Name of the Student: Tolga Talha Yıldız

Name of the Company: Bilkent University

Address of the Company: Üniversiteler, 06800 Çankaya/Ankara, Turkey

Dates of Summer Practice: 04.08.2020-08.10.2020

Bilkent University
Faculty of Engineering

Contents

1 Introduction

I did my internship with Nail Akar on the research area of Age of Information. I was fortunate enough to work with him during my internship period. First week of the internship consisted of choosing what to study for Internship from the given sources. I read literature surveys [1],[2] and recent research papers [3],[4],[5],[6] about given topics which were Age of Information and Server Provisioning [7] related. Area of work then chosen to be Age of Information.

It is important to evaluate systems which depends on the status updates from the peripheral devices and networks of sensors because the low latency cyberphysical system application growing in time, from this need Age of Information emerges and inspects the timeliness of status updates.

My internship consisted of learning these new concepts, understanding the theory behind the papers, simulation of the constructed systems in the papers and reproducing the results in these papers. Paper in matter was "The Multi-Source Probabillistically Preemptive M/PH/1/1 Queue with Packet Errors".[2]. I first study the previously published "Finding the Exact Distribution of (Peak) Age of Information for Queues of PH/PH/1/1 and M/PH/1/2 Type"[3], and gained insight about the queue systems and simulations of them.

During this internship I learned concepts of Markov Chains, Queue Systems, Age of Information, PH-Type Distributions, Simulation Methods, Python Simulation Frameworks.

2 Products and Production Systems

Nail Akar is one of the professor in Bilkent University Electrical and Electronics Engineering Department and his research interests are performance evaluation of computer and communication networks, future Internet, wireless and optical networks, queuing systems and Markov Chains. He recently published papers about Markov Fluid Queues, Age of Information Queue Models and its analysis. He is teaching the courses EEE212 Microprocessors and EEE533 Random Processes in Bilkent University.

3 Work Done

I worked on one project, which was the reproduction of the results gathered in the research paper "The Multi-Source Probabilistically Preemptive M/PH/1/1 Queue with Packet Errors"[2]. In this paper we have a server which gathers information from 3 sources with different rates and different preemption schemes. Each source produces packets with poisson process which means the interarrival times of these packets are distributed according to exponential random variables. Packets are processed without any difference between the sources with the 0.1 probability of packet errors with retransmission probabilities set to 1. Goal was to simulate such a server queue system with no buffers for incoming packets.

3.1 Literature Survey

I read papers about the Age of Information which helped me to understand the concept and how it was developed through the years, how it was emerged [1],[2]. I also read some interesting papers on the Age of Information, which were discussing the protocol implementation of a such concept that will run on the network [3]. After literature surveys I decided to simulate the network system described in the [4]. Before implementing a multi-source system, I implemented a single source, rather simple network described in the [5].

3.2 Primer Work to the Papers

3.2.1 Markov Chains and Queue Systems

I learned Markov Chains and Queue Systems by accessing course materials of the IE325 Stochastic Models [8] and Wikipedia page [9]. Then I learned the implementation of them in python with discrete event simulation framework SimPy [10]. Followed SimPy tutorials given in the documentation page for SimPy [10].

3.2.2 PH-type Distributions and Tools

I learned PH-type distributions from a YouTube video [11] and it was interesting to see another approach to distribution functions, I loved how it can be so dynamic and yet can be expressed in a simple form. I used a tool for generating samples from such distribution, which was called butools [12]. I used this tools library in Python.

3.3 Simulations

Simulations can be classified under two main type: Continuous Time Simulations and Discrete Time Simulations. For our purposes we will focus on discrete time simulations and these can be implemented in two ways: Event based simulations, time based simulations.

3.3.1 Event Based Simulations

In the event based simulations simulation time is decided with the arrival times of the events and not by a unit time increment. This type of simulation is much faster but harder to implement because during the service process of the packets there can be an preemption or a packet failure which will change the event type between two simulation time. This requires interrupt handling, which is possible but harder to implement. In this code flow there needs to be interrupt handlers in case of a packet arrival during the processing of another packet and deciding whether preempt the packet in the process or discard the received packet. Also there needs to be some event cycle handlers for packet errors. I could not implement these dynamics but I achieved to make a simulation where there were no preemption or packet errors, which was the queue system of M/PH/1/1. I used a framework called SimPy, which utilizes 'yield' statement of Python and creates a timeout event for the packet in the server. Code is in the appendix A.

I did not used this code to produce any data for the reproduction of the plots in the paper. However this was a nice introduction to the discrete event frameworks in Python. Also this was the first usage of the 'PH distribution tools' which were taken from butools library. Upon failing to recreate a multi source preemptive network with packet errors I switched to time based simulations which enables us to control every millisecond of the simulation.

3.3.2 Time Based Simulations

In time based simulations, simulation times are incremented by unit time, this is easier to implement but very inefficient in the means of running time since in each time step we control whether any new packet received or not and decide to preempt or not. In each simulation I created poisson arrival times by using exponential variable as the interarrival time for the packets of a source, then created the arrival times from this data then sorted across the time. I also generated the service times using the butools library which lets us take samples from a PH-type distribution. Then implemented a simulation system which defines different probabilistic schemes for the given input and then simulates the system. Code for this simulation is provided in the appendix B.

I used this code to generate data sets for the Multi source preemptive bufferless network with packet errors, simulation can be done at any length, any number and any time resolution. In the code I can tweak the scov variance, number of sources, source rate vector, probability scheme for preemption. For the paper I used these variables as follows, 3 sources with 3 probability schemes: global preemption, self preemption, prioritized preemption. There were two modes for scov variance: 0.25 and 0.5. I recorded the results of the simulations in a .txt folder.

