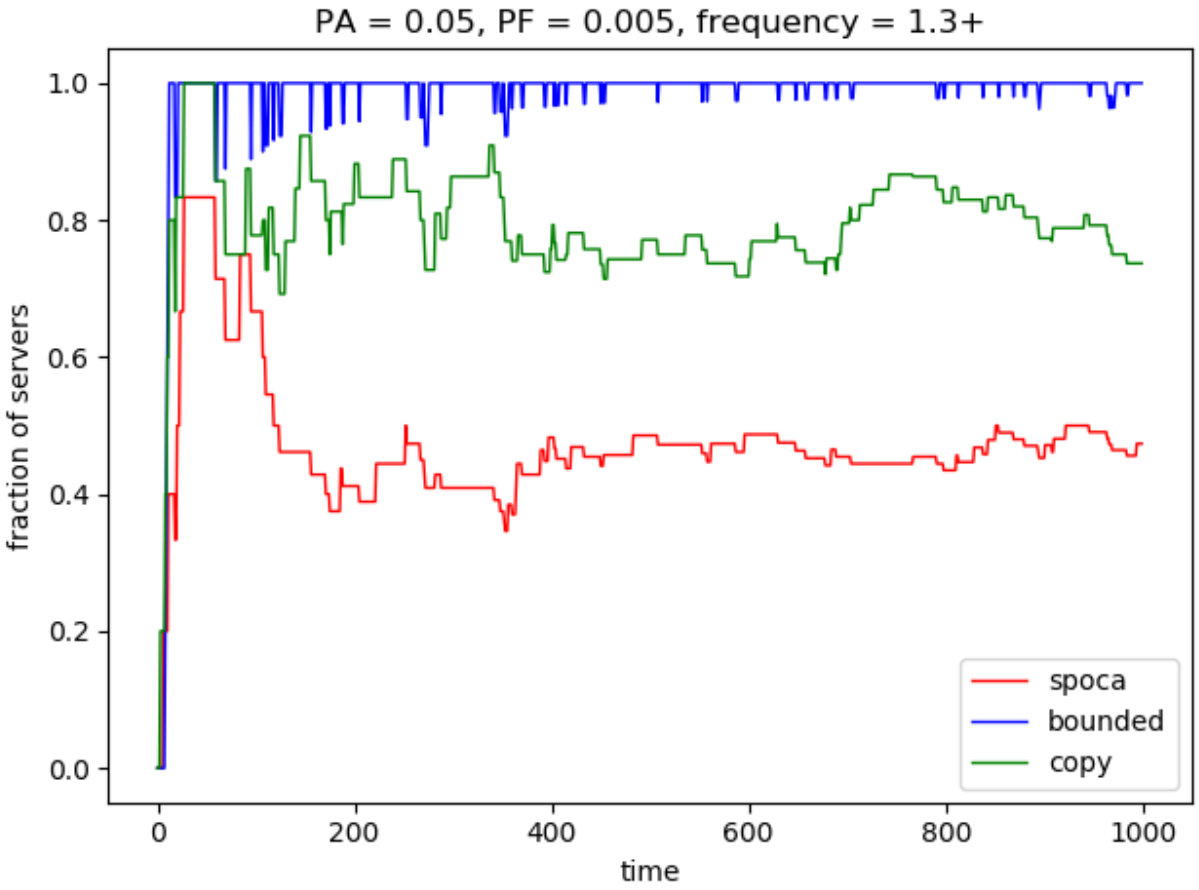

PA = 0.05, PF = 0.01, frequency = 1-1.3
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

