Computer Simulation
Slotted Aloha

Amano Shuya – BP20046
Gabriel Kishida – Z122232
Han Ying Yu – Z123111

Table of contents

01

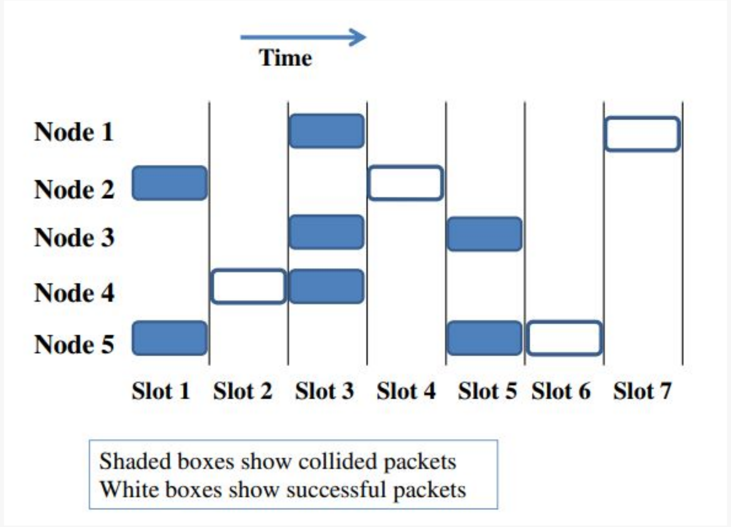
Definition: Slotted Aloha

Slotted Aloha

The Slotted Aloha is a data transfer protocol that divides time into time slots. Each time slot allows for the transmission of one packet of data.

If two or more users try to transfer data at the same time slot, collision occurs and transfer fails.

If only one user transfers data, data transfer succeeds.



The Assignment

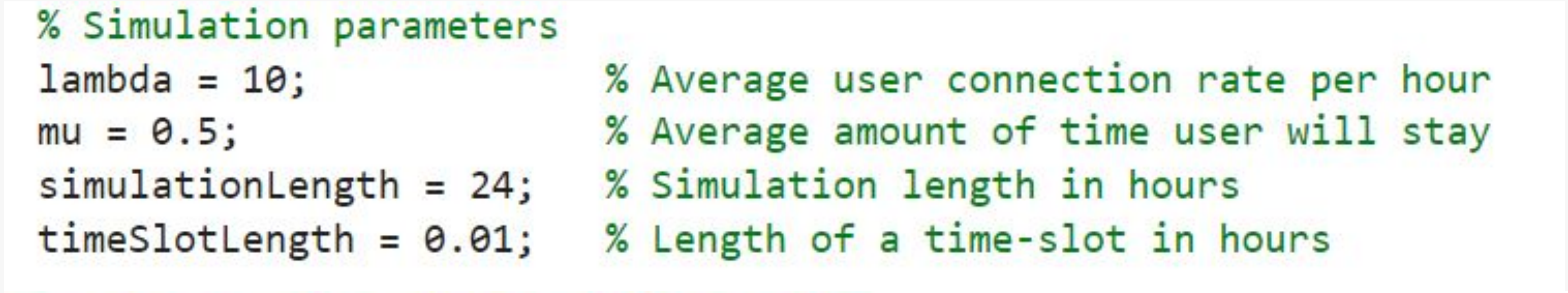
The assignment consists of analyzing the throughput – that is, how many messages are successfully transmitted per time slot – when the probability of access of each user varies.

That is, we will study how the probability of access by each user will affect the total data transfer.