3.4 Visualization

I used MATLAB to get a final plot for the comparion. I requested the data for the plots in the paper and then combined these two in the MATLAB with the script in the appendix C. This MATLAB script works only for one plot. To get the other plots imported data file references must be changed. Results are give below, there is a pretty good match between the plots of Global Preemption and Self Preemption but Prioritized Preemption does not follow the exact same behavior even though the general behavior of sources still the same.

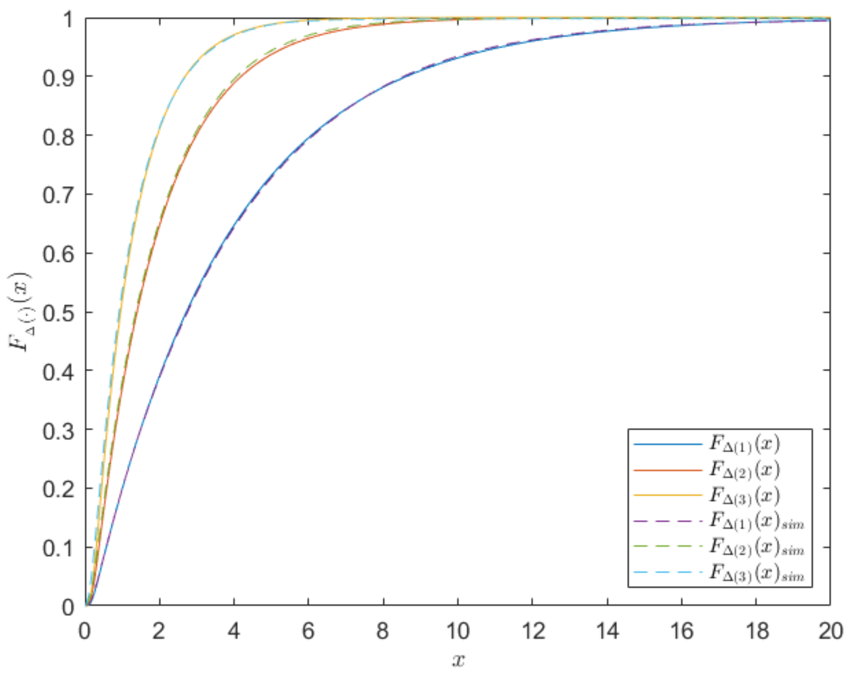

Figure 1: CDF of Age of Information process when global preemption scheme is selected and $c_\Theta^2 = 1/4$

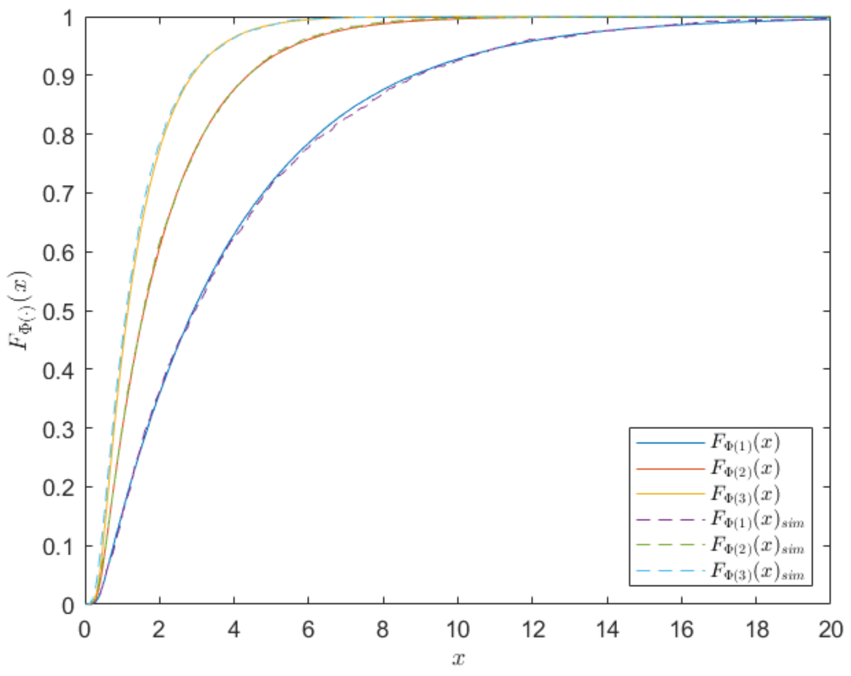

Figure 2: CDF of Peak Age of Information process when global preemption scheme is selected and $c_\Theta^2 = 1/4$

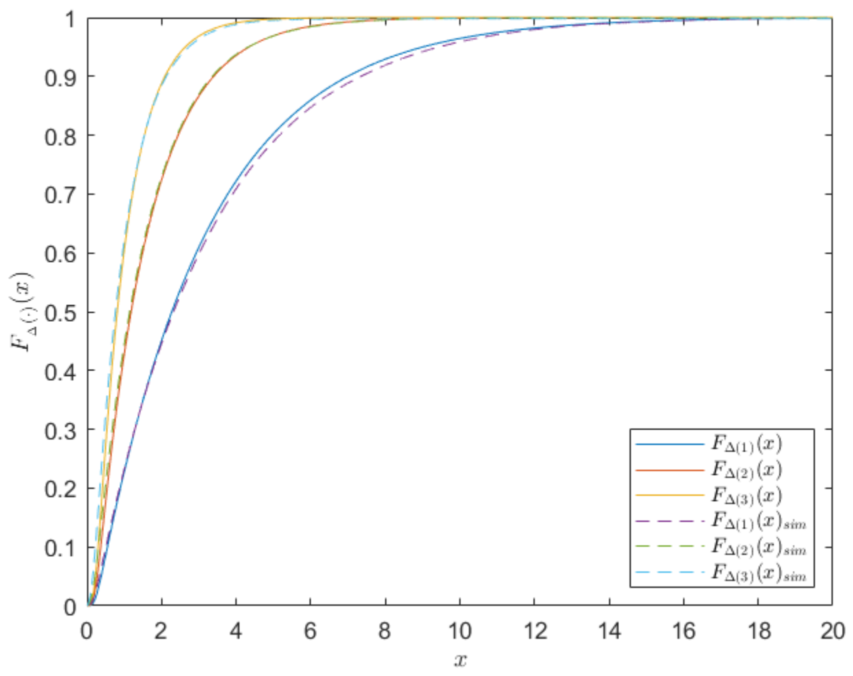

Figure 3: CDF of Age of Information process when self preemption scheme is selected and $c_{\Theta}^{2} = 1/4$