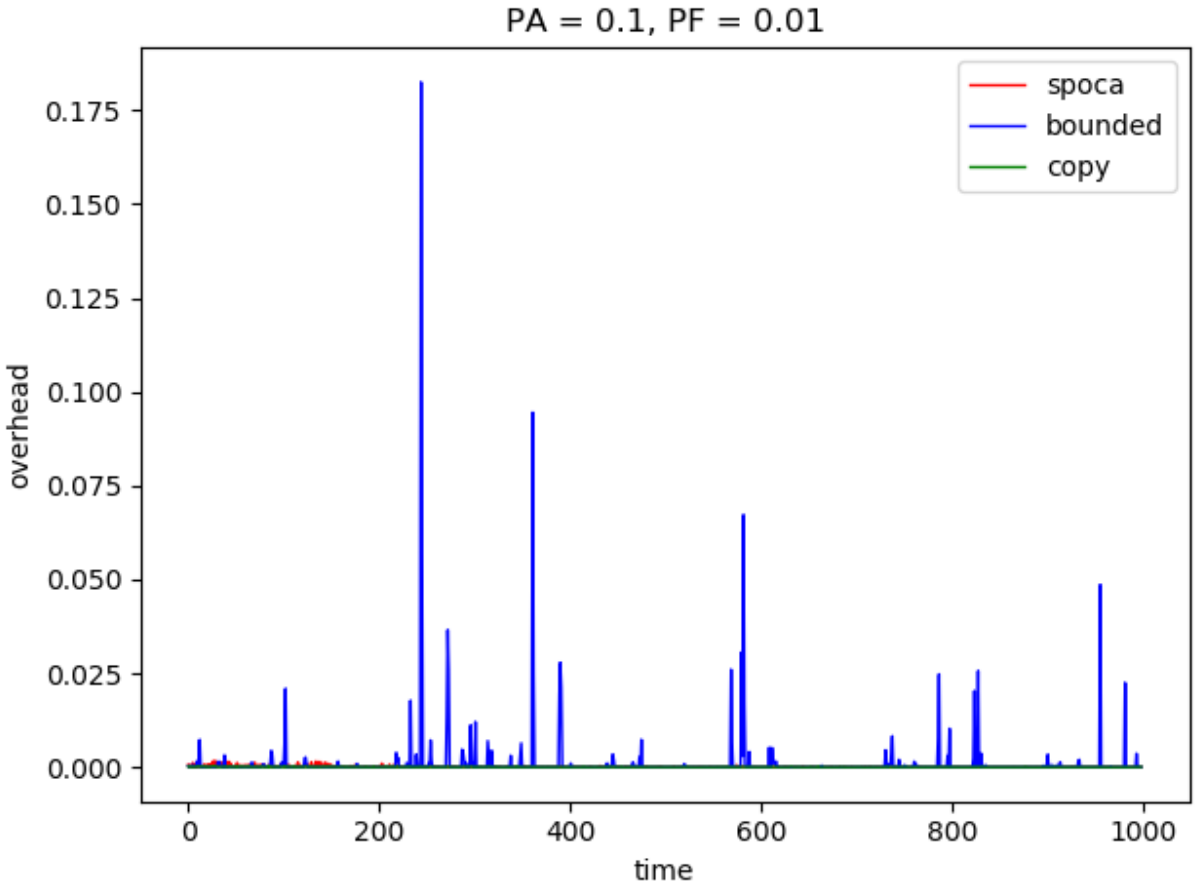

PA = 0.05, PF = 0.01, frequency = 1.3+
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

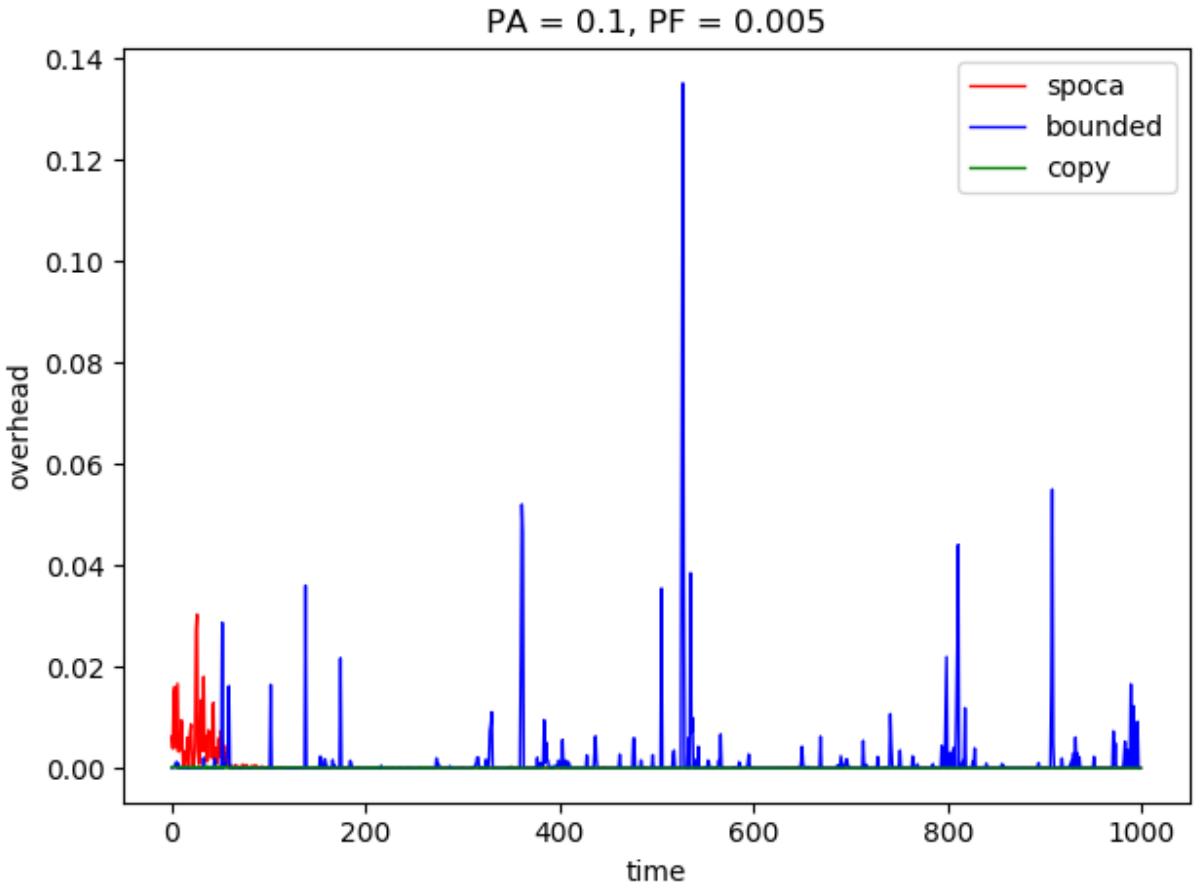

PA = 0.05, PF = 0.005, frequency = 0-0.3
spoca
bounded
copy
fraction of servers
time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
200
400
600
800
1000