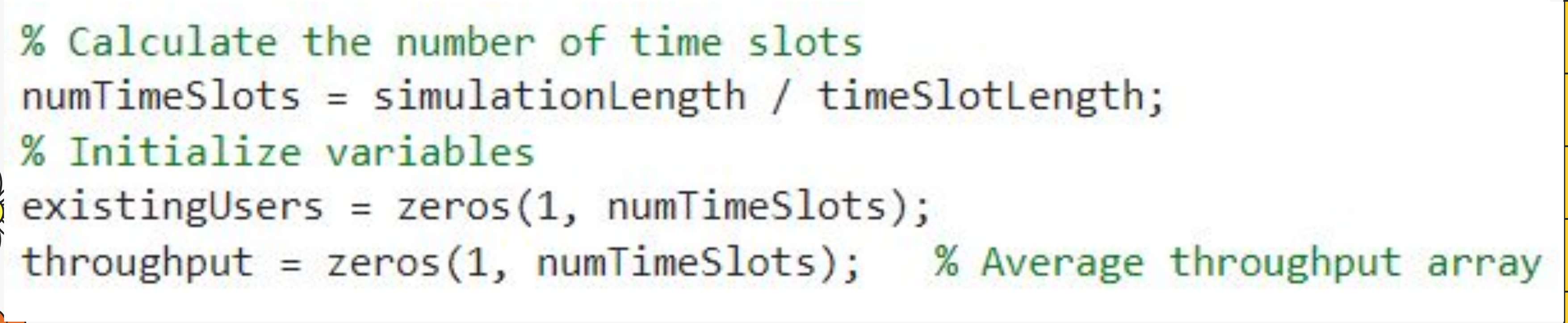
According to theory, when more users access at the same time, more data collision will occur, and therefore less data will be transferred.

However, if no user accesses the time slot, then no data is transferred, and time will be wasted.

Setting up Variables



```
% Simulation parameters
lambda = 10;                % Average user connection rate per hour
mu = 0.5;                   % Average amount of time user will stay
simulationLength = 24;      % Simulation length in hours
timeSlotLength = 0.01;      % Length of a time-slot in hours
```

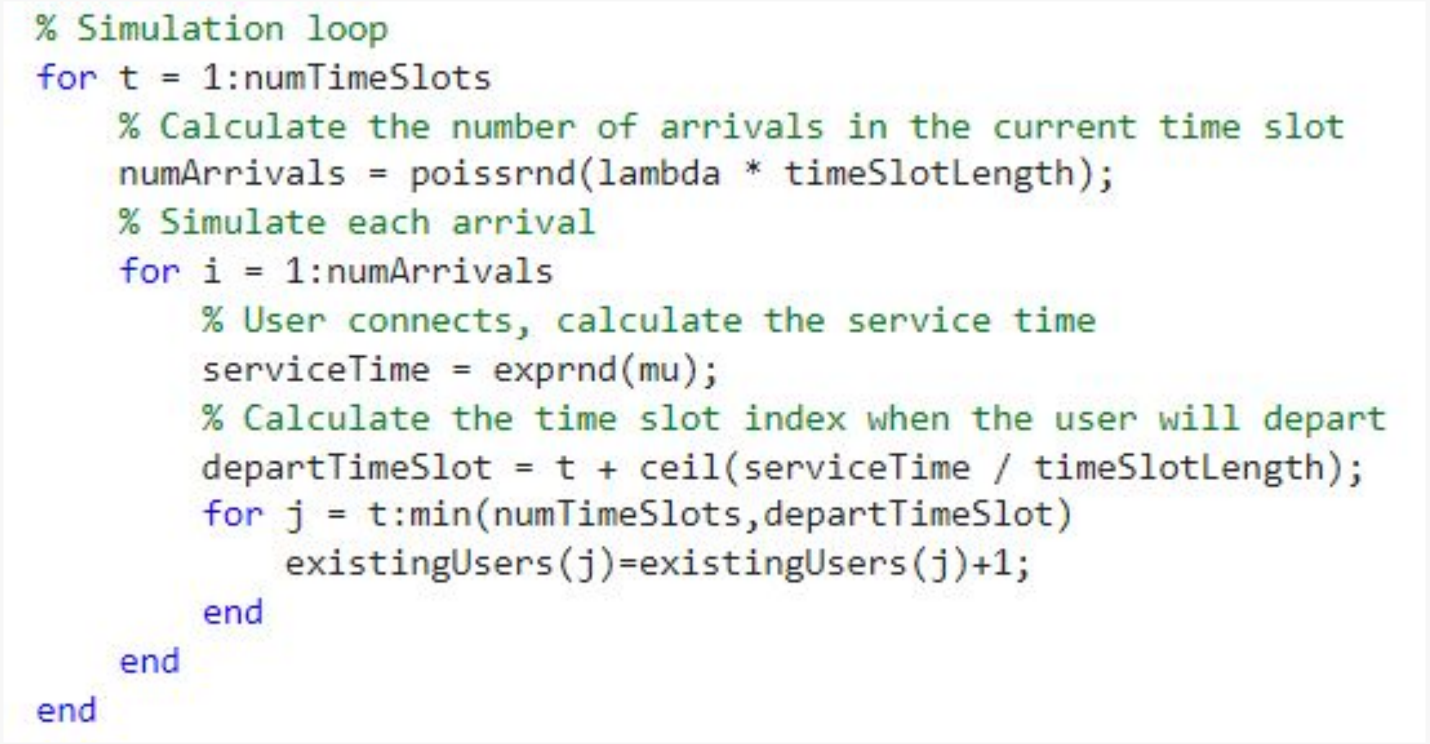


```
% Calculate the number of time slots
numTimeSlots = simulationLength / timeSlotLength;
% Initialize variables
existingUsers = zeros(1, numTimeSlots);
throughput = zeros(1, numTimeSlots);    % Average throughput array
```

