11.18 목 - [MATLAB] scheduling

scheduling_lab1.m - MCS 고정

```
%-----------------------------------------
% INC4103: 스케줄링 실습1
% MCS가 고정된 경우
%-----------------------------------------

% 3가지 스케줄링 정책 고려함
% case1: max-min
% case2: proportional fair
% case3: round-robin

% 하나의 시간 슬롯은 한 명의 사용자에게 할당된다고 가정

clear;
close all;

N_slot = 1000;    % 전체 슬롯 수
N_user = 3;       % 사용자 수

% 변수 선언 & 초기화
% 각 스케줄링 기법별 사용자별 누적 전송 데이터 양
bytes_sent_case1 = zeros(1,N_user);
bytes_sent_case2 = zeros(1,N_user);
bytes_sent_case3 = zeros(1,N_user);

% 각 스케줄링 기법별 사용자별 평균 처리율 (누적 평균)
% (행,열)=(시간,사용자)
avg_th_case1 = zeros(N_slot,N_user);
avg_th_case2 = zeros(N_slot,N_user);
avg_th_case3 = zeros(N_slot,N_user);

% 채널 상태는 변하지 않는다고 가정
% 3명의 사용자 고려함
%   - User1: SNR=30dB --> Rate = 8 bit/sec
%   - User2: SNR=10dB --> Rate = 4 bit/sec
%   - User3: SNR= 5dB --> Rate = 1 bit/sec
inst_rate = [8 4 1]; % 각 사용자별 슬롯당 전송 바이트 양

% 매 슬롯별 스케줄링 결과 선택된 사용자 인덱스
user_index_case1 = zeros(N_slot,1);
user_index_case2 = zeros(N_slot,1);
user_index_case3 = zeros(N_slot,1);

% 초기값은 random하게 설정함
user_index_case1(1) = ceil(rand*N_user);
user_index_case2(1) = ceil(rand*N_user);
user_index_case3(1) = ceil(rand*N_user);

% 사용자별 할당된 (스케줄링된) 슬롯의 합계
counter_user_index_case1 = zeros(1,N_user);
counter_user_index_case2 = zeros(1,N_user);
counter_user_index_case3 = zeros(1,N_user);


for i = 2:N_slot

   % case1: max-min
   for j=1:N_user
       if (j==user_index_case1(i-1))
           % i번째 슬롯을 할당받는 사용자 j는
           % 이전 (i-1)번째 슬롯 시간동안의 평균 처리율로부터 계산하고
           % 사용자 j는 전송 데이터 양을 전송속도만큼 증가시키고
           % 할당받은 슬롯수를 1 증가
           bytes_sent_case1(j) =  bytes_sent_case1(j)+inst_rate(j);
           counter_user_index_case1(j) = counter_user_index_case1(j)+1;
       end
       % 사용자별 처리율 업데이트
       % i번째 슬롯 시간까지 누적하여 전송한 데이터 양
       avg_th_case1(i,:) = (bytes_sent_case1)/i;
       % 다음 슬롯 시간의 스케줄링 사용자 결정
       [temp user_index_case1(i)] = min(avg_th_case1(i,:));
       % max-min은 평균 처리율이 가장 낮은 사용자를 선택함
       % 만약, 같은 최소값을 가진다면 그 중 첫번째
       % min()함수의 return값은 2개인데,
       % 첫번째가 최소값, 두번째가 최소값을 가지는 인덱스
```

```
 end

% case2: proportional fair
for j=1:N_user
    if (j==user_index_case2(i-1))
         bytes_sent_case2(j) =  bytes_sent_case2(j)+inst_rate(j);
         counter_user_index_case2(j) = counter_user_index_case2(j)+1;
    end
    avg_th_case2(i,:) = max((bytes_sent_case2)/i,eps);
    % 평균 처리율이 0이면 나눗셈 연산시 오류가 발생하여
    % 아주 작은값 (eps)로 대체
end
[temp user_index_case2(i)] = max(inst_rate./avg_th_case2(i,:));
% PF 스케줄링은 전송속도/평균처리율값이 가장 큰 사용자 선택

% case3: round-robin
for j=1:N_user
     if (j==user_index_case3(i-1))
         bytes_sent_case3(j) =  bytes_sent_case3(j)+inst_rate(j);
         counter_user_index_case3(j) = counter_user_index_case3(j)+1;
     end
     avg_th_case3(i,:) = (bytes_sent_case3)/i;
end
user_index_case3(i) = mod(i,N_user)+1;
% RR 스케줄링은 순서대로 한번씩
% mod함수 이용
end

% 최종 처리율
avg_th_case1(N_slot,:);
avg_th_case2(N_slot,:);
avg_th_case3(N_slot,:);

% 스케줄링 확률
% = 할당받은 슬롯수 / 전체 슬롯수
scheduling_prob1 = counter_user_index_case1 / N_slot;
scheduling_prob2 = counter_user_index_case2 / N_slot;
scheduling_prob3 = counter_user_index_case3 / N_slot;

avg_th = [avg_th_case1(N_slot,:), sum(avg_th_case1(N_slot,:))
    avg_th_case2(N_slot,:), sum(avg_th_case2(N_slot,:))
    avg_th_case3(N_slot,:), sum(avg_th_case3(N_slot,:))]

scheduling_probability = [scheduling_prob1
    scheduling_prob2
    scheduling_prob3]

figure;

T = [1:N_slot];
plot(avg_th_case1);
hold on; plot(T,sum(avg_th_case1'),'k');
xlabel('time'); ylabel('throughput (bit/sec)');
legend('user1', 'user2', 'user3', 'total');
title('Max-min scheduling');
axis([1 N_slot, 0 6]);

figure;
stem(user_index_case1);
xlabel('time'); ylabel('index of user scheduled');
title('Max-min scheduling');
axis([1 30 0 3]);

figure;
plot(avg_th_case2);
hold on; plot(T,sum(avg_th_case2'),'k');
xlabel('time'); ylabel('throughput (bit/sec)');
legend('user1', 'user2', 'user3', 'total');
title('Proportional fair scheduling');
axis([1 N_slot, 0 6]);

figure;
stem(user_index_case2);
xlabel('time'); ylabel('index of user scheduled');
title('Proportional fair scheduling');
axis([1 30 0 3]);

figure;
plot(avg_th_case3);
hold on; plot(T,sum(avg_th_case3'),'k');
xlabel('time'); ylabel('throughput (bit/sec)');
legend('user1', 'user2', 'user3', 'total');
title('Round-robin scheduling');
axis([1 N_slot, 0 6]);

figure;
```

```
stem(user_index_case3);
xlabel('time'); ylabel('index of user scheduled');
title('Round-robin scheduling');
axis([1 30 0 3]);
```

case1 - 사용자1/2/3의 전송 속도 = 8/4/1

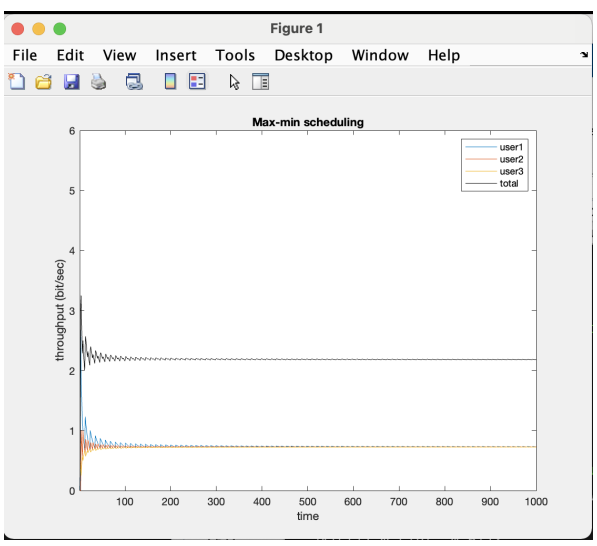
(R1, R2, R3) = (8, 4, 1)

R1 = 8

α = R1/R2 = 8/4 = 2

β = R1/R3 = 8/1 = 8

```
avg_th =

    0.7280    0.7280    0.7260    2.1820
    2.6640    1.3320    0.3330    4.3290
    2.6720    1.3280    0.3330    4.3330

scheduling_probability =

    0.0910    0.1820    0.7260
    0.3330    0.3330    0.3330
    0.3340    0.3320    0.3330
```



max-min

user1의 tput은 약 0.73bps,

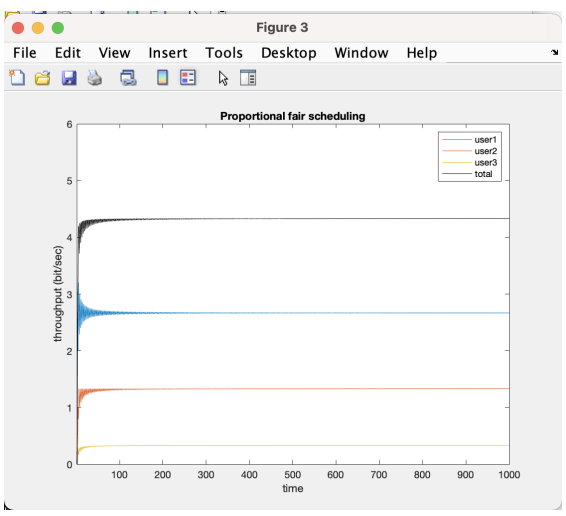
user2의 tput은 약 0.73bps,

user3의 tput은 약 0.73bps

으로 수렴했다.