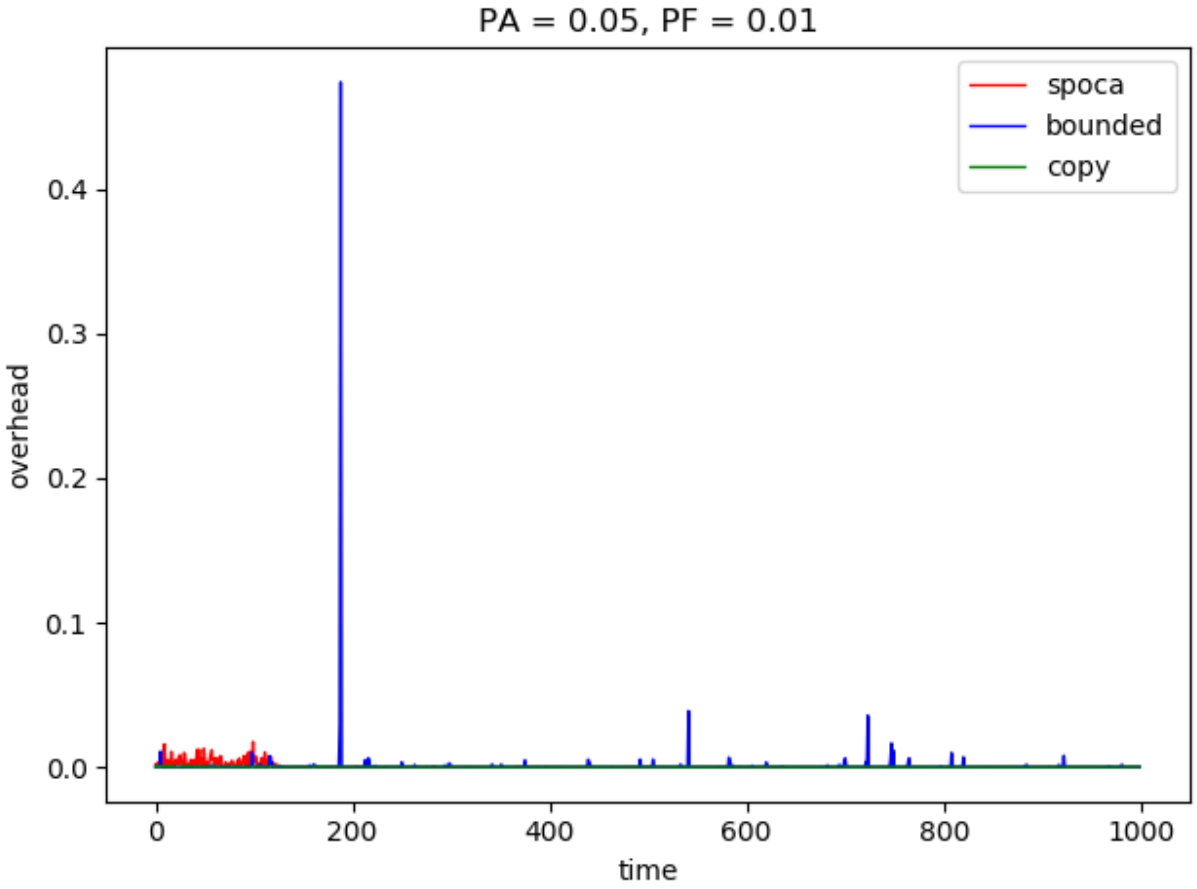

PA = 0.05, PF = 0.005, frequency = 0.3-0.7
spoca
bounded
copy
fraction of servers
time
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
200
400
600
800
1000

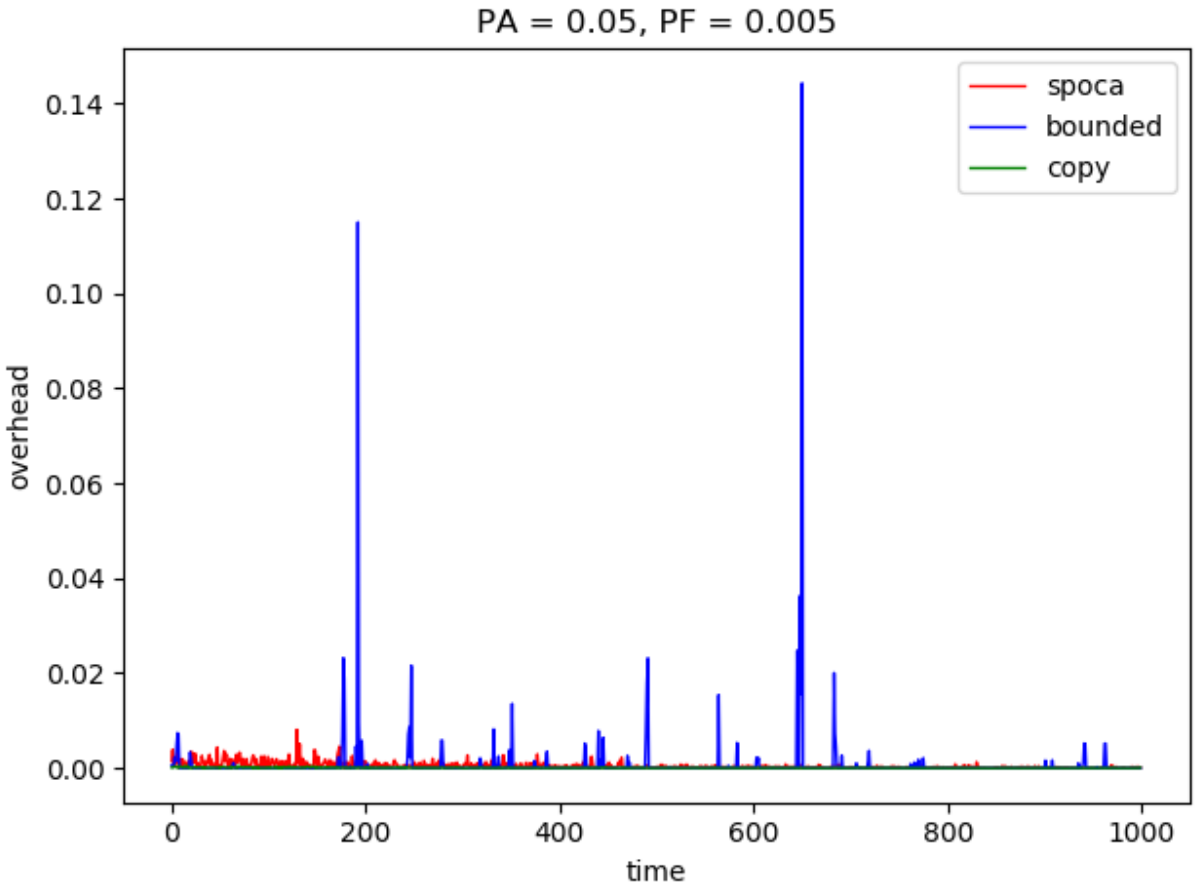

PA = 0.05, PF = 0.005, frequency = 1-1.3
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