Setting up users and service time

Our strategy was to determine how many users would arrive per time slot, calculate how long they would stay, and build an array of how many users exist within a given time slot.

Setting up users and service time



```matlab
% Simulation loop
for t = 1:numTimeSlots
    % Calculate the number of arrivals in the current time slot
    numArrivals = poissrnd(lambda * timeSlotLength);
    % Simulate each arrival
    for i = 1:numArrivals
        % User connects, calculate the service time
        serviceTime = exprnd(mu);
        % Calculate the time slot index when the user will depart
        departTimeSlot = t + ceil(serviceTime / timeSlotLength);
        for j = t:min(numTimeSlots,departTimeSlot)
            existingUsers(j)=existingUsers(j)+1;
        end
    end
end
```

02

Formula Solution

Formula Solution

The formula solution uses the following formula for calculating the throughput of a given user i:

- x_i and x_j are probabilities of user i and j, respectively, accessing the network,

$$x_i = \prod_{j=1, j \neq i}^{n} (1 - x_j)$$

Formula Solution

However, since we know that the probabilities of all users accessing are the same, known as p, we can instead use the following formula:

$$x_i = (1 - p)^n$$

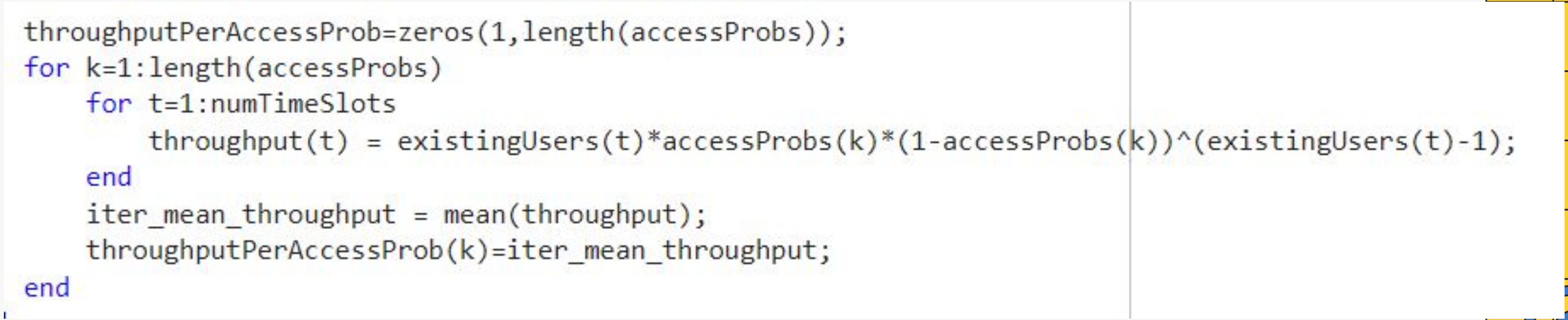
Formula Solution

Finally, since we want the formula that gives the total throughput, not the throughput of each user, we can get the following:

$$\sum_{i=1}^{n} x_i = n(1-p)^n$$

Formula Solution in Code

This gives us the following code:



```
throughputPerAccessProb=zeros(1,length(accessProbs));
for k=1:length(accessProbs)
    for t=1:numTimeSlots
        throughput(t) = existingUsers(t)*accessProbs(k)*(1-accessProbs(k))^(existingUsers(t)-1);
    end
    iter_mean_throughput = mean(throughput);
    throughputPerAccessProb(k)=iter_mean_throughput;
end
```

03

Brute Force Solution

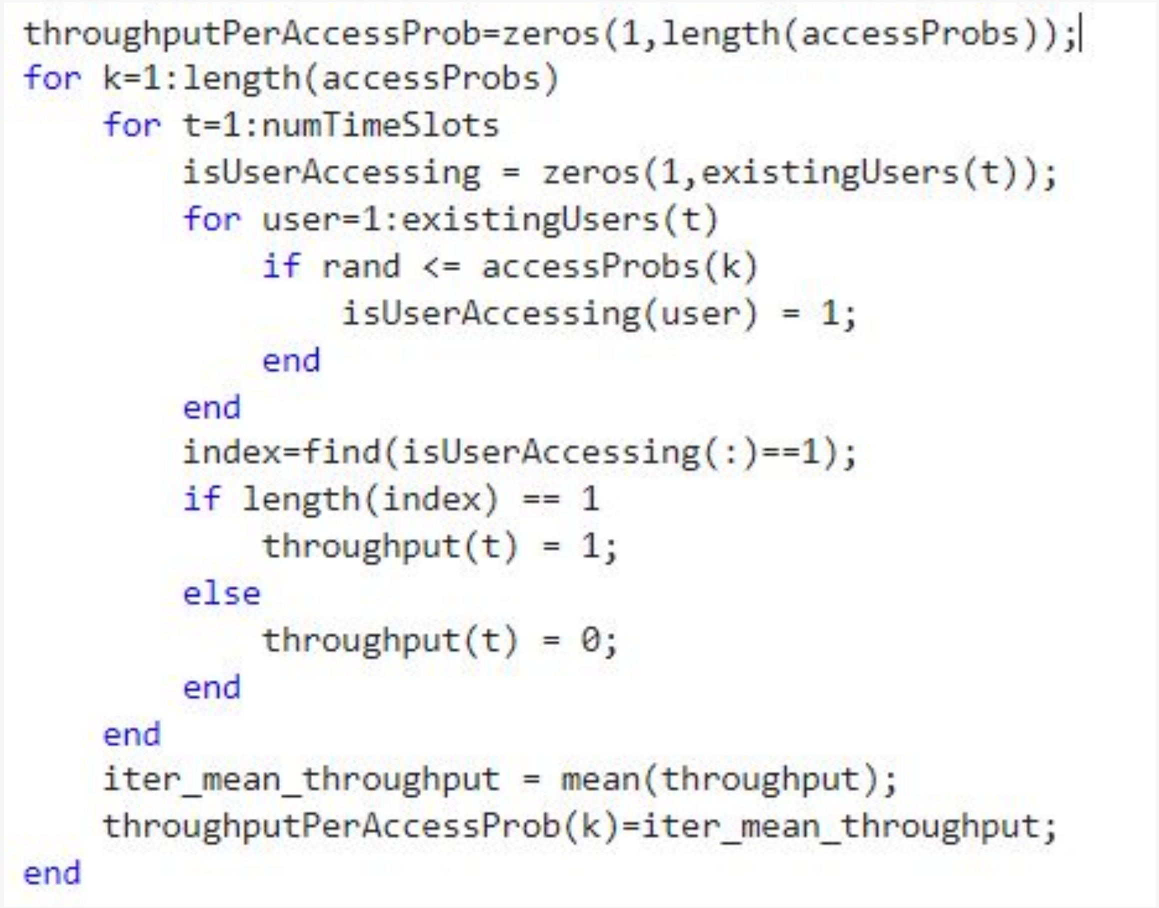
Brute Force Solution

The brute force solution consists of, for each user, generating a random number between 0 and 1. If the number is smaller than p, then the user will attempt a connection. If it is bigger, then it won't.