1:1:1이다.

전체 tput은 약 2.18bps로, PF 방식과 RR 방식 대비 절반 정도의 처리율이다.



proportional fair

user1의 tput은 약 2.66bps,

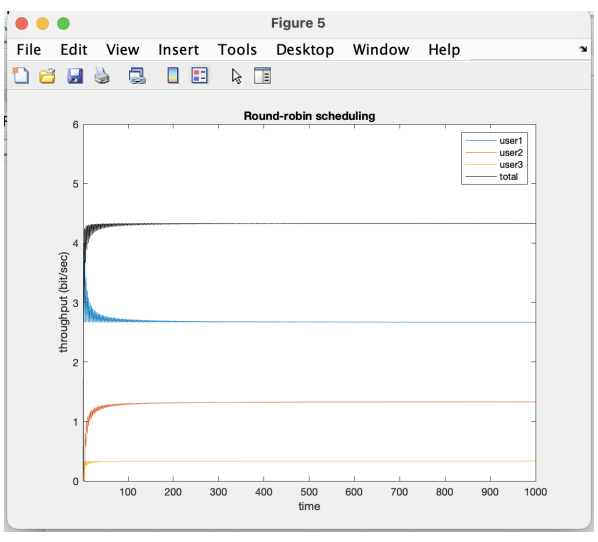
user2의 tput은 약 1.33bps,

user3의 tput은 약 0.33bps

으로 수렴했다.

약 8:4:1이다.

전체 tput은 약 4.33bps이다.



round-robin

user1의 tput은 약 2.67bps,

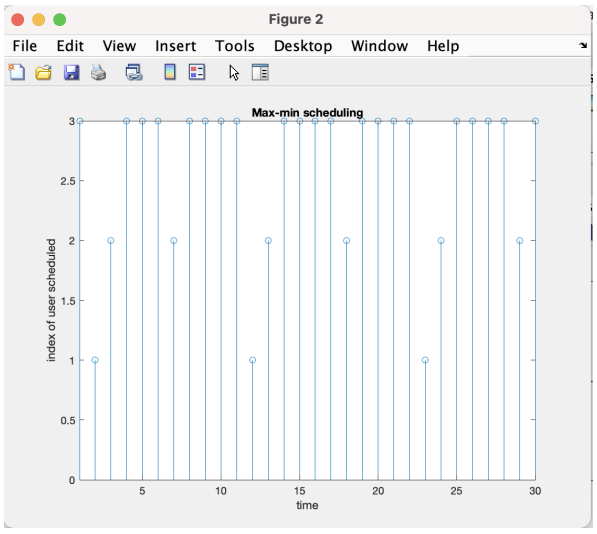
user2의 tput은 약 1.33bps,

user3의 tput은 약 0.33bps

으로 수렴했다.

약 8:4:1이다.

전체 tput은 약 4.33bps으로, PF 방식과 비슷한 처리율이다.



max-min

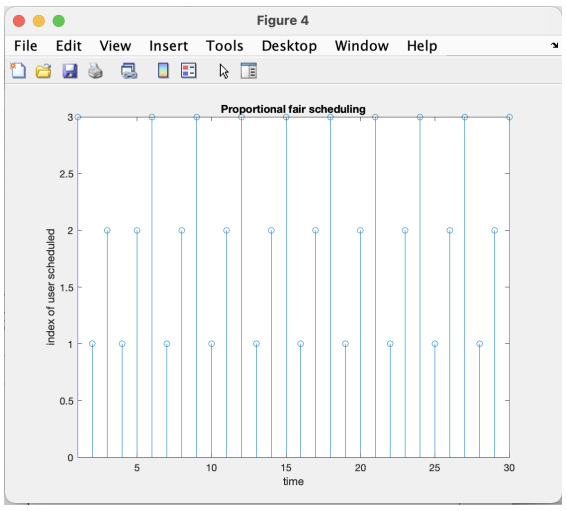
11번의 단위 slot 동안

user1은 1번,

user2는 2번,

user3는 8번 할당 받았다.

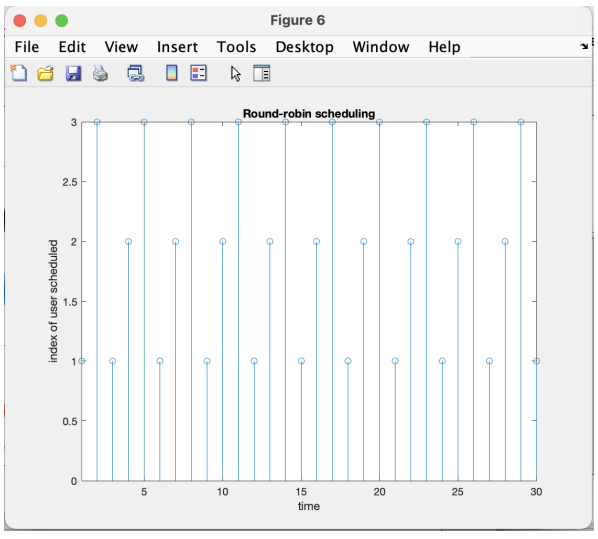
8bit*1s = 4bit*2s = 1bit*8s = 8bps로, tput
이 같아지도록 할당되었다.



proportional fair

대체적으로 주기성을 띄며 user1, user2, user3가 교대하여 slot을 할당 받는다.

PF 방식의 수식에 의해, 효율과 형평성을 고려하여 할당하므로 세 user들은 대체로 교대로 스케줄링된다.



round-robin

user1, user2, user3은 정확히 교대하여 slot을 할당 받는다.

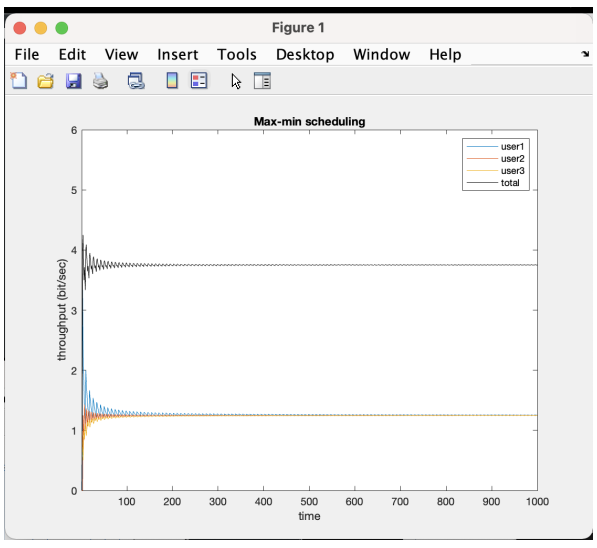
case2 - (R1, R2, R3) = (10, 5, 2)

R1 = 10

α = R1/R2 = 10/5 = 2

β = R1/R3 = 10/2 = 5

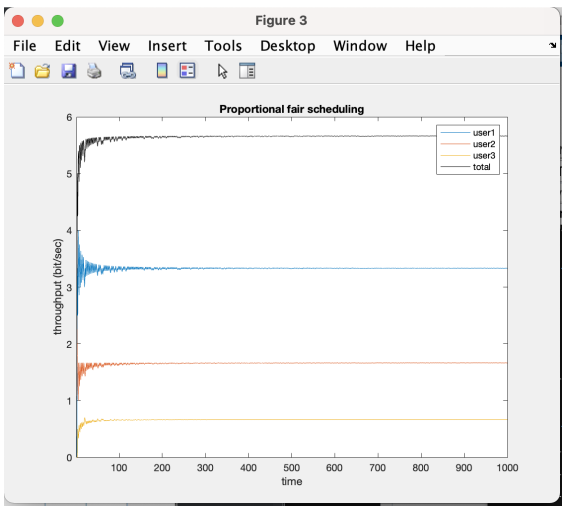
```
avg_th =

    1.2500    1.2500    1.2480    3.7480
    3.3300    1.6650    0.6660    5.6610
    3.3400    1.6600    0.6660    5.6660


scheduling_probability =

    0.1250    0.2500    0.6240
    0.3330    0.3330    0.3330
    0.3340    0.3320    0.3330

>>
```