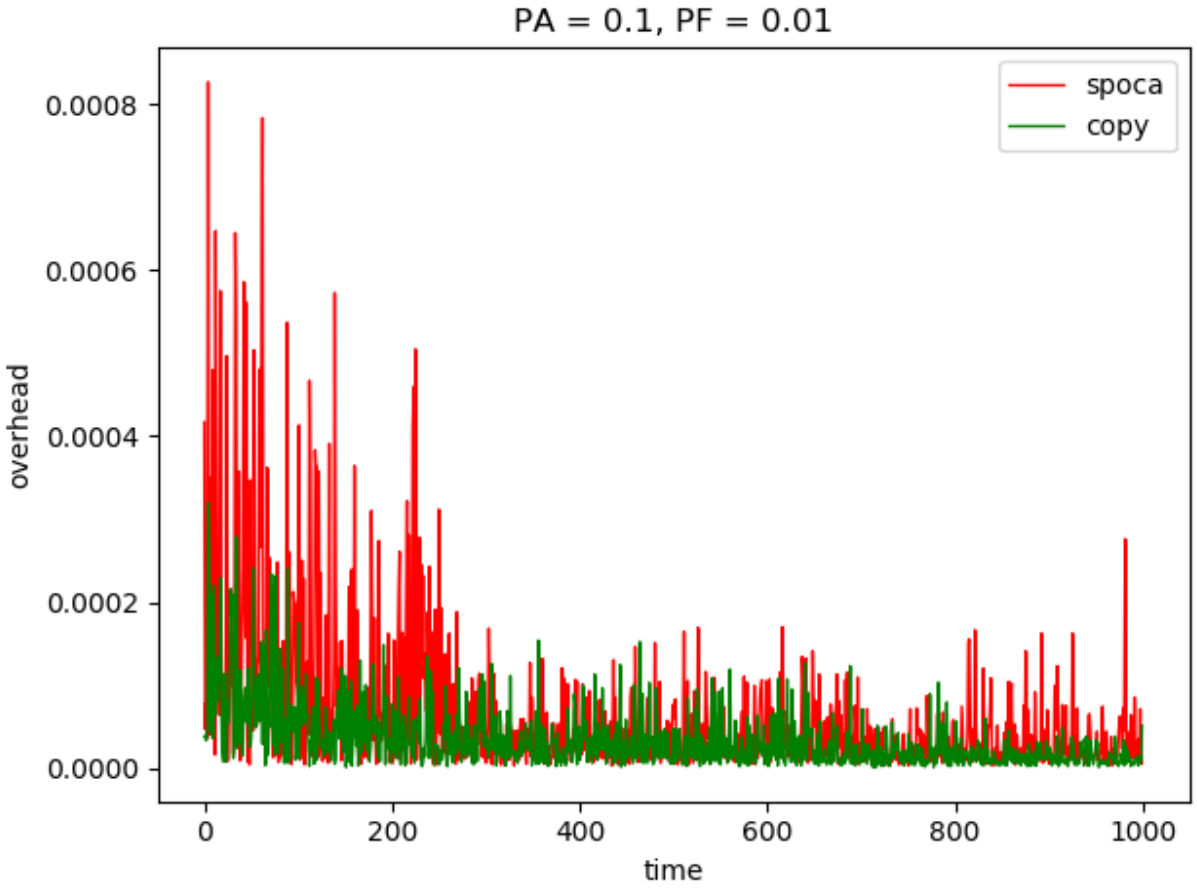

PA = 0.05, PF = 0.005, frequency = 1.3+
spoca
bounded
copy
fraction of servers
time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

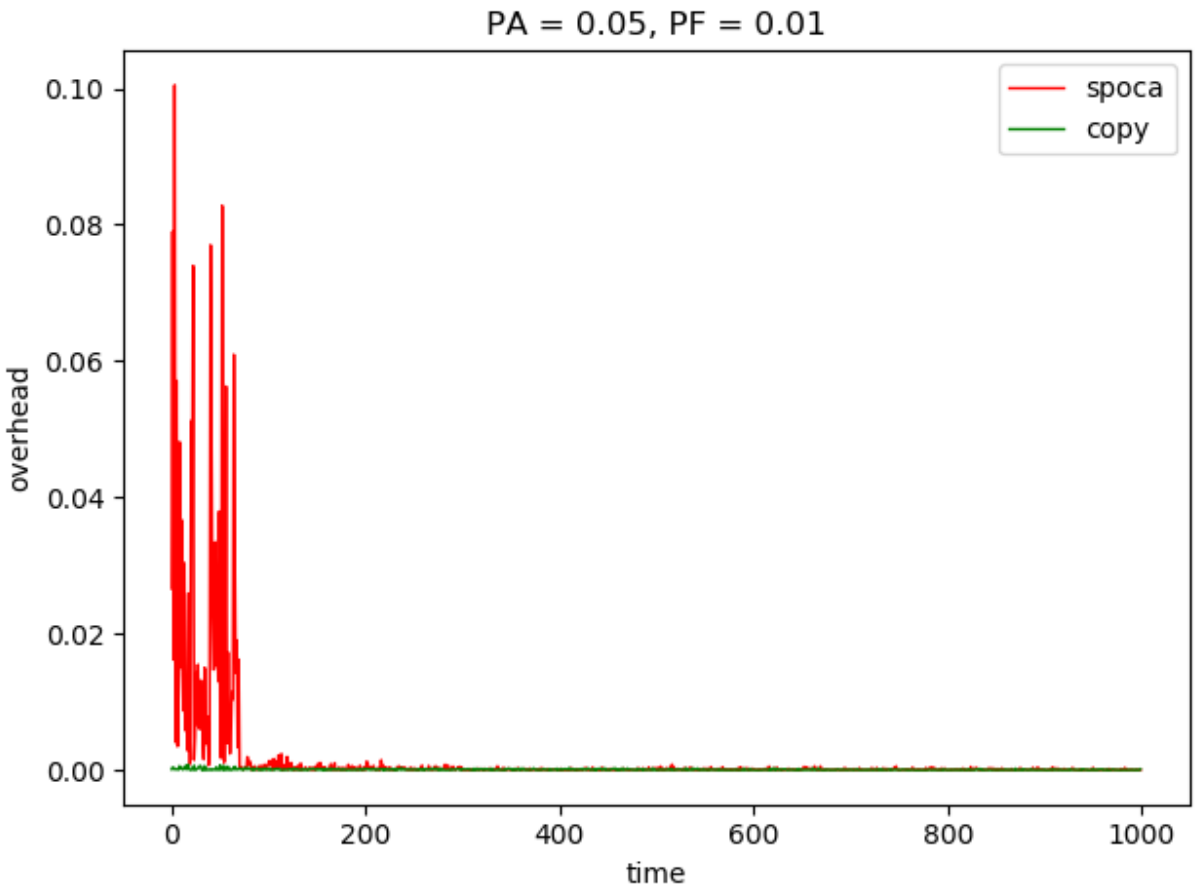

PA = 0.1, PF = 0.01
spoca
bounded
copy
overhead
time
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
200
400
600
800
1000