If more than 1 user attempts to connect, or if no user connects, the throughput for that time slot will be 0.

If one and only one user connects, then the throughput will be 1 (one data packet was transferred).

Brute Force Solution Code



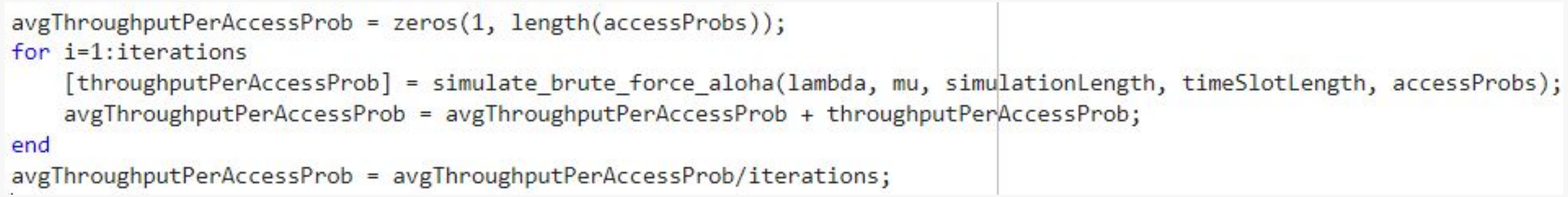
```
throughputPerAccessProb=zeros(1,length(accessProbs));
for k=1:length(accessProbs)
    for t=1:numTimeSlots
        isUserAccessing = zeros(1,existingUsers(t));
        for user=1:existingUsers(t)
            if rand <= accessProbs(k)
                isUserAccessing(user) = 1;
            end
        end
        index=find(isUserAccessing(:)==1);
        if length(index) == 1
            throughput(t) = 1;
        else
            throughput(t) = 0;
        end
    end
    iter_mean_throughput = mean(throughput);
    throughputPerAccessProb(k)=iter_mean_throughput;
end
```

04

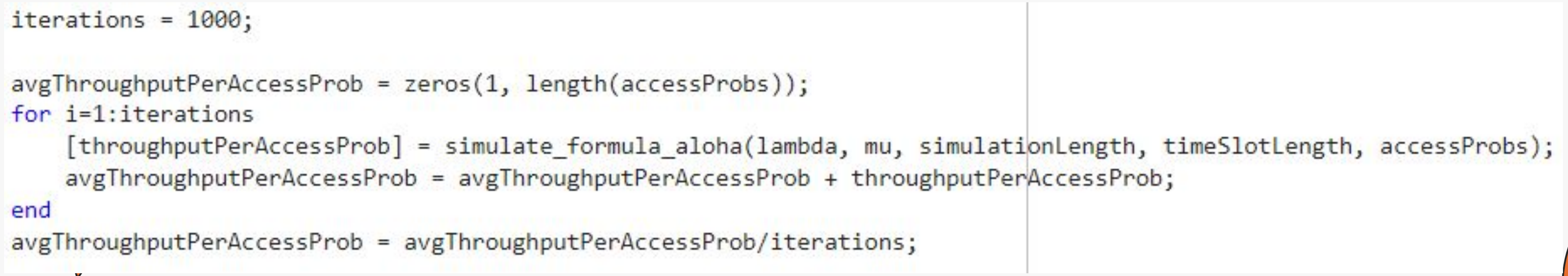
Results

Running Many Iterations

Since we wish to collect reliable data, we must run a reasonable number of iterations of each simulation. Therefore, we turned each simulation method into a function, and ran it 1000 times.