max-min

user1의 tput은 약 1.25bps,

user2의 tput은 약 1.25bps,

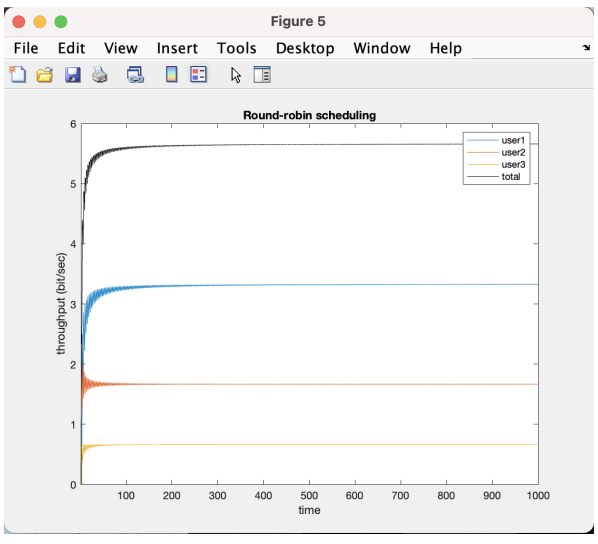


proportional fair

user1의 tput은 약 3.33bps,

user2의 tput은 약 1.67bps,

user3의 tput은 약 0.67bps



round-robin

user1의 tput은 약 3.34bps,

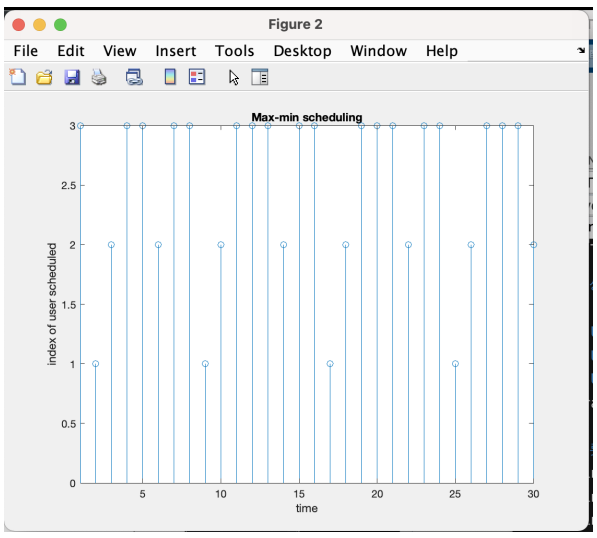
user2의 tput은 약 1.66bps,

user3의 tput은 약 1.25bps

으로 수렴했다.

1:1:1이다.

전체 tput은 약 3.75bps로, PF 방식과 RR 방식 대비 처리율이 낮다.



max-min

8번의 단위 slot 동안

user1은 1번,

user2는 2번,

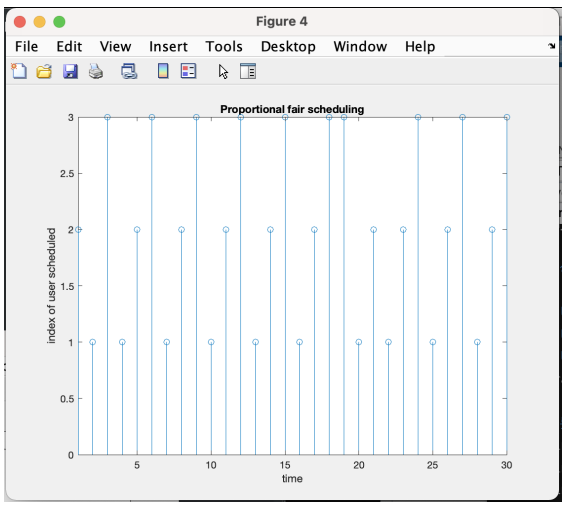
user3는 5번 할당 받았다.

10bit*1s = 5bit*2s = 2bit*5s = 10bps로, tput이 같아지도록 할당되었다.

으로 수렴했다.

약 10:5:2이다.

전체 tput은 약 5.66bps이다.



proportional fair

대체적으로 주기성을 띄며 user1, user2, user3가 교대하여 slot을 할당 받는다.

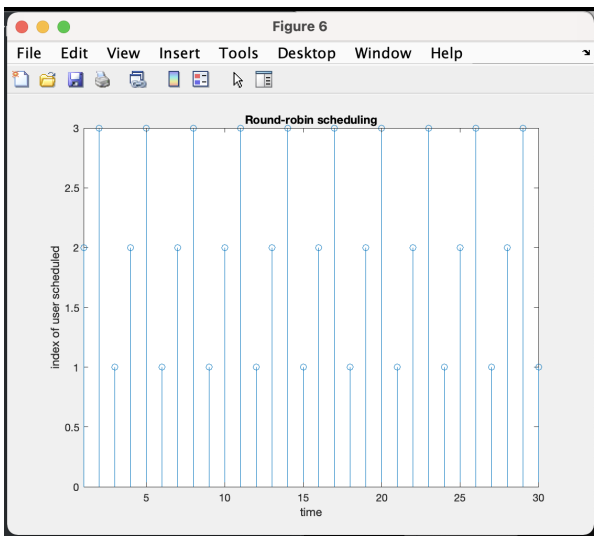
PF 방식의 수식에 의해, 효율과 형평성을 고려하여 할당하므로 세 user들은 대체로 교대로 스케줄링된다.

user3의 tput은 약 0.67bps

으로 수렴했다.

약 10:5:2이다.

전체 tput은 약 5.67bps으로, PF 방식과 비슷한 처리율이다.



round-robin

user1, user2, user3은 정확히 교대하여 slot을 할당 받는다.

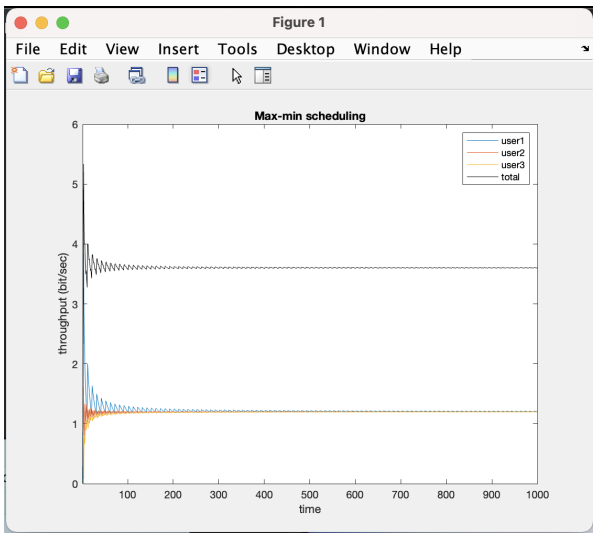
case3 - (R1, R2, R3) = (12, 4, 2)

R1 = 12

α = R1/R2 = 12/4 = 3

β = R1/R3 = 12/2 = 6

```
avg_th =

    1.2000    1.2000    1.1980    3.5980
    3.9960    1.3320    0.6660    5.9940


scheduling_probability =

    0.1000    0.3000    0.5990
    0.3330    0.3330    0.3330
    0.3330    0.3320    0.3340

>>
```



max-min

user1의 tput은 약 1.20bps,

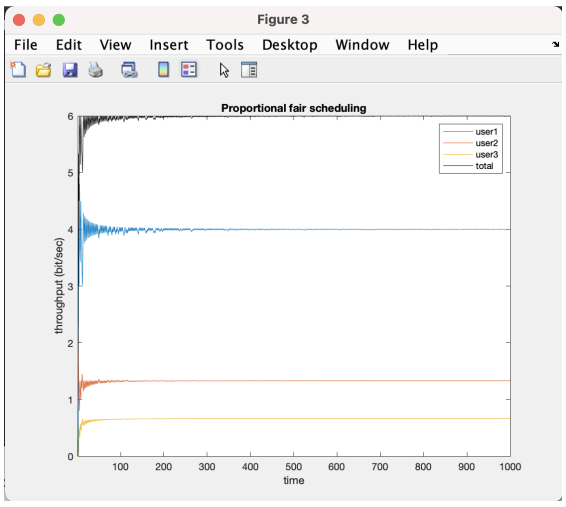
user2의 tput은 약 1.20bps,

user3의 tput은 약 1.20bps

으로 수렴했다.

1:1:1이다.

전체 tput은 약 3.60bps로, PF 방식과 RR 방식 대비 처리율이 낮다.



proportional fair

user1의 tput은 약 4bps,

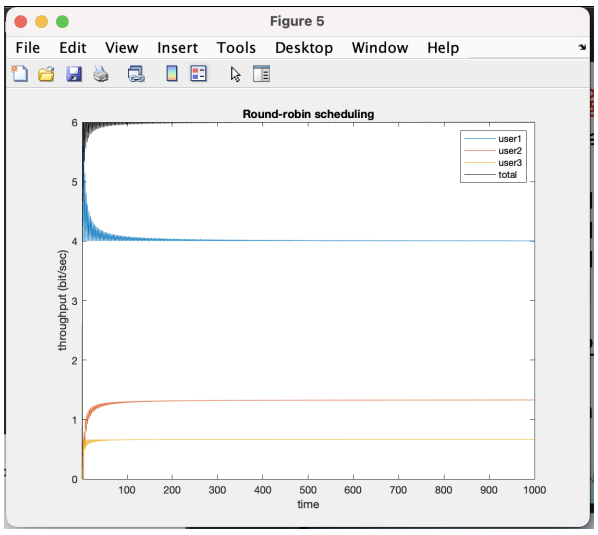
user2의 tput은 약 1.33bps,

user3의 tput은 약 0.67bps

으로 수렴했다.

약 12:4:2이다.

전체 tput은 약 6bps이다.



round-robin

user1의 tput은 약 4bps,

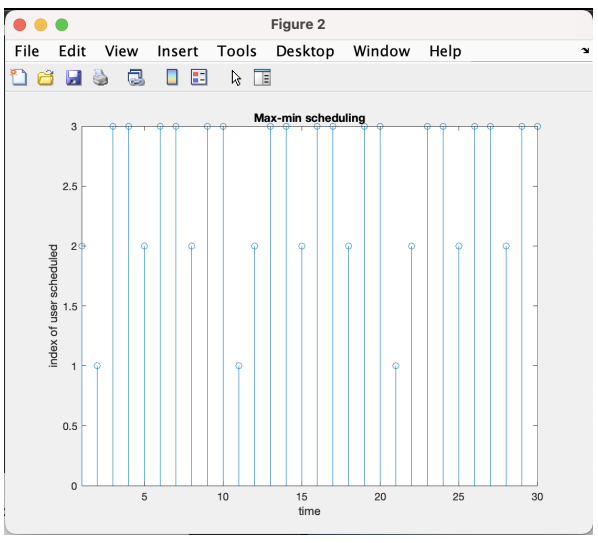
user2의 tput은 약 1.33bps,

user3의 tput은 약 0.67bps

으로 수렴했다.