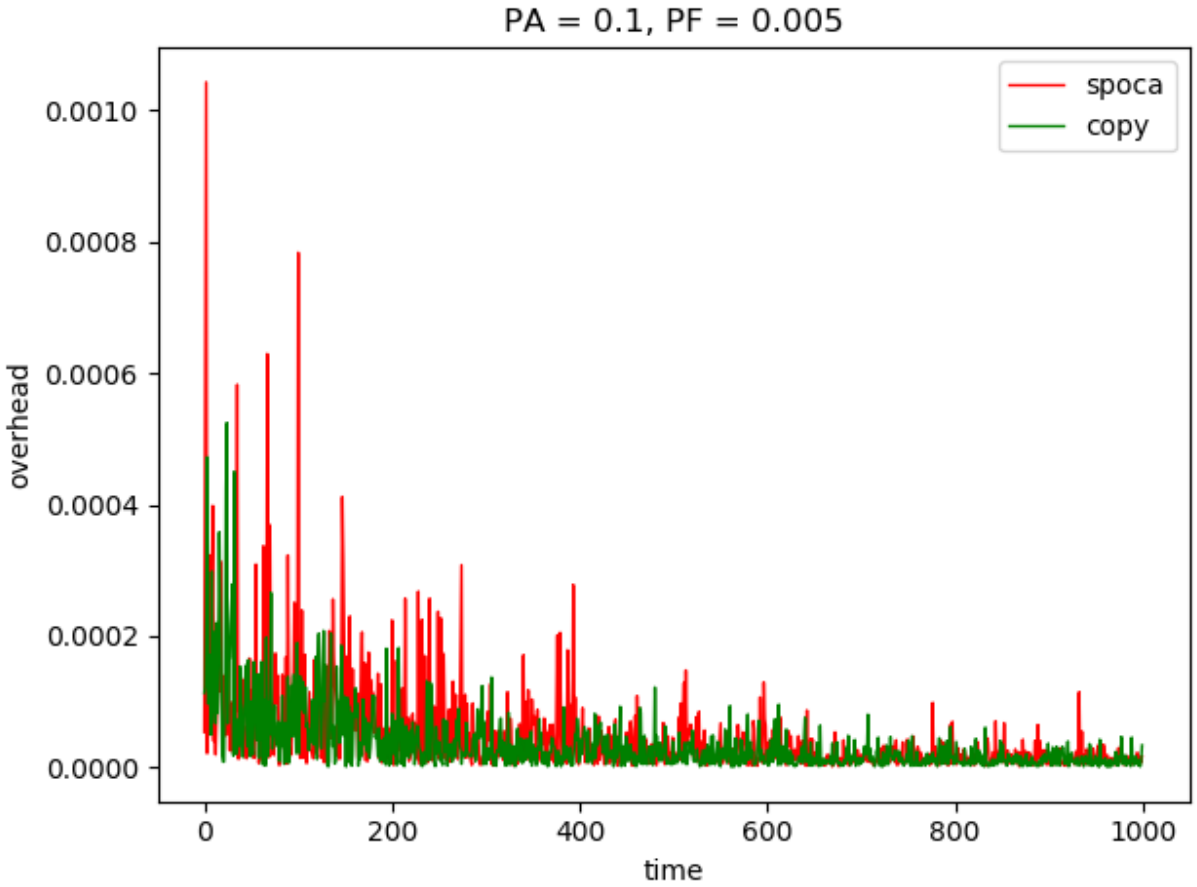

PA = 0.1, PF = 0.005
spoca
bounded
copy
overhead
time
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
200
400
600
800
1000