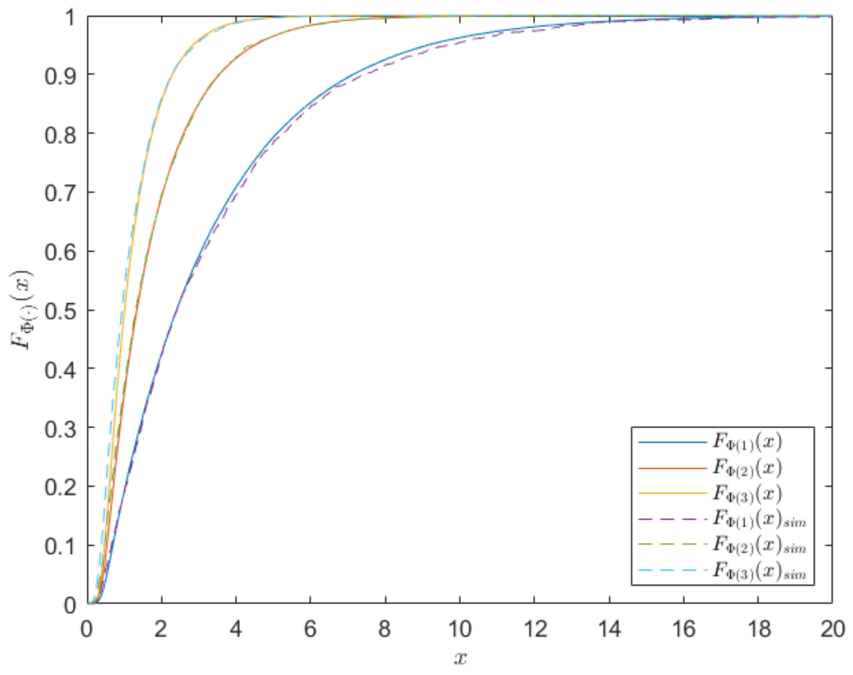

Figure 4: CDF of Peak Age of Information process when self preemption scheme is selected and $c_{\Theta}^{2} = 1/4$

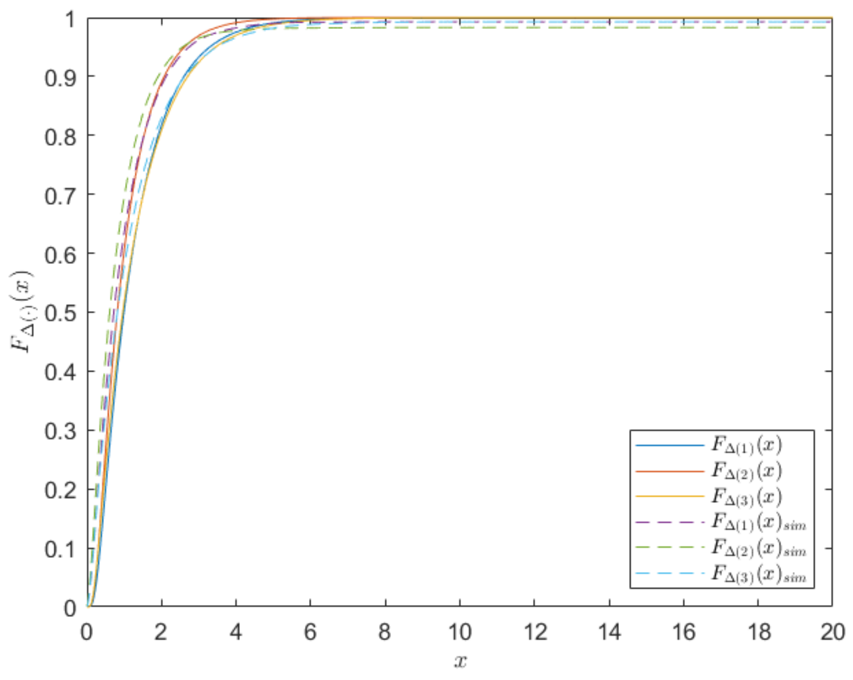

Figure 5: CDF of Age of Information process when prioritized preemption scheme is selected and $c_\Theta^2 = 1/4$