약 12:4:2이다.

전체 tput은 약 6bps으로, PF 방식과 비슷한 처리율이다.



max-min

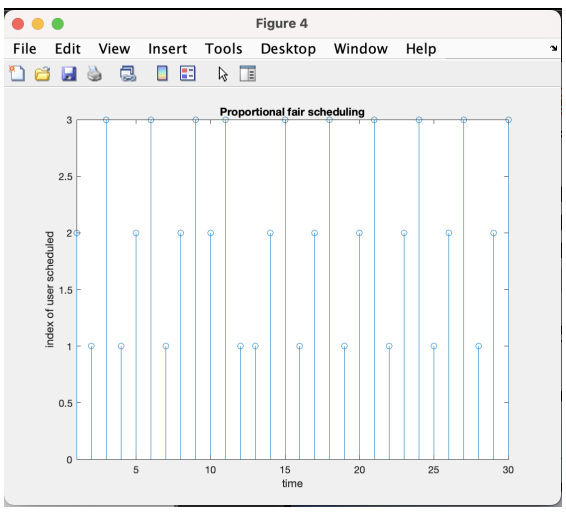
10번의 단위 slot 동안

user1은 1번,

user2는 3번,

user3는 6번 할당 받았다.

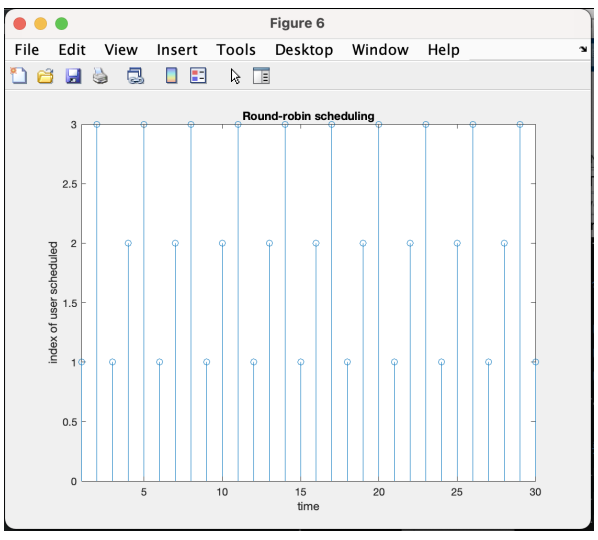
12bit*1s = 4bit*3s = 2bit*6s = 12bps로, tput이 같아지도록 할당되었다.



proportional fair

대체적으로 주기성을 띄며 user1, user2, user3가 교대하여 slot을 할당 받는다.

PF 방식의 수식에 의해, 효율과 형평성을 고려하여 할당하므로 세 user들은 대체로 교대로 스케줄링된다.



round-robin

user1, user2, user3은 정확히 교대하여 slot을 할당 받는다.

case4 - (R1, R2, R3) = (12, 3, 1)

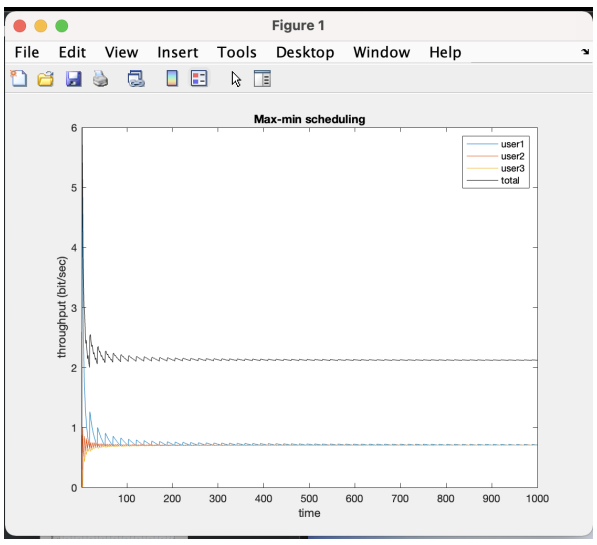
R1 = 12

α = R1/R2 = 12/3 = 4

β = R1/R3 = 12/1 = 12

```
avg_th =
```

```
    0.7080    0.7050    0.7050    2.1180
    3.9960    0.9990    0.3330    5.3280
    4.0080    0.9960    0.3330    5.3370


scheduling_probability =

    0.0590    0.2350    0.7050
    0.3330    0.3330    0.3330
    0.3340    0.3320    0.3330

>>
```



max-min

user1의 tput은 약 0.71bps,

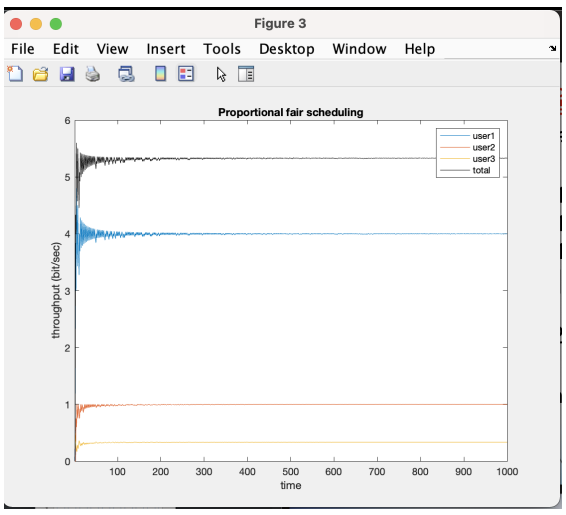
user2의 tput은 약 0.71bps,

user3의 tput은 약 0.71bps

으로 수렴했다.

1:1:1이다.

전체 tput은 약 2.12bps로, PF 방식과 RR 방식 대비 절반 정도의 처리율이다.



proportional fair

user1의 tput은 약 4bps,

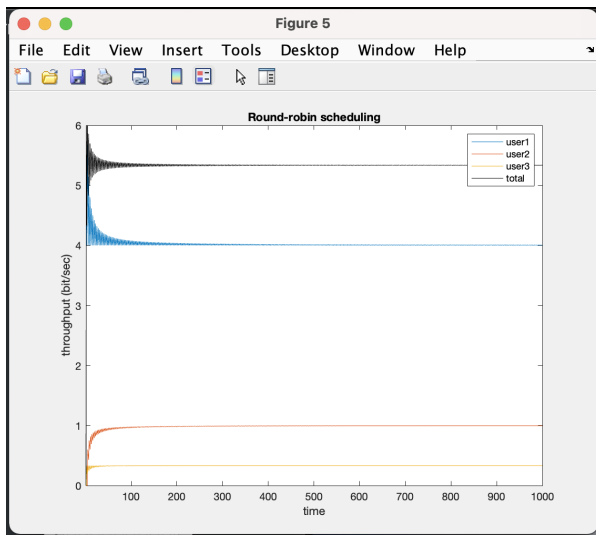
user2의 tput은 약 1bps,

user3의 tput은 약 0.33bps

으로 수렴했다.

약 12:3:1이다.

전체 tput은 약 5.33bps이다.



round-robin

user1의 tput은 약 4.01bps,

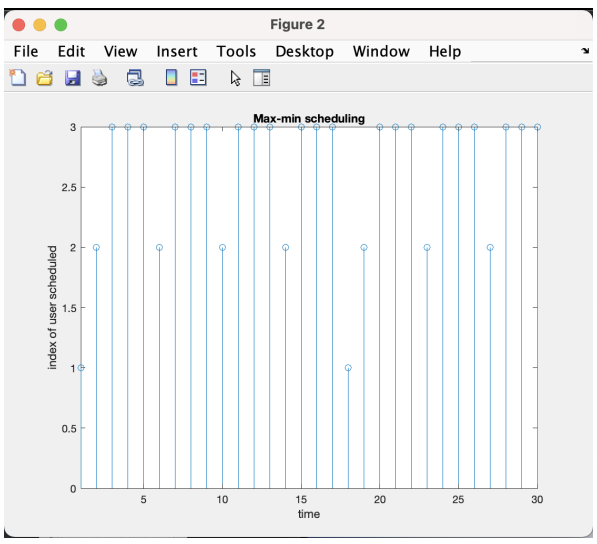
user2의 tput은 약 1bps,

user3의 tput은 약 0.33bps

으로 수렴했다.

약 12:3:1이다.