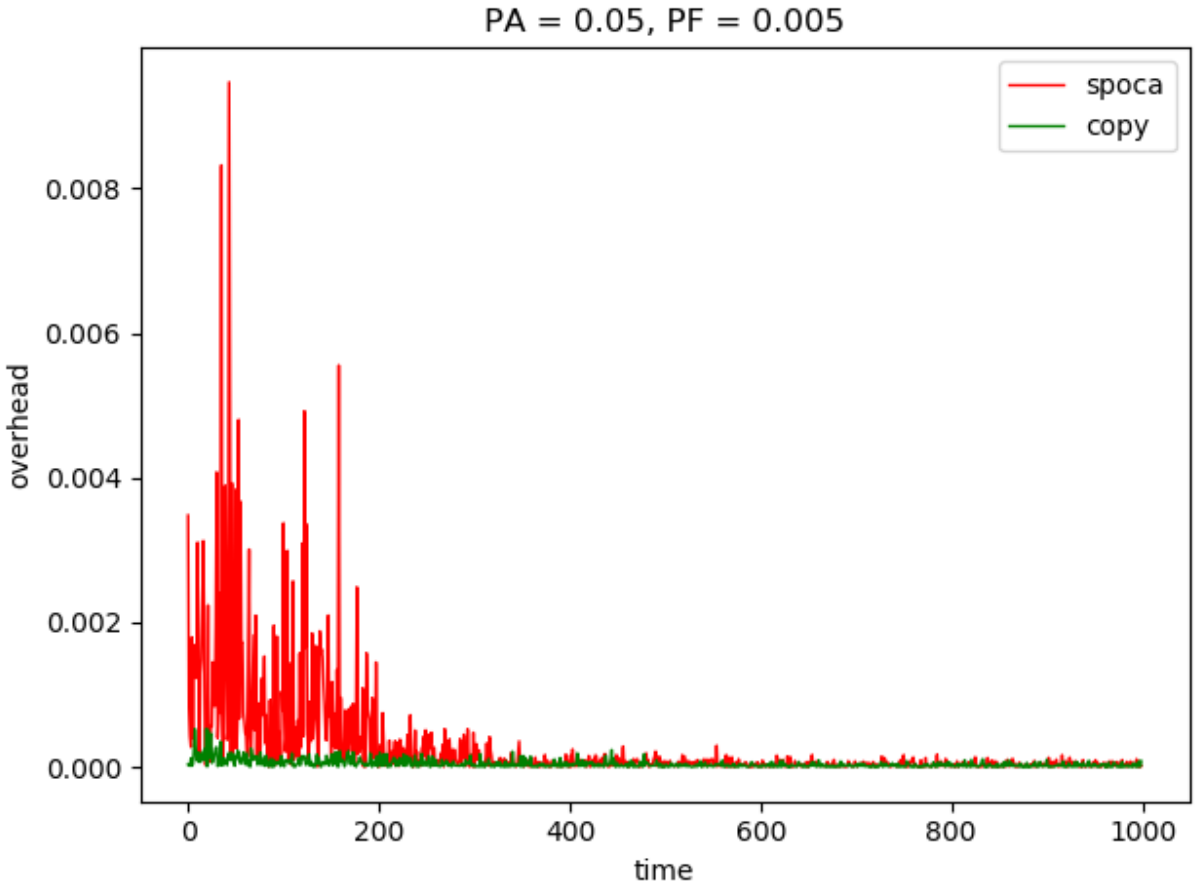

PA = 0.05, PF = 0.01
spoca
bounded
copy
overhead
time
0.0
0.1
0.2
0.3
0.4
0
200
400
600
800
1000

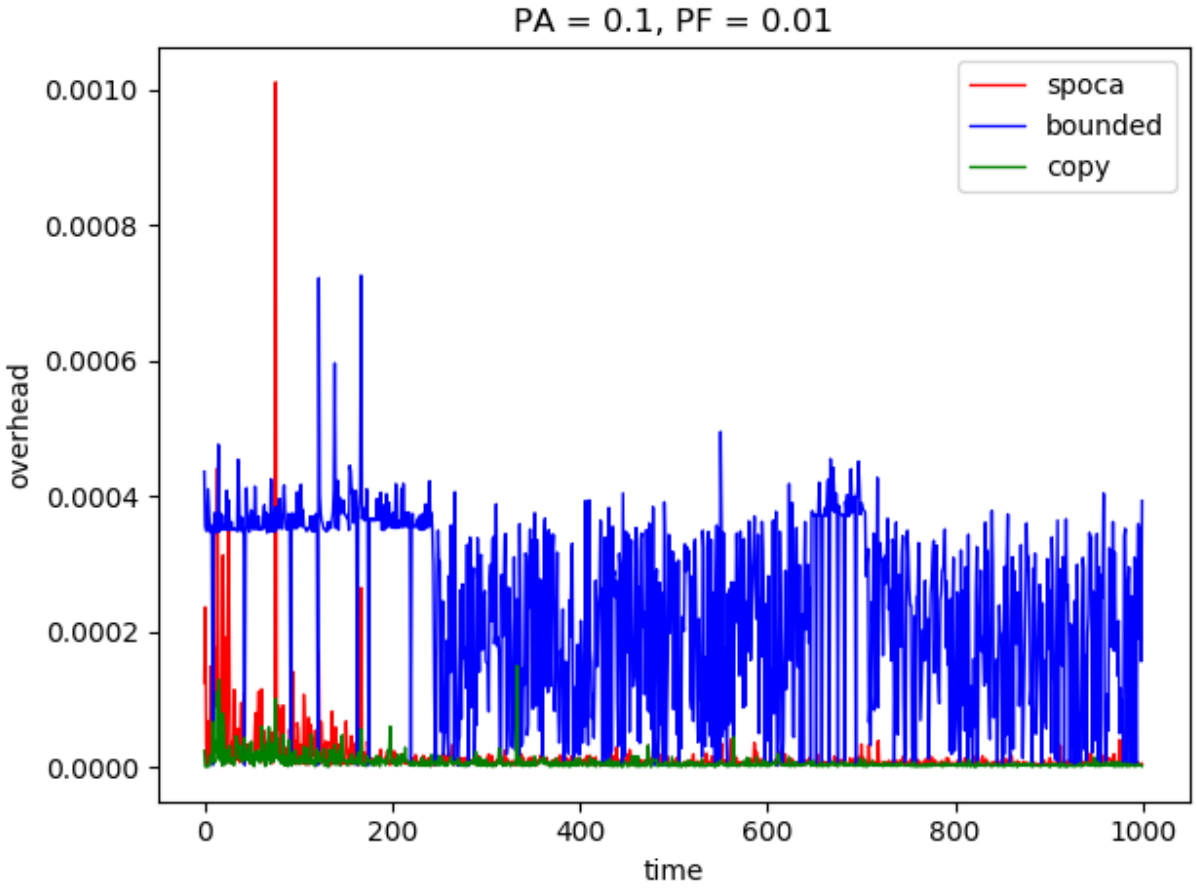

PA = 0.05, PF = 0.005
spoca
bounded
copy
overhead
time
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
200
400
600
800
1000