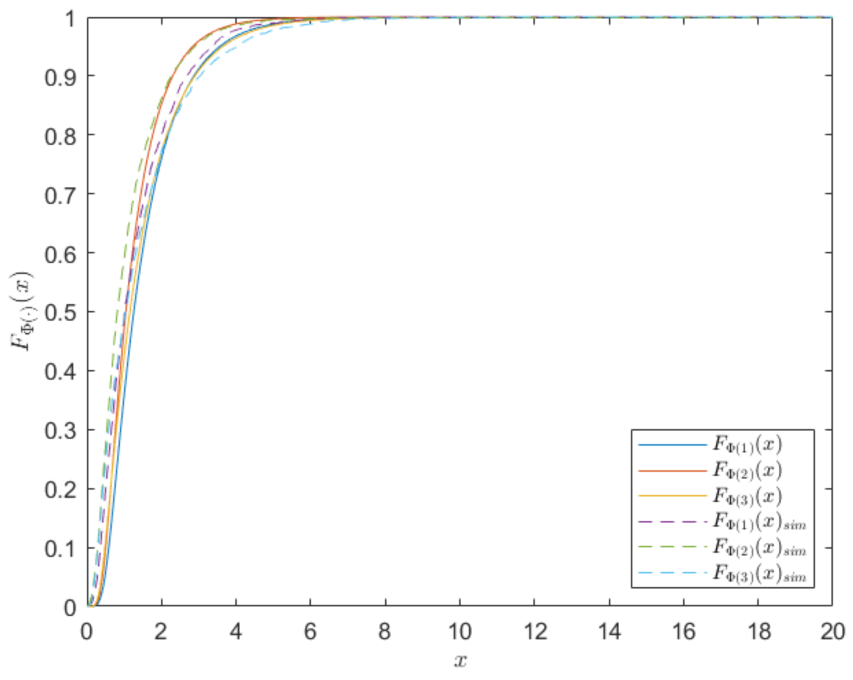

Figure 6: CDF of Peak Age of Information process when prioritized preemption scheme is selected and $c_\Theta^2 = 1/4$

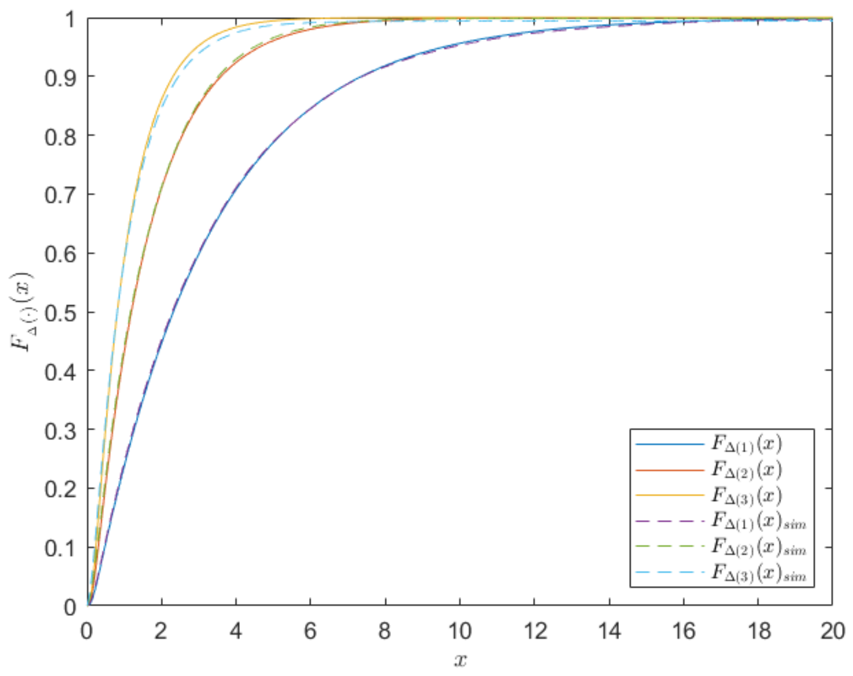

Figure 7: CDF of Age of Information process when global preemption scheme is selected and $c_\Theta^2 = 1/2$

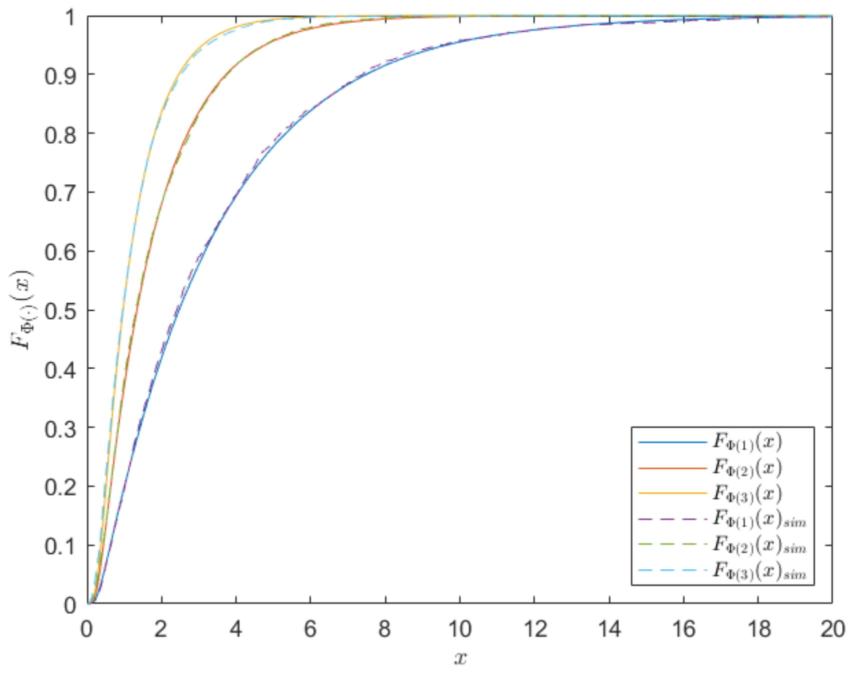

Figure 8: CDF of Peak Age of Information process when global preemption scheme is selected and $c_\Theta^2 = 1/2$