전체 tput은 약 5.34bps으로, PF 방식과 비슷한 처리율이다.



max-min

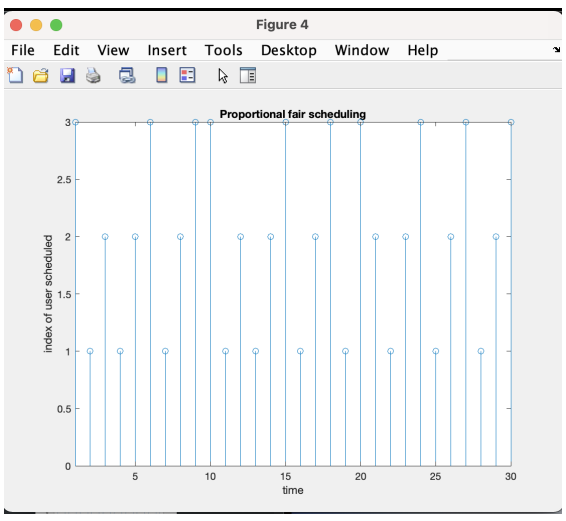
17번의 단위 slot 동안

user1은 1번,

user2는 4번,

user3는 12번 할당 받았다.

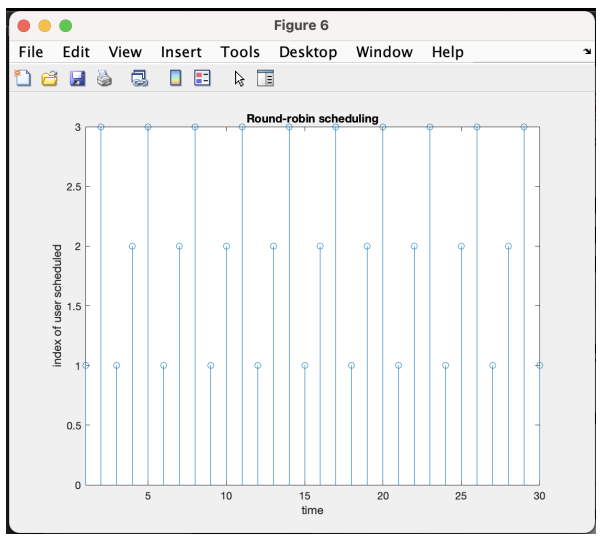
12bit*1s = 3bit*4s = 1bit*12s = 12bps로, tput이 같아지도록 할당되었다.



proportional fair

대체적으로 주기성을 띄며 user1, user2, user3가 교대하여 slot을 할당 받는다.

PF 방식의 수식에 의해, 효율과 형평성을 고려하여 할당하므로 세 user들은 대체로 교대로 스케줄링된다.



round-robin

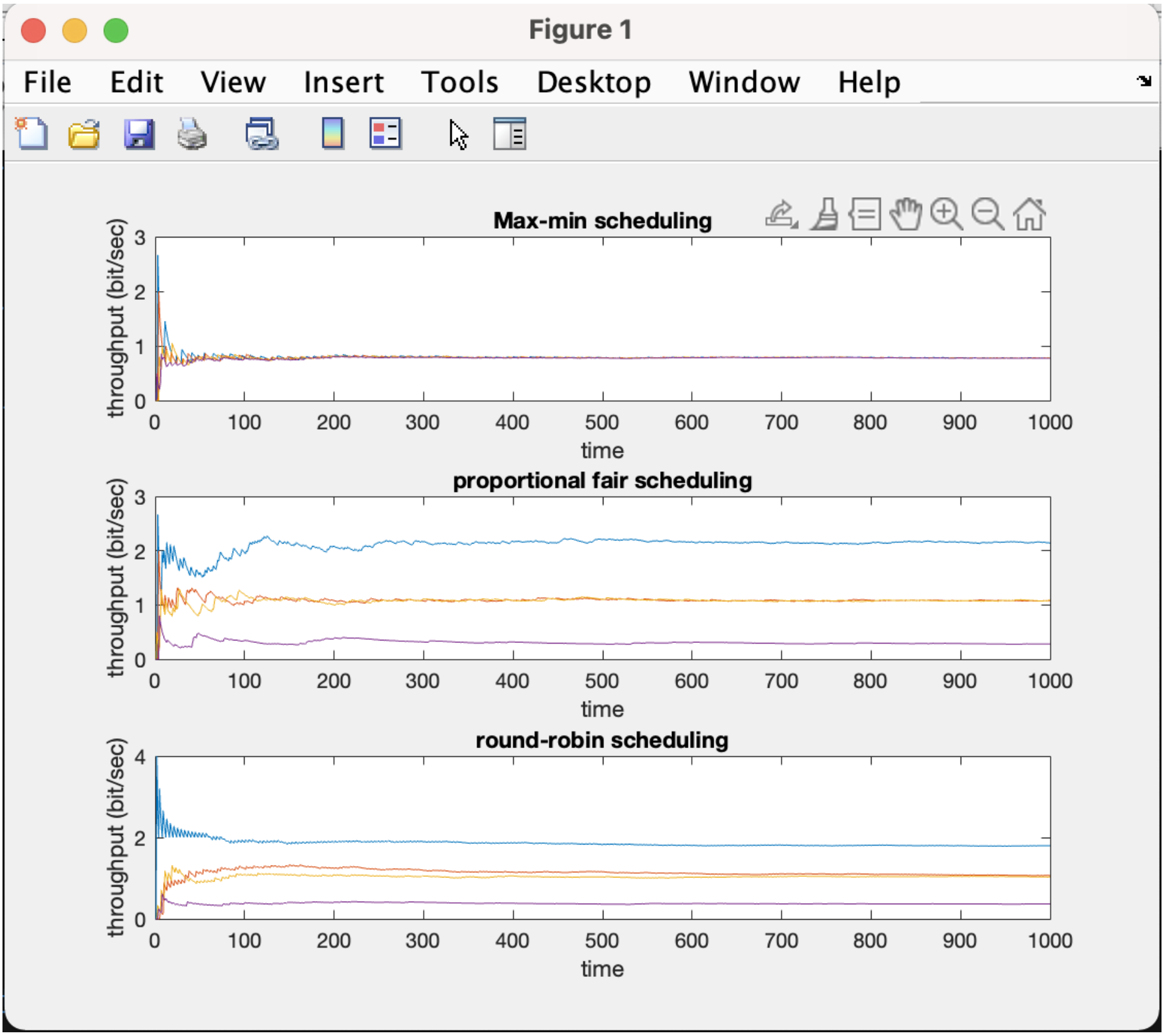
user1, user2, user3은 정확히 교대하여 slot을 할당 받는다.

scheduling_lab2.m - MCS 확률적 분포

```
%-----------------------------------------
% INC4103: 스케줄링 실습2
% MCS가 확률적으로 변경되는 경우
%-----------------------------------------

% 3가지 스케줄링 정책 고려함
% case1: max-min
% case2: proportional fair
% case3: round-robin

% 하나의 시간 슬롯은 한 명의 사용자에게 할당된다고 가정

clear;
close all;

N_slot = 1000;    % 전체 슬롯 수
N_user = 4;       % 사용자 수

% 변수 선언 & 초기화
% 각 스케줄링 기법별 사용자별 누적 전송 데이터 양
bytes_sent_case1 = zeros(1,N_user);
bytes_sent_case2 = zeros(1,N_user);
bytes_sent_case3 = zeros(1,N_user);

% 각 스케줄링 기법별 사용자별 평균 처리율
% (행,열)=(시간,사용자)
avg_th_case1 = zeros(N_slot,N_user);
avg_th_case2 = zeros(N_slot,N_user);
avg_th_case3 = zeros(N_slot,N_user);

% 매 슬롯별 스케줄링 결과 선택된 사용자 인덱스
user_index_case1 = zeros(N_slot,1);
user_index_case2 = zeros(N_slot,1);
user_index_case3 = zeros(N_slot,1);

% 초기값은 random하게 설정함
user_index_case1(1) = ceil(rand*N_user);
user_index_case2(1) = ceil(rand*N_user);
user_index_case3(1) = ceil(rand*N_user);

% 사용자별 할당된 (스케줄링된) 슬롯의 합계
counter_user_index_case1 = zeros(1,N_user);
counter_user_index_case2 = zeros(1,N_user);
counter_user_index_case3 = zeros(1,N_user);

% 전송 속도는 채널 상태에 따라 다음중 하나로 결정됨
TX_RATE = [8, 4, 1];
% 사용자1: 채널 상태 좋음
% 사용자2&3: 채널 상태 보통
% 사용자4: 채널 상태 나쁨
channel_quality_prob = [0.8   0.2   0.0;  % 사용자1
                        0.2   0.6   0.2;  % 사용자2
                        0.2   0.6   0.2;  % 사용자3
                        0.0   0.2   0.8];  % 사용자4

for i = 2:N_slot

 % 각 사용자별 MCS 결정
 % 매 슬롯시간마다 확률적으로 변경된다고 가정함
 for j=1:N_user
     p = rand;
     if (p < channel_quality_prob(j,1))
         inst_rate(j) = TX_RATE(1);
     elseif (p < channel_quality_prob(j,1)+channel_quality_prob(j,2))
         inst_rate(j) = TX_RATE(2);
     else
         inst_rate(j) = TX_RATE(3);
     end
 end

% case1: max-min
for j=1:N_user
    if (j==user_index_case1(i-1))
        % 스케줄링 결과 슬롯을 할당받은 사용자
        % 전송 데이터 양을 전송속도만큼 증가시키고
        % 할당받은 슬롯수를 1 증가
        bytes_sent_case1(j) =  bytes_sent_case1(j)+inst_rate(j);
        counter_user_index_case1(j) = counter_user_index_case1(j)+1;
    end
    % 사용자별 처리율 업데이트
    avg_th_case1(i,:) = (bytes_sent_case1)/i;
    % 다음 슬롯 시간의 스케줄링 사용자 결정
```

```
        [temp user_index_case1(i)] = min(avg_th_case1(i,:));
        % max-min은 평균 처리율이 가장 낮은 사용자를 선택함
        % 만약, 같은 최소값을 가진다면 그 중 첫번째
     end

    % case2: proportional fair
    for j=1:N_user
        if (j==user_index_case2(i-1))
             bytes_sent_case2(j) =  bytes_sent_case2(j)+inst_rate(j);
             counter_user_index_case2(j) = counter_user_index_case2(j)+1;
        end
        avg_th_case2(i,:) = max((bytes_sent_case2)/i,eps);
        % 평균 처리율이 0이면 나눗셈 연산시 오류가 발생하여
        % 아주 작은값 (eps)로 바꿈
    end
    [temp user_index_case2(i)] = max(inst_rate./avg_th_case2(i,:));
    % PF 스케줄링은 전송속도/평균처리율값이 가장 큰 사용자 선택

    % case3: round-robin
    for j=1:N_user
         if (j==user_index_case3(i-1))
             bytes_sent_case3(j) =  bytes_sent_case3(j)+inst_rate(j);
             counter_user_index_case3(j) = counter_user_index_case3(j)+1;
         end
         avg_th_case3(i,:) = (bytes_sent_case3)/i;
    end
    user_index_case3(i) = mod(i,N_user)+1;
       % RR 스케줄링은 순서대로 한번씩
       % mod함수 이용

    end

    % 사용자별 최종 처리율
    final_per_user_th = [avg_th_case1(N_slot,:);
                         avg_th_case2(N_slot,:);
                         avg_th_case3(N_slot,:)]
    final_total_th = sum(final_per_user_th')'

    % scheduling probability
    scheduling_prob = [counter_user_index_case1 / N_slot;
                       counter_user_index_case2 / N_slot;
                       counter_user_index_case3 / N_slot]

    figure;
    subplot(3,1,1)
    plot(avg_th_case1);
    xlabel('time')
    ylabel('throughput (bit/sec)');
    title('Max-min scheduling');

    subplot(3,1,2)
    plot(avg_th_case2);
    xlabel('time')
    ylabel('throughput (bit/sec)');
    title('proportional fair scheduling');


    subplot(3,1,3)
    plot(avg_th_case3);
    xlabel('time')
    ylabel('throughput (bit/sec)');
    title('round-robin scheduling');
```