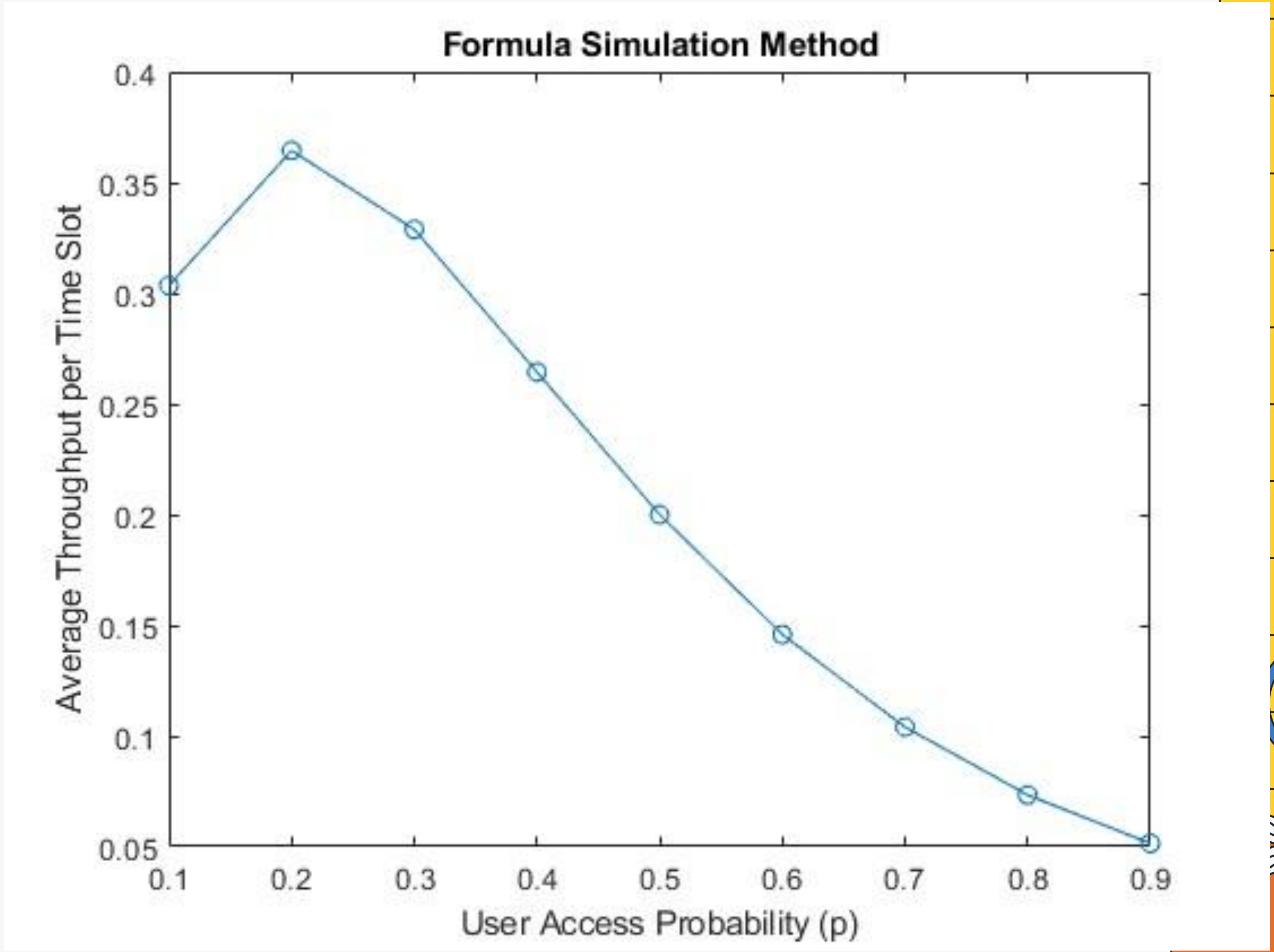
```
avgThroughputPerAccessProb = zeros(1, length(accessProbs));
for i=1:iterations
    [throughputPerAccessProb] = simulate_brute_force_aloha(lambda, mu, simulationLength, timeSlotLength, accessProbs);
    avgThroughputPerAccessProb = avgThroughputPerAccessProb + throughputPerAccessProb;
end
avgThroughputPerAccessProb = avgThroughputPerAccessProb/iterations;
```