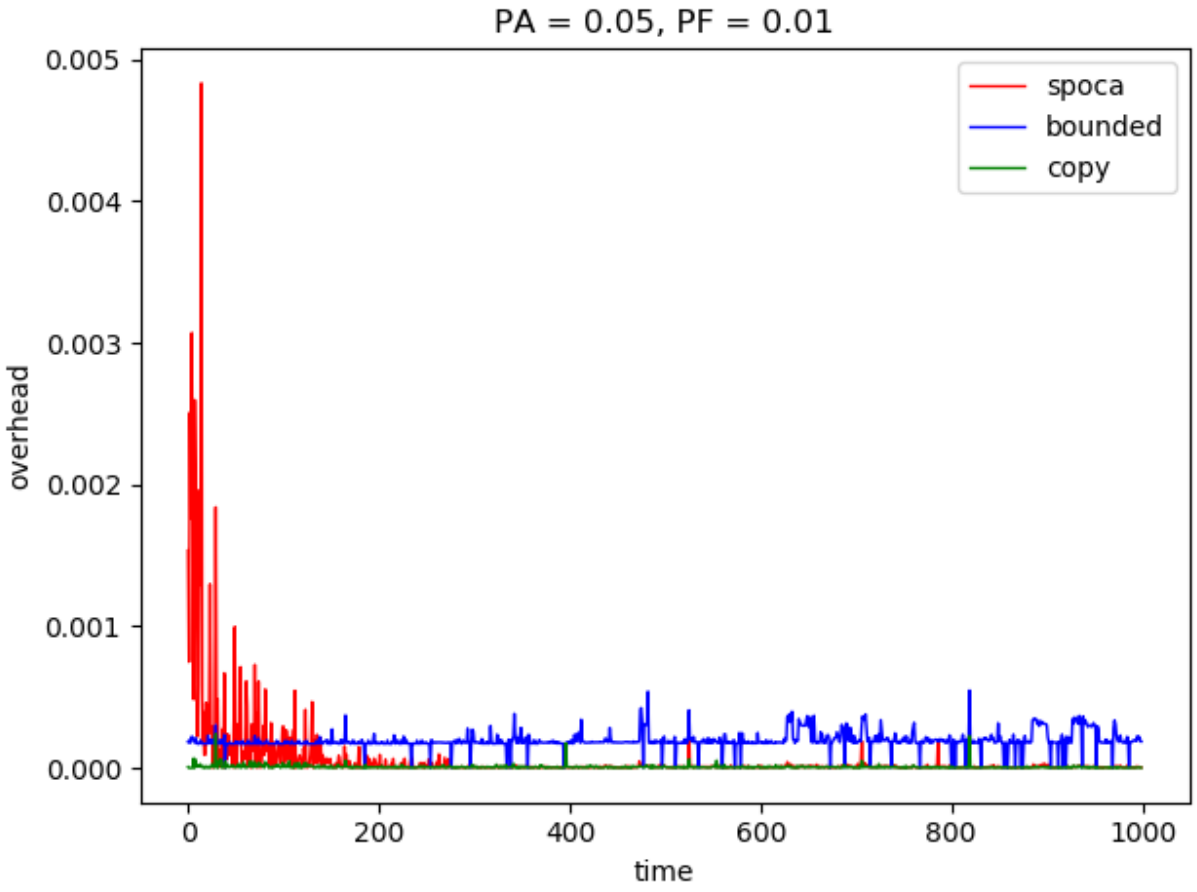

PA = 0.1, PF = 0.01
spoca
copy
overhead
0.0000
0.0002
0.0004
0.0006
0.0008
0
200
400
600
800
1000
time

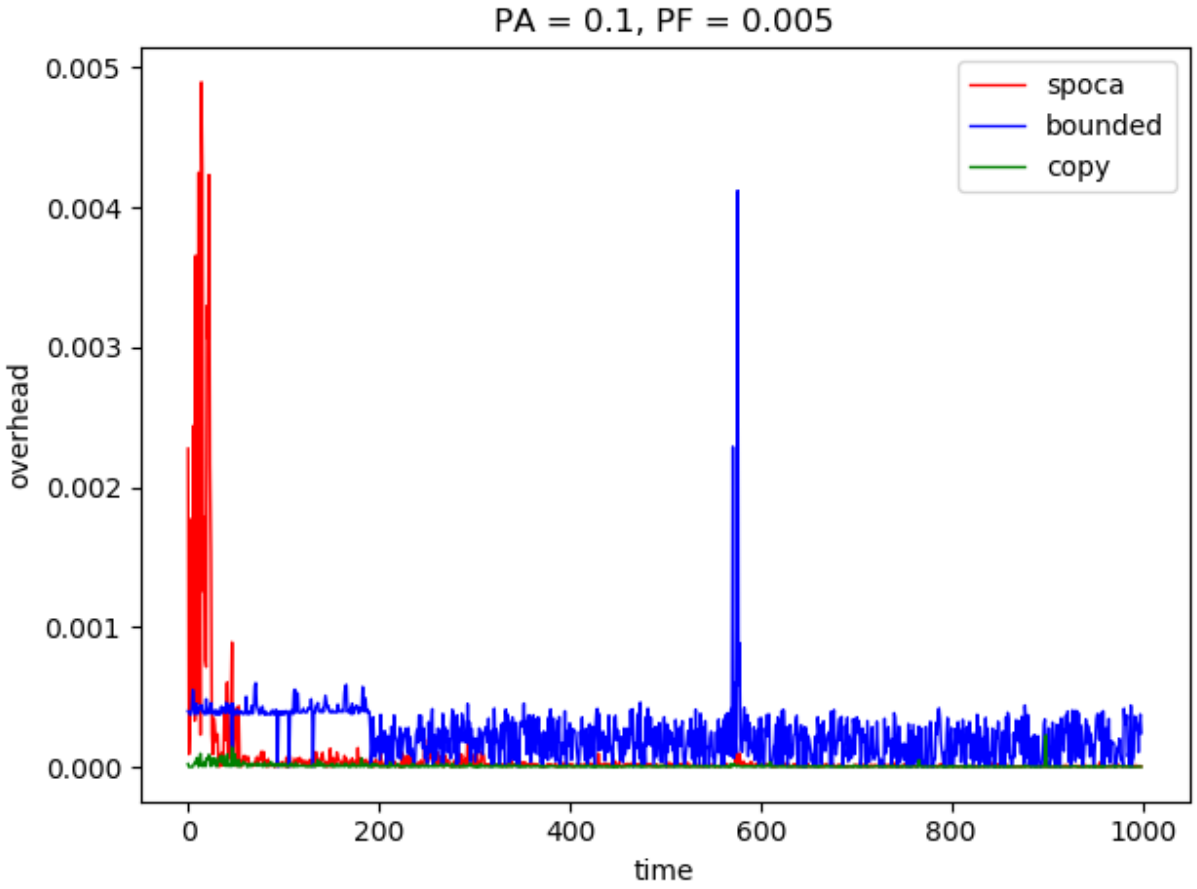

PA = 0.05, PF = 0.01
spoca
copy
overhead
0.00
0.02
0.04
0.06
0.08
0.10
0
200
400
600
800
1000
time