case0 - 사용자 4명의 채널 품질이 각각 좋음, 보통, 보통, 나쁨




```
final_per_user_th =

    0.7760    0.7760    0.7800    0.7750
    2.1400    1.0750    1.0900    0.2800
    1.7960    1.0660    1.0330    0.3670


final_total_th =

    3.1070
    4.5850
    4.2620


scheduling_prob =

    0.1080    0.1780    0.1870    0.5260
    0.2980    0.2610    0.2680    0.1720
    0.2500    0.2490    0.2500    0.2500
```

매 슬롯마다 전송 속도가 8, 4, 1일 확률을 다음과 같이 설정하였다.

user 1: 0.8, 0.2, 0.0 → 채널 상태 좋음, blue line

user 2: 0.2, 0.6, 0.2 → 채널 상태 보통, orange line

user 3: 0.2, 0.6, 0.2 → 채널 상태 보통, yellow line

user 4: 0, 0.2, 0.8 → 채널 상태 나쁨, purple line

max-min

user1, user2, user3, user4 모두 tput이 약 0.78bps로 수렴하였다.

네 user들의 tput 총합은 약 3.1070bps이다.

형평성(평균 처리율)을 고려하여 스케줄링되었으므로, 네 user들의 tput은 거의 동일하다.

proportional fair

user1의 tput은 약 2.1400bps(round robin보다 큼),

user2의 tput은 약 1.0750bps,

user3의 tput은 약 1.0900bps,

user4의 tput은 약 0.2800bps(round robin보다 작음) 으로 수렴하였다.

네 user들의 tput 총합은 약 4.585bps(round robin보다 큼)이다.

scheduling probability는 [0.2980 0.2610 0.2680 0.1720]이다.

채널 상태가 좋은 user1이 많이 할당되었고,

채널 상태가 나쁜 user4가 적게 할당되었다.

Round Robin

user1의 tput은 약 1.7960bps,

user2의 tput은 약 1.0660bps,

user3의 tput은 약 1.0330bps,

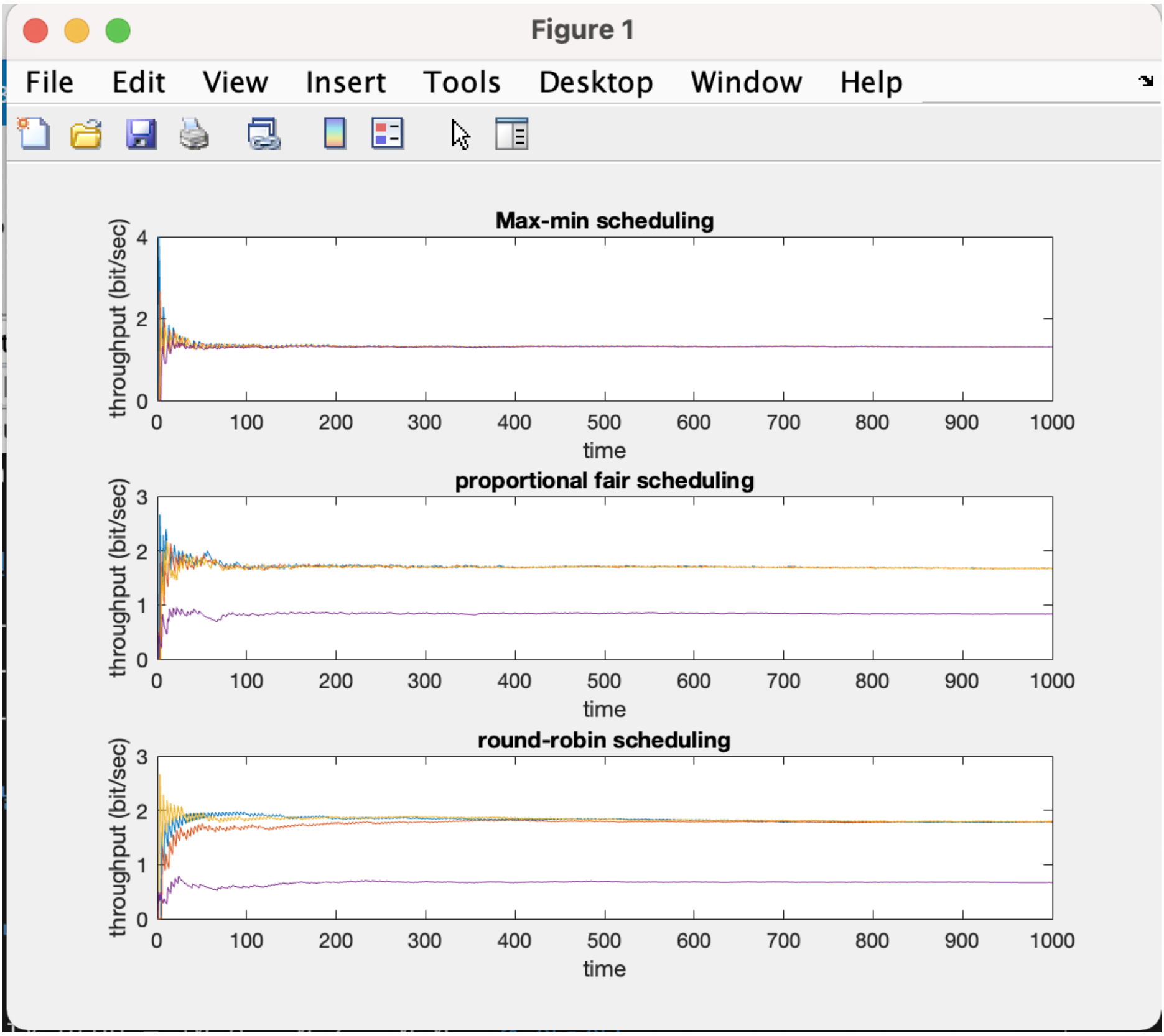
user4의 tput은 약 0.3670bps 로 수렴하였다.

네 user들의 tput 총합은 약 4.262bps이다.

단말의 전송 속도(채널 상태)와 무관하게 모든 단말에게 동일한 시간이 할당되었다. user1은 전송 속도가 8일 확률이 높고, user2와 user3은 전송 속도가 4일 확률이 높고, user4는 전송 속도가 1일 확률이 높다. 따라서 단말의 처리율은 전송 속도에 대체로 비례하게 나온 것이 관측된다.

case1 - 대부분 사용자의 채널 품질이 우수함

- 사용자1/2/3: 0.8, 0.2, 0.0 → 채널 상태 좋음
- 사용자4: 0, 0.6, 0.4 → 체널 상태 보통 이하




```
final_per_user_th =

    1.2920    1.2840    1.2840    1.2850
    1.6480    1.6520    1.6480    0.8240
    1.7840    1.7640    1.7960    0.7000


final_total_th =

    5.1450
    5.7720
    6.0440


scheduling_prob =

    0.1820    0.1830    0.1770    0.4570
    0.2380    0.2320    0.2270    0.3020
    0.2500    0.2490    0.2500    0.2500

>>
```

max-min

user1, user2, user3, user4 모두 tput이 약 1.29bps로 수렴하였다.