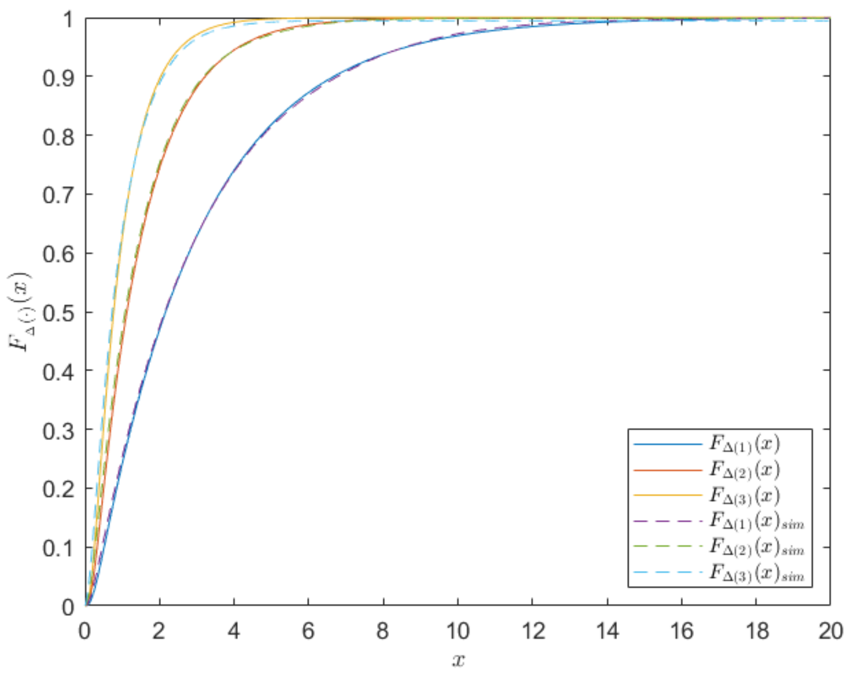

Figure 9: CDF of Age of Information process when self preemption scheme is selected and $c_\Theta^2 = 1/2$

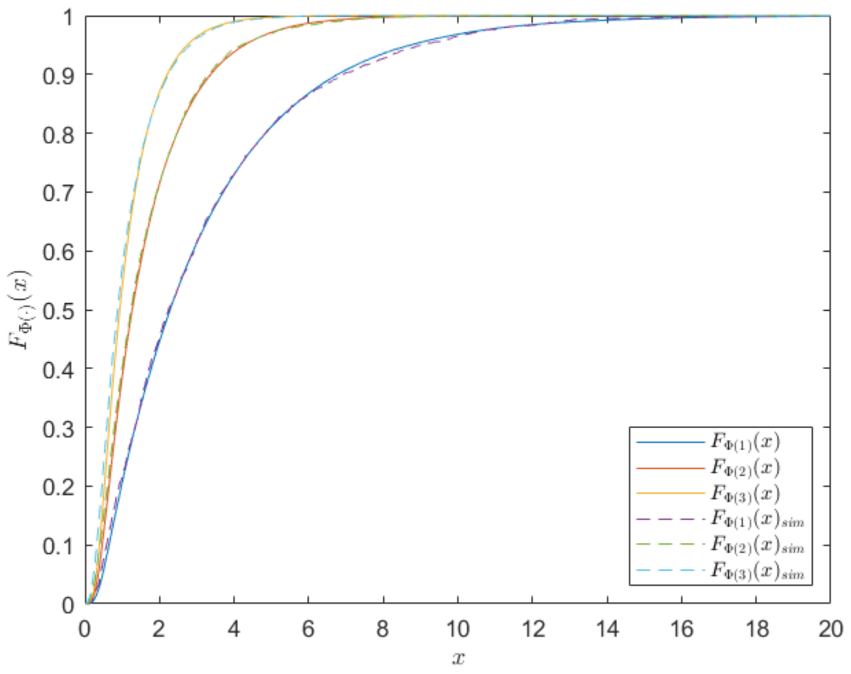

Figure 10: CDF of Peak Age of Information process when self preemption scheme is selected and $c_\Theta^2 = 1/2$

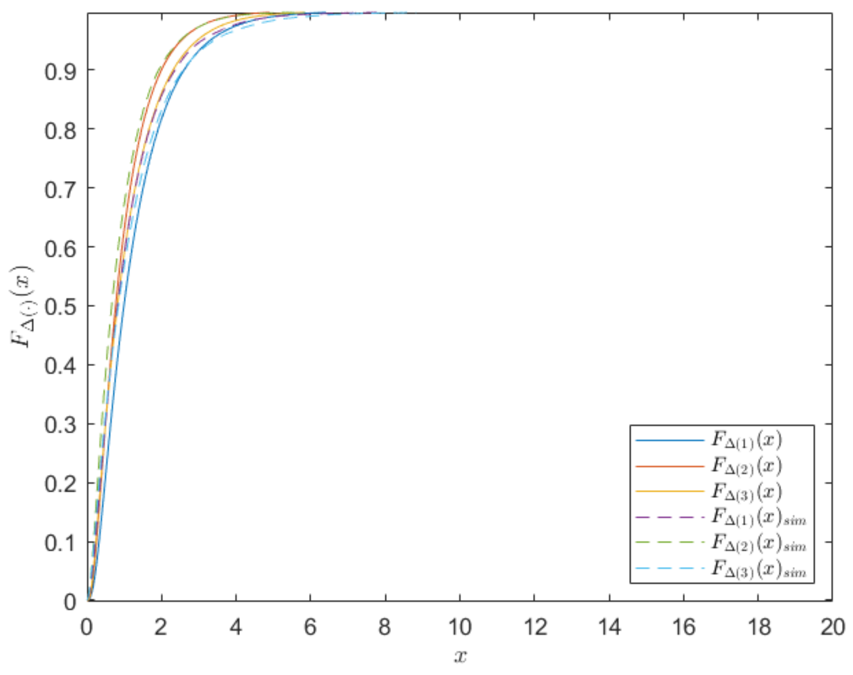

Figure 11: CDF of Age of Information process when prioritized preemption scheme is selected and $c_{\Theta}^2 = 1/2$