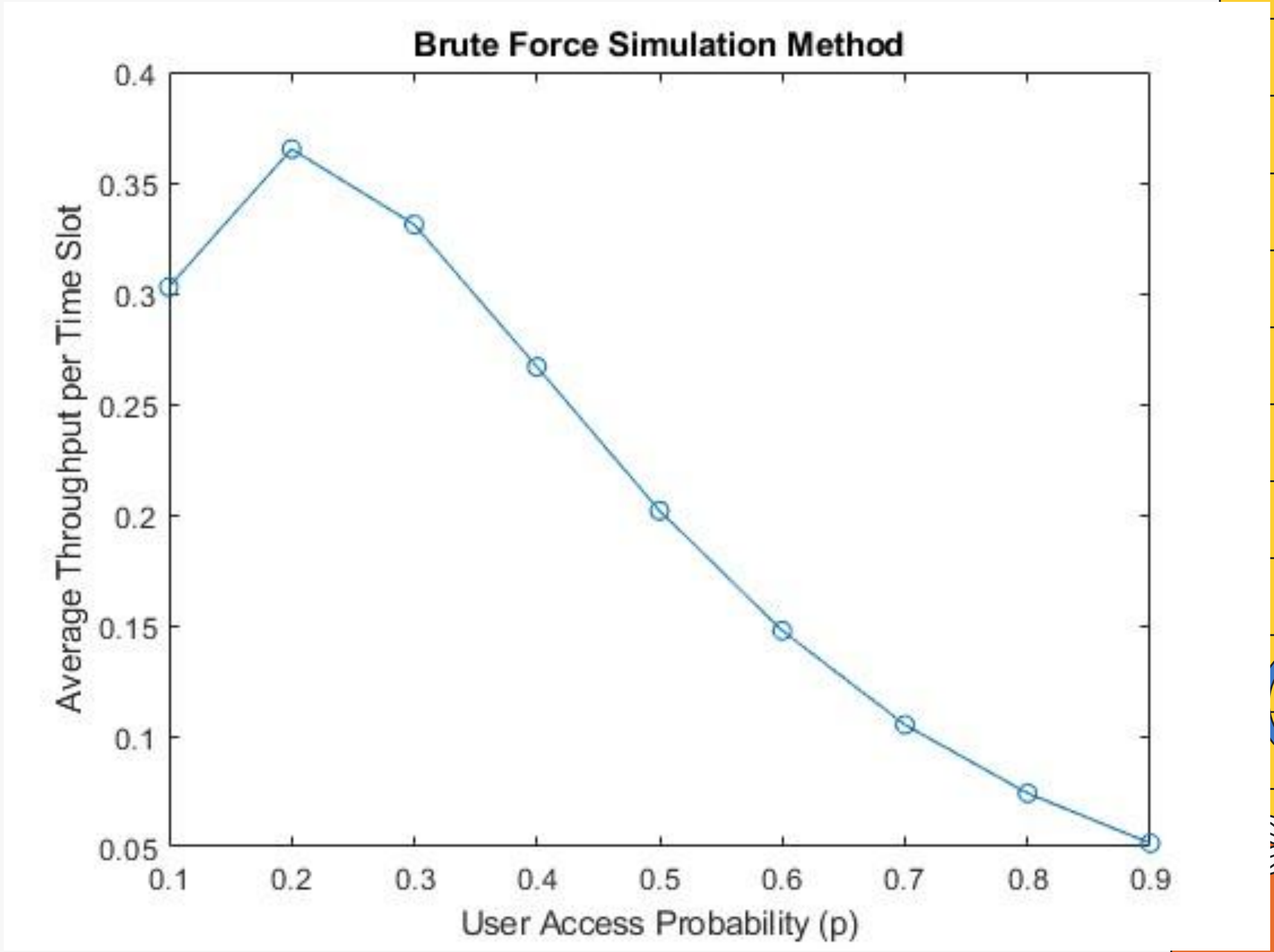
```
iterations = 1000;

avgThroughputPerAccessProb = zeros(1, length(accessProbs));
for i=1:iterations
    [throughputPerAccessProb] = simulate_formula_aloha(lambda, mu, simulationLength, timeSlotLength, accessProbs);
    avgThroughputPerAccessProb = avgThroughputPerAccessProb + throughputPerAccessProb;
end
avgThroughputPerAccessProb = avgThroughputPerAccessProb/iterations;
```