네 user들의 tput 총합은 약 5.1450bps이다.

형평성(평균 처리율)을 고려하여 스케줄링되었으므로, 네 user들의 tput은 모두 동일하다.

proportional fair

user1의 tput은 약 1.65bps,

user2의 tput은 약 1.65bps,

user3의 tput은 약 1.65bps,

user4의 tput은 약 0.82bps 로 수렴하였다.

네 user들의 tput 총합은 약 5.77bps이다.

scheduling probability는 [0.2380 0.2320 0.2270 0.3020]이다.

평균 처리율이 낮은 user4에게 할당이 많이 되었다.

Round Robin

user1의 tput은 약 1.78bps,

user2의 tput은 약 1.76bps,

user3의 tput은 약 1.80bps,

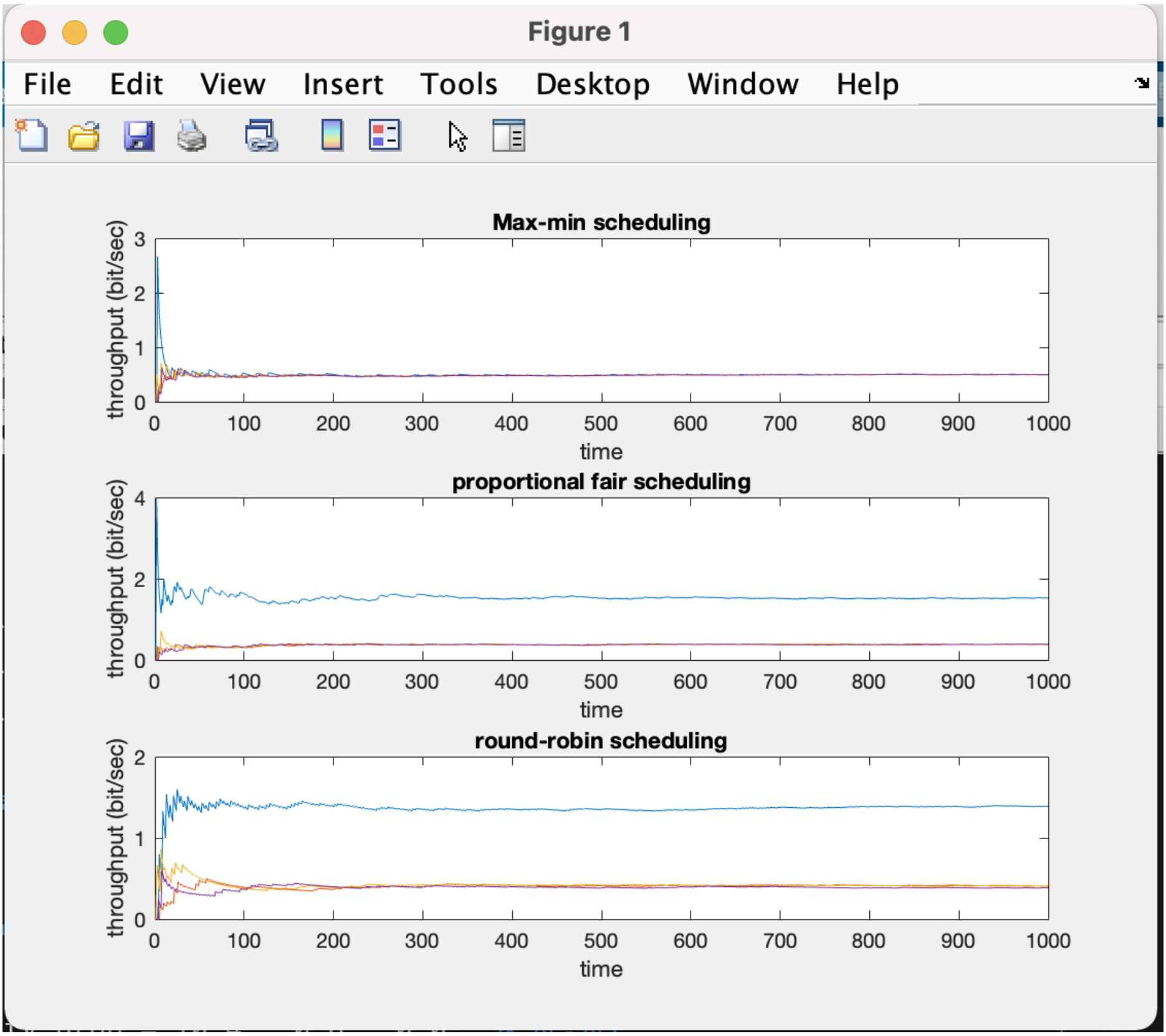
user4의 tput은 약 0.70bps 로 수렴하였다.

네 user들의 tput 총합은 약 6.04bps이다.

단말의 전송 속도(채널 상태)와 무관하게 모든 단말에게 동일한 시간이 할당되었다. user1, user2, user3는 전송 속도가 8일 확률이 높고, user4는 전송 속도가 4일 확률이 높다. 따라서 단말의 처리율은 전송 속도에 대체로 비례하게 나온 것이 관측된다.

case2 - 대부분 사용자의 채널 품질이 열악함

- 사용자1 : 0.4, 0.6, 0.0 → 채널 상태 보통 이상
- 사용자2/3/4: 0, 0.2, 0.8 → 채널 상태 나쁨




```
final_per_user_th =

    0.4960    0.4950    0.4930    0.4930
    1.5120    0.3770    0.3790    0.3810
    1.3880    0.4050    0.4030    0.4160


final_total_th =

    1.9770
    2.6490
    2.6120


scheduling_prob =

    0.0850    0.3030    0.2980    0.3130
    0.2730    0.2360    0.2560    0.2340
    0.2490    0.2490    0.2500    0.2510

>>
```

max-min

user1, user2, user3, user4 모두 tput이 약 0.495bps로 수렴하였다.

네 user들의 tput 총합은 약 1.98bps이다.

형평성(평균 처리율)을 고려하여 스케줄링되었으므로, 네 user들의 tput은 모두 동일하다.

proportional fair

user1의 tput은 약 1.51bps,

user2의 tput은 약 0.38bps,

user3의 tput은 약 0.38bps,

user4의 tput은 약 0.38bps 으로 수렴하였다.

네 user들의 tput 총합은 약 2.65bps이다.

scheduling probability는 [0.2730 0.2360 0.2560 0.2340] 이다.

채널 상태가 좋은 user1이 많이 할당되었다.

Round Robin

user1의 tput은 약 1.39bps,

user2의 tput은 약 0.41bps,

user3의 tput은 약 0.40bps,

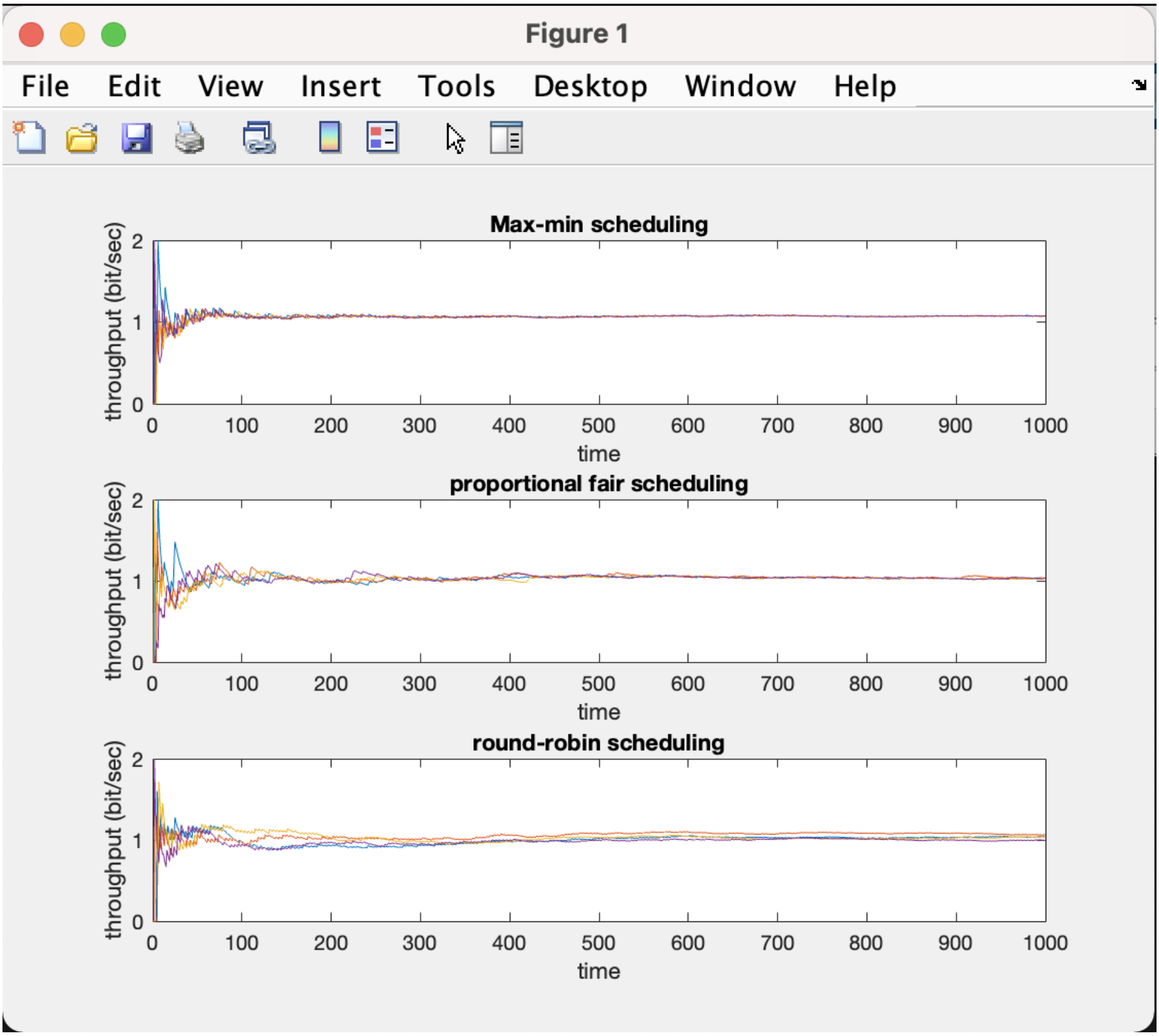
user4의 tput은 약 0.42bps 로 수렴하였다.