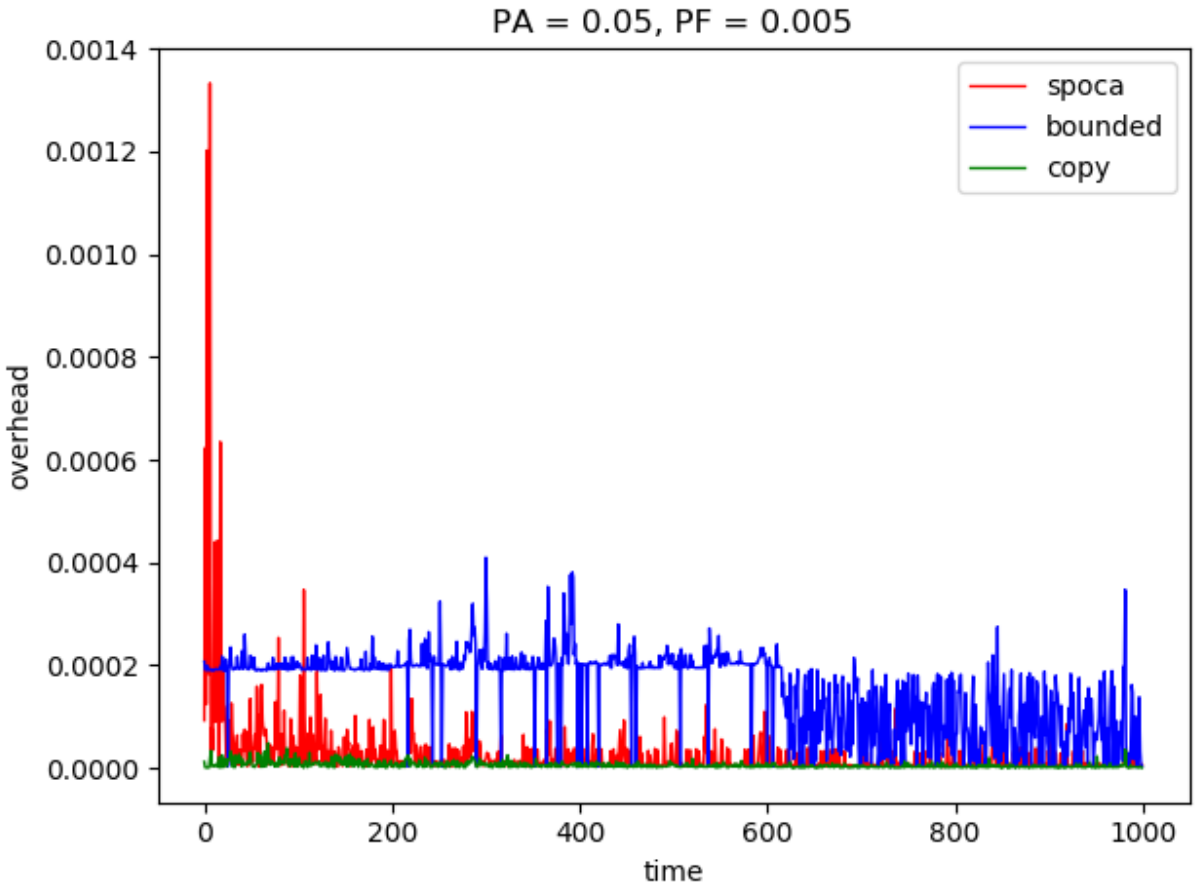

PA = 0.1, PF = 0.005
spoca
copy
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
overhead
0
200
400
600
800
1000
time


PA = 0.05, PF = 0.005
spoca
copy
0.008
0.006
0.004
0.002
0.000
overhead
0
200
400
600
800
1000
time


PA = 0.1, PF = 0.01
spoca
bounded
copy
overhead
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
200
400
600
800
1000
time


PA = 0.05, PF = 0.01
spoca
bounded
copy
overhead
0.005
0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000
time


PA = 0.1, PF = 0.005
spoca
bounded
copy
overhead
time
0.005
0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000


PA = 0.05, PF = 0.005
spoca
bounded
copy
overhead
time
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
200
400
600
800
1000

Comparison Report:

From the above plots, we can see that SPOCA has the worst load balancing. After 1000 iterations, we still have about 30% servers has load 0-0.3 of their capacity for SPOCA which is the highest among these three methods. Meanwhile, SPOCA has only 50% servers overloaded which is the lowest. This result indicates that the loads are not balanced in SPOCA and we have many spare spaces while lots of the servers are already overloaded. For Consistent Hashing with multiple copies, we have about 20% servers has load 0-0.3 of their capacity and about 75% servers overloaded. For Consistent Hashing with Bounded Loads, almost all the servers are overloaded which means the loads are very balanced. This is because in the implementation, we will only add to the servers that are not overloaded.

For the overhead, Consistent Hashing with Bounded Loads is the slowest. This is because when all the servers are loaded, this method has to find the server with the least overflow. For the SPOCA, it is pretty slow at first since we cannot find a server easily if they are sparse. However, as we have more servers, SPOCA will have less overhead compared to Consistent Hashing with Bounded Loads. For Consistent Hashing with multiple copies, it is very fast, way more faster then SPOCA and Consistent Hashing with Bounded Loads. This is because we stored server into multiple places so it is easier to find a server to assign the task.