Formula Results



Brute Force Results



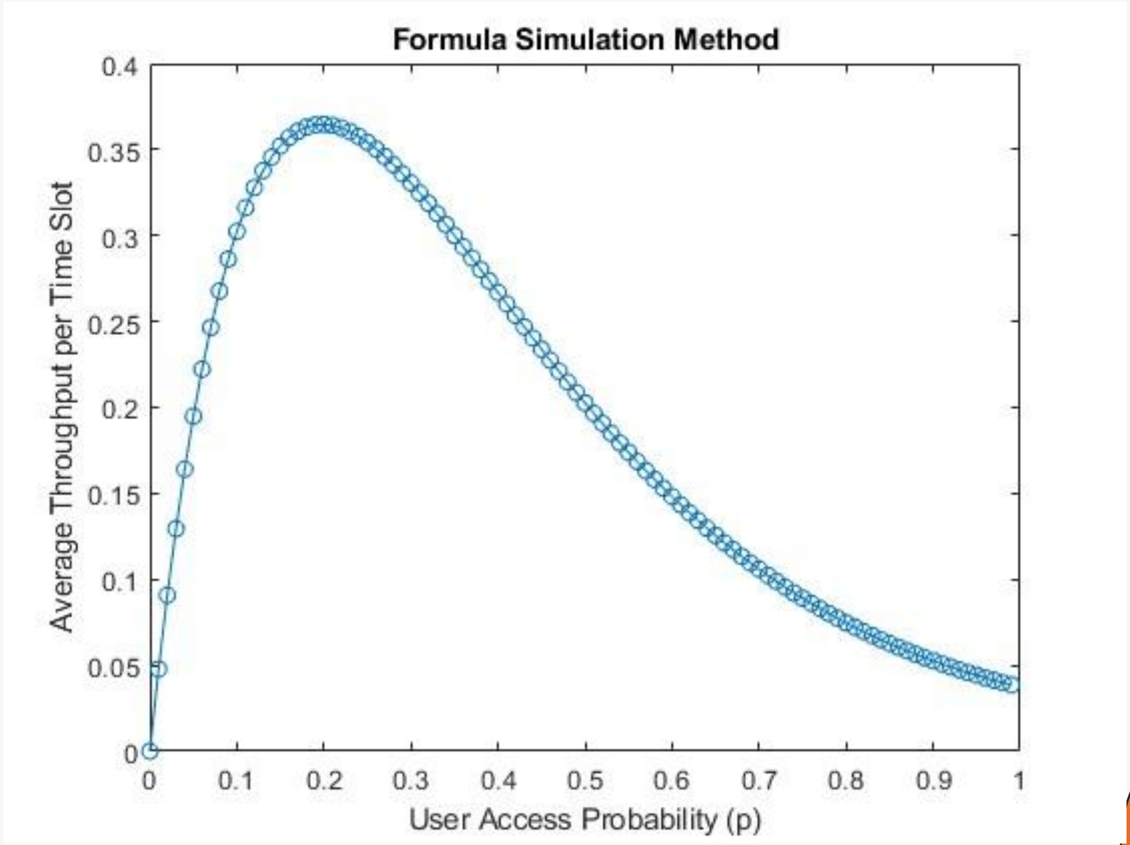
Conclusion

The fact that two different methods attained almost identical results is proof of its accuracy.

The Brute Force method may be better since it has a simpler logic and is easier to apply if each user has a different probability of access.

Bonus: Smooth Curve

Obtained using range 0 to 1, 0.01 steps



Thank you!

Do you have any questions?

Code available at:

https://github.com/GabrielKishida/computer_simulation/tree/main