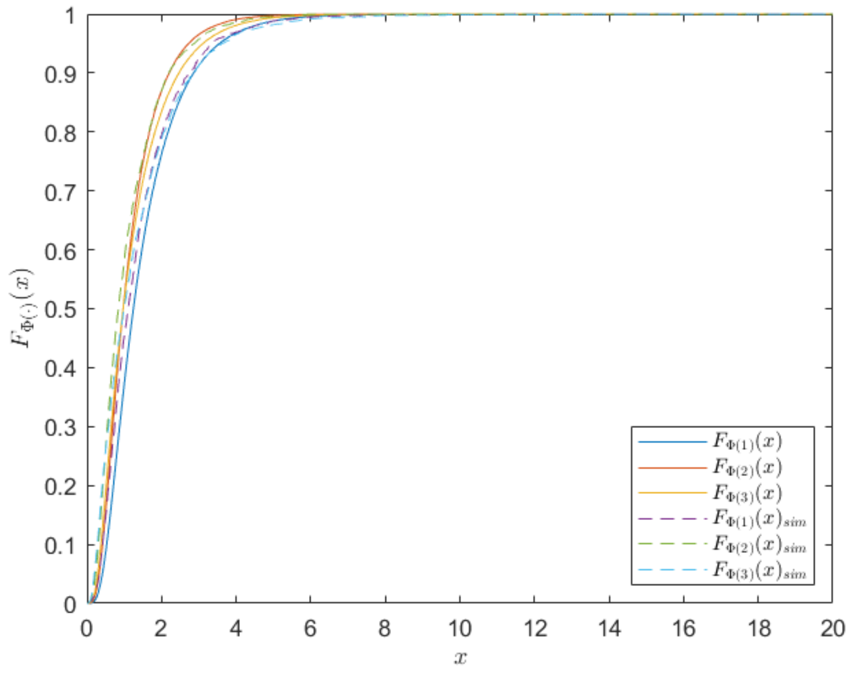

Figure 12: CDF of Peak Age of Information process when prioritized preemption scheme is selected and $c_{\Theta}^2 = 1/2$

4 Conclusion

Work I did in the internship is heavily related to academic work and it is essential to know how to simulate a system and gather statistical data from this simulation to confirm in the analysis in the paper. Even though I could not get perfect simulation results for one type of simulation, namely prioritized preemption, it was satisfying to see the behavior of the simulation was matching the analysis done in the paper.

I got to see the academic work done in Bilkent University and got familiar with the research content. I studied in a field I would not know otherwise if I was not advised to study these concepts. Internship this being online was weird because there is no workspace dedicated for such work and everyone is trying to get a project done in the given time in their home with what they can get their hands on online.

5 References

[1] R. D. Yates, Y. Sun and D. R. Brown, "Age of Information: An Introduction and Survey," *arXiv:200708564[cs]*, July 2020.

[2] A. Kosta, N. Pappas and V. Angelakis, "Age of Information: A New Concept, Metric and Tool," *Foundations and Trends in Networking,* vol. 12, no.3, Nov. 2017, pp. 162-259.

[3] T. Shreedbar, S. K. Kaul and R. D. Yates, "An Age Control Protocol for Delivering Fresh Updates in the Internet-of-Things," *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, June. 2019.

[4] O. Doğan and N. Akar, "The Multi-Source Probabilistically Preemptive M/PH/1/1 Queue with Packet Errors," *arXiv:2007.11656[cs]*, July 2020.

[5] N. Akar, O. Doğan and E. Ü. Atay, "Finding the Exact Distribution of (Peak) Age of Information for Queues of PH/PH/1/1 and M/PH/1/2 Type," *arXiv:1911.07274 [cs]*, July 2020.

[6] M. H. Mamhudi, J. P. Champati and J. Gross, "Where Freshness Matters in the Control Loop: Mixed Age-of-Information and Event-based Co-design for Multi-loop Networked Control Systems," *Architecture and Protocols for Wireless Sensor Networks and Actuator Networks*, June 2020.

[7] B. Bartan and N. Akar, "Energy-Efficient Dynamic Server Provisioning in Server Farms," May 2016.

[8] S. Dayanık, *IE-325 Stochastic Models*, Aug. 18, 2014. Available: https://www.youtube.com/playlist?list=PLhwVAYxlh5dvUzmxzY91ttEJF-cj_29Sv

[9] Wikipedia contributors, "Markov chain — Wikipedia, The Free Encyclopedia," Feb. 2020. Avaliable: https://en.wikipedia.org/wiki/Markov_chain

[10] Team Simpy Revision, "Documentation for SimPy," April 2020. Available: https://simpy.readthedocs.io/en/latest/contents.html

[11] G. Casale, *Tutorial on Continuous Time Markov Chains, Phase-Type Distributions and Markovian Arrival Processes*, Aug. 28 2019. Available: https://www.youtube.com/watch?v=oUwrCxqOQVU

[12] The BuTools Team, "BuTools Reference," 2013. Avaliable: http://webspn.hit.bme.hu/~telek/tools/butools/doc/index.html

[13] B. R. Haverkort, "Part V: Simulation," *in Performance of computer communication systems: a model-based approach*, Chichester: John Wiley; Sons, 1999, pp. 409–437.

Appendices

A Event Based Simulation

```
import simpy
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import random
from butools.ph import *

def exponentialTime(env, arrivalRate, arrivals):
    while True:
        t = np.random.exponential(arrivalRate, 1)
        arrivals.append(t[1])
        yield env.timeout(t)

def PHTime(env, alpha, A, arrivals, peaks):
    count = 0
    while True:
        t = SamplesFromPH(alpha, A, 1)
        peaks.append(arrivals[count] + t)
        count += 1
        yield env.timeout(t)

def feasibleBounds (n):
    ly = []
    uy = []
    x = []
    for m2 in np.linspace(APH2ndMomentLowerBound(1,n), 1.8, 500):
        # convert to normalized moments
        ml = NormMomsFromMoms ([1, m2, APH3rdMomentLowerBound(1,m2,n)])
        mu = NormMomsFromMoms ([1, m2, APH3rdMomentUpperBound(1,m2,n)])
        # record bounds
        ly.append(ml[2])
        uy.append(min(mu[2],15))
        x.append(ml[1])
    return (x, ly, uy)

env = simpy.Environment()
butools.verbose = True

# single buffer M/PH/1/2/R(r) Poisson arrival PH service R-LCFS which has a
#replacement prob of r(only replaces the buffer not the ongoing packet)
```

```
# M/PH/1/2 NP-FCFS
r = 0
j = 4
lam = 1.0
rho = 0.75
scov_service = 1.0/j
mu = lam*j/rho
# j > 1 and integer, scov < 1 => Erlang with order j
alpha = ml.matrix([[1, 0, 0, 0]])
S = ml.matrix(
    [[-mu, mu, 0, 0],
     [0, -mu, mu, 0],
     [0, 0, -mu, mu],
     [0, 0, 0, -mu]])
t = []
sig = []
PAoI = []

env.process(exponentialTime(env, lam, t))
env.process(PHTime(env, alpha, S, t, PAoI))
env.run(until=1000)
print(t)
print(PAoI)
```

B Time Based Simulation

```
import numpy as np
import copy
import matplotlib.pyplot as plt
from butools.ph import SamplesFromPH
from butools.ph import ml

sim_time = 40
sim_step = 1000
num_sim = 100
num_src = 3
gen_age = []
gen_peak = []
for i in range(num_src):
    gen_age.append([])
    gen_peak.append([])
    for j in range(sim_time*sim_step):
        gen_age[i].append(0.0)
        gen_peak[i].append(0.0)

num_process = 10*sim_time
p_mode = 3 #1: global, 2: self, 3: priotirized
averages = []

for l in range(num_sim):
    print("Sim %d" % l )
    IAT_list = []
    ST_list = []
    AT_list = []
    ST_Backups = []
    all_arr_times = []
    p = []
    e = []
    r = []
    lam = [1.0, 2.0, 3.0]
    for i in range(num_src):
        k = 4.0 #this has two modes: 2 and 4
        rho = 2.0/3.0
        mu = sum(lam)*rho/k
        if int(round(k)) == 4:
            alpha = ml.matrix([[1, 0, 0, 0]])
            S = ml.matrix(
                [[-mu, mu, 0, 0],
                [0, -mu, mu, 0],
                [0, 0, -mu, mu],
                [0, 0, 0, -mu]])
        else:
            alpha = ml.matrix([[1, 0]])
            S = ml.matrix(
                [[-mu, mu],
                [0, -mu]])
        averages.append([])
        p.append([])

        #global preemption
        if p_mode == 1:
            for j in range(num_src):
                p[i].append(1.0)
        #global preemption
        elif p_mode == 2:
            for j in range(num_src):
                if i == j:
                    p[i].append(1.0)
                else:
                    p[i].append(-1.0)
        #priotirized preemption
        else:
            for j in range(num_src):
                if i >= j:
                    p[i].append(1.0)
                else:
                    p[i].append(-1.0)

        e.append(0.1)
        r.append(1.0)
        IAT = []
        ST = []
        AT = []
        #Generating Interarrival Times
        for j in range(num_process):
            tmp = np.random.exponential(1/lam[i])*sim_step
            IAT.append(int(round(tmp)))

        #Generating Service Times
        for j in range(num_process):
            tmp = SamplesFromPH(alpha, S, 1)[0]*sim_step
            ST.append(int(round(tmp)))
        #Calculating the Arrival Times and stamping the source info and
           process no of the packet
        for j in range(num_process):
            if j == 0:
                AT.append(IAT[j])
                all_arr_times.append((IAT[j], i, j))
            else:
                AT.append(AT[j-1] + IAT[j])
                all_arr_times.append((AT[j-1] + IAT[j], i, j))

        IAT_list.append(IAT)
        ST_list.append(ST)
        AT_list.append(AT)

    #Backing up the service times in case of a packet error
    ST_Backups = copy.deepcopy(ST_list)

    #Sorting according the arrival times
    all_arr_times.sort()

    server_busy = False
    list_age = []
    list_processed = []
    ages = []
    for i in range(num_src):
        ages.append(0.0)
        list_age.append([])
        list_processed.append([])

    for i in range(sim_time*sim_step):

        if server_busy:
            ST_list[int(current_src)][int(current_prc)] -= 1
            if ST_list[int(current_src)][int(current_prc)] == 0:
                if np.random.uniform() < e[int(current_src)]:
                    #Packet transmission error occured
                    if np.random.uniform() < r[int(current_src)]:
                        #Packet retransmitted
                        ST_list[int(current_src)][int(current_prc)] = ST_
                           Backups[int(current_src)][int(current_prc)]
                    else:
                        server_busy = False
                else:
                    server_busy = False
                    last_src = 0
```

```
                    if(len(list_processed[int(current_src)]) < 1):
                        (arr, dummy, spec_prc) = all_arr_times[int(current_prc
                           )]
                        dec = arr
                    else:
                        last_prc = list_processed[int(current_src)][-1]
                        (arr_now, dummy, spec_prc) = all_arr_times[int(current
                           _prc)]
                        (arr_last, dummy, spec_prc) = all_arr_times[int(last_
                           prc)]
                        dec = arr_now - arr_last + 1
                    list_processed[int(current_src)].append(current_prc)

                    #Recording the Peak Age of Information
                    gen_peak[int(current_src)][int(round(ages[int(current_src)
                       ]))] += 1.0
                    ages[int(current_src)] -= dec

        for j in range(num_src):
            #recording the Age of Information
            gen_age[j][int(round(ages[j]))] += 1.0
            ages[j] += 1.0

        for j in range(num_process):
            (proc_time, proc_src, spec_prc) = all_arr_times[j]
            if i == proc_time:
                if not server_busy:
                    (current_prc, current_spec_prc, current_src) = (j, spec_
                       prc, proc_src)
                    server_busy = True
                elif np.random.uniform() < p[current_src][proc_src]:
                    #Packet preempted
                    (preempted_pckt, preempted_spec_pckt, preempted_src) = (
                       current_prc, current_spec_prc, current_src)
                    (current_prc, current_spec_prc, current_src) = (j, spec_
                       prc, proc_src)
                    server_busy = True
        for j in range(num_src):
            list_age[j].append(ages[j])
    for i in range(num_src):
        averages[i].append(sum(list_age[i])/len(list_age[i]))

#This part is used for visualization, recording and scaling of the data to
   get PDF and CDF of age and peak age
#Plots produced in this part functions as a previsualization, Final plots
```

```
   produced in MATLAB with recorded data

#Scaling of the data to fit the CDF and PDF convention
for i in range(num_src):
    tot = sum(gen_age[i])
    for j in range(sim_time*sim_step):
        gen_age[i][j] = gen_age[i][j]/tot

for i in range(num_src):
    tot = sum(gen_peak[i])
    for j in range(sim_time*sim_step):
        gen_peak[i][j] = gen_peak[i][j]/tot

o = 1
#Plot of the PDF of AoI
for j in range(num_src):
    ran = int(sim_time*sim_step/o)
    plt.plot([(float(i)+1)/float(sim_step) for i in range(ran)], gen_age[j
       ][:ran], label='Source %d' % (j + 1))
    print("Average age of source %d is %d" % (j+1, sum(averages[j])/len(
       averages[j])))
    plt.legend()
plt.ylabel("AoI PDF")
plt.xlabel("Time (ms)")
plt.show()

#Recording PDF of AoI
with open('AOI_PDF_%d_%f.txt'% (p_mode, k), 'w') as file1:
    file1.writelines("%s\n" % place for place in gen_age)

#Plotting CDF of AoI
cdf = []
for i in range(num_src):
    cdf.append([])
    sums = 0
    cdf[i].append(0)
    for j in range(sim_step*sim_time):
        sums += gen_age[i][j]
        cdf[i].append(sums)
    ran = int(sim_time*sim_step/o)
    plt.plot([(float(l)+1)/float(sim_step) for l in range(ran)], cdf[i][:ran
       ], label='Source %d' % (i + 1))
    plt.legend()
plt.ylabel("AoI CDF")
plt.xlabel("Time (ms)")
plt.show()

#Recording CDF of AoI
with open('AOI_CDF_%d_%f.txt'% (p_mode, k), 'w') as file2:
       file2.writelines("%s\n" % place for place in cdf)

#Plotting PDF of PAoI
for j in range(num_src):
    ran = int(sim_time*sim_step/o)
    plt.plot([(float(l)+1)/float(sim_step) for l in range(ran)], gen_peak[j
       ][:ran], label='Source %d' % (i + 1))
    plt.legend()
plt.ylabel("PAoI PDF")
plt.xlabel("Time (ms)")
plt.show()

#Recording PDF of PAoI
with open('PAOI_PDF_%d_%f.txt' % (p_mode, k), 'w') as file3:
    file3.writelines("%s\n" % place for place in gen_peak)


#Plotting CDF of PAoI
cdf_peak = []
for i in range(num_src):
    cdf_peak.append([])
    sum = 0
    for j in range(sim_step*sim_time):
        sum += gen_peak[i][j]
        cdf_peak[i].append(sum)
    ran = int(sim_time*sim_step/o)
    plt.plot([(float(l)+1)/float(sim_step) for l in range(ran)], cdf_peak[i
       ][:ran], label='Source %d' % (i + 1))
    plt.legend()
plt.ylabel("PAoI CDF")
plt.xlabel("Time (ms)")
plt.show()

#Recording CDF of PAoI
with open('PAOI_CDF_%d_%f.txt' % (p_mode, k), 'w') as file4:
    file4.writelines("%s\n" % place for place in cdf_peak)
```

C MATLAB Script for Plots

```
clear all
a=importdata('a_11_1_sim_peak.dat');
clear t
for i = 1:2001
dat(i)=a(i+2001);
t(i)=a(i);
end
plot(t,dat)
for i = 1:2001
dat(i)=a(i+2001);
t(i)=a(i);
end
plot(t,dat)
hold on
a=importdata('a_11_2_sim_peak.dat');
for i = 1:2001
dat(i)=a(i+2001);
t(i)=a(i);
end
plot(t,dat)
a=importdata('a_11_3_sim_peak.dat');
for i = 1:2001
dat(i)=a(i+2001);
t(i)=a(i);
end
plot(t,dat)
a=importdata('PAOI_CDF_1_4.000000.txt');
a=a';
for i = 1:8001
f(i)=a(i);
s(i)=a(i+length(a));
th(i)=a(i+2*length(a));
end
t=linspace(0,20,8001);
plot(t,f,'--')
plot(t,s,'--')
plot(t,th,'--')
legend('Src1','Src2','Src3','MySRC1','MySRC2','MySRC3','Location','southeast')
```