네 user들의 tput 총합은 약 2.61bps이다.

단말의 전송 속도(채널 상태)와 무관하게 모든 단말에게 동일한 시간이 할당되었다. user1은 전송 속도가 4일 확률이 높고, user2와 user3, user4는 전송 속도가 1일 확률이 높다. 따라서 단말의 처리율은 전송 속도에 대체로 비례하게 나온 것이 관측된다.

case3 - 모든 사용자의 채널 품질의 차이가 크지 않음

- 사용자1/2/3/4: 0.2, 0.6, 0.2 → 채널 상태 보통




```
final_per_user_th =

    1.0580    1.0550    1.0530    1.0560
    1.0240    1.0390    1.0280    1.0220
    1.0470    1.0510    1.0800    1.0700


final_total_th =

    4.2220
    4.1130
    4.2480


scheduling_prob =

    0.2490    0.2540    0.2530    0.2430
    0.2390    0.2560    0.2540    0.2500
    0.2490    0.2490    0.2510    0.2500

>>
```

max-min

user1, user2, user3, user4 모두 tput이 약 1.06bps로 수렴하였다.

네 user들의 tput 총합은 약 4.22bps이다.

형평성(평균 처리율)을 고려하여 스케줄링되었으므로, 네 user들의 tput은 모두 동일하다.

proportional fair

user1의 tput은 약 1.02bps,

user2의 tput은 약 1.04bps,

user3의 tput은 약 1.03bps,

user4의 tput은 약 1.02bps 로 수렴하였다.

네 user들의 tput 총합은 약 4.11bps이다.

scheduling probability는 [0.2390 0.2560 0.2540 0.2500] 이다.

네 user는 모두 채널 상태가 비슷하기 때문에 거의 균등하게 할당을 받았다.

Round Robin

user1의 tput은 약 1.05bps,

user2의 tput은 약 1.05bps,

user3의 tput은 약 1.08bps,

user4의 tput은 약 1.07bps 로 수렴하였다.

네 user들의 tput 총합은 약 4.25bps이다.

단말의 전송 속도(채널 상태)와 무관하게 모든 단말에게 동일한 시간이 할당되었다. 따라서 네 user 모두 tput이 거의 같게 수렴한다.

scheduling_lab3.m - AMC

```
%-----------------------------------------
% INC4103: 스케줄링 실습3
% 채널 상태 변경되고 AMC 고려함
%-----------------------------------------

% 4가지 스케줄링 정책 고려함
% case1: max-min
% case2: proportional fair
% case3: round-robin
% case4: max-SNR

% 하나의 시간 슬롯은 한 명의 사용자에게 할당된다고 가정

clear;
close all;

N_slot = 1000;    % 전체 슬롯 수
N_user = 4;       % 사용자 수
distance = [100 200 300 400]; % 기지국과의 거리

% SNR값에 따라 4가지 MCS 고려
SNR_TH = [30 20 10];     % unit: dB
TX_RATE = [16 8 4 1]; % unit: bit/sec
% if SNR>=30 dB      --> 16 bit/sec (64QAM 2/3)
% if 20<= SNR <30 dB --> 8 bit/sec (16QAM 1/2)
% if 10<= SNR <20 dB --> 4 bit/sec (QPSK 1/2)
% if SNR < 10 dB     --> 1 bit/sec (BPSK 1/2)

% 채널 관련 파라미터
P_TX = 20;      % 전송 전력 (dBm)
alpha = 3.5;    % path-loss exponent
PL_0 = 40;       % 기준거리(D_0)에서의 경로 손실
D_0 = 10;        % 기준거리
SHADOW_STD = 10;  % shadowing 표준편차;
NOISE = -80;     % 잡음 전력 (dBm)

% 변수 선언 & 초기화
% 각 스케줄링 기법별 사용자별 누적 전송 데이터 양
bytes_sent_case1 = zeros(1,N_user);
bytes_sent_case2 = zeros(1,N_user);
bytes_sent_case3 = zeros(1,N_user);
bytes_sent_case4 = zeros(1,N_user);

% 각 스케줄링 기법별 사용자별 평균 처리율
% (행,열)=(시간,사용자)
avg_th_case1 = zeros(N_slot,N_user);
avg_th_case2 = zeros(N_slot,N_user);
avg_th_case3 = zeros(N_slot,N_user);
```

```
avg_th_case4 = zeros(N_slot,N_user);

% 매 슬롯별 스케줄링 결과 선택된 사용자 인덱스
user_index_case1 = zeros(N_slot,1);
user_index_case2 = zeros(N_slot,1);
user_index_case3 = zeros(N_slot,1);
user_index_case4 = zeros(N_slot,1);

% 초기값은 random하게 설정함
user_index_case1(1) = ceil(rand*N_user);
user_index_case2(1) = ceil(rand*N_user);
user_index_case3(1) = ceil(rand*N_user);
user_index_case4(1) = ceil(rand*N_user);

% 사용자별 할당된 (스케줄링된) 슬롯의 합계
counter_user_index_case1 = zeros(1,N_user);
counter_user_index_case2 = zeros(1,N_user);
counter_user_index_case3 = zeros(1,N_user);
counter_user_index_case4 = zeros(1,N_user);

for i = 2:N_slot

    % 각 사용자별 MCS 결정
    % 매 슬롯시간마다 채널 상태 변경된다고 가정
    % SNR값 계산 : path-loss와 shadowing 고려
    % 1) path-loss model: path-loss exponent = alpha
    % 2) shadowing: log-normal distribution, STD = SHADOWING
 for j = 1:N_user
     path_loss = PL_0 + 10*alpha*log10(distance(j)/D_0);
     shadowing = randn*SHADOW_STD;
     p_rx(j) = P_TX - (path_loss + shadowing);
     snr(j) = p_rx(j) - NOISE;  % dB

     for k = 1:length(SNR_TH)
         if (snr(j) >= SNR_TH(k))
             inst_rate(i,j) = TX_RATE(k);
             break;
         end
     end
     if (snr(j) < min(SNR_TH))
         inst_rate(i,j) = min(TX_RATE);
     end
 end

% case1: max-min
for j=1:N_user
    if (j==user_index_case1(i-1))
        % 스케줄링 결과 슬롯을 할당받은 사용자
        % 전송 데이터 양을 전송속도만큼 증가시키고
        % 할당받은 슬롯수를 1 증가
        bytes_sent_case1(j) =  bytes_sent_case1(j)+inst_rate(i,j);
        counter_user_index_case1(j) = counter_user_index_case1(j)+1;
    end
    % 사용자별 처리율 업데이트
    avg_th_case1(i,:) = (bytes_sent_case1)/i;
    % 다음 슬롯 시간의 스케줄링 사용자 결정
    [temp user_index_case1(i)] = min(avg_th_case1(i,:));
    % max-min은 평균 처리율이 가장 낮은 사용자를 선택함
    % 만약, 같은 최소값을 가진다면 그 중 첫번째
 end

% case2: proportional fair
for j=1:N_user
    if (j==user_index_case2(i-1))
         bytes_sent_case2(j) =  bytes_sent_case2(j)+inst_rate(i,j);
         counter_user_index_case2(j) = counter_user_index_case2(j)+1;
    end
    avg_th_case2(i,:) = max((bytes_sent_case2)/i,eps);
end
[temp user_index_case2(i)] = max(inst_rate(i,:)./avg_th_case2(i,:));
% PF 스케줄링은 전송속도/평균처리율값이 가장 큰 사용자 선택

% case3: round-robin
for j=1:N_user
     if (j==user_index_case3(i-1))
         bytes_sent_case3(j) =  bytes_sent_case3(j)+inst_rate(i,j);
         counter_user_index_case3(j) = counter_user_index_case3(j)+1;
     end
     avg_th_case3(i,:) = (bytes_sent_case3)/i;
end
user_index_case3(i) = mod(i,N_user)+1;
% RR 스케줄링은 순서대로 한번씩
% mod함수 이용

% case4: max-SNR
for j=1:N_user
     if (j==user_index_case4(i-1))
```

```
            bytes_sent_case4(j) =  bytes_sent_case4(j)+inst_rate(i,j);
            counter_user_index_case4(j) = counter_user_index_case4(j)+1;
        end
        avg_th_case4(i,:) = (bytes_sent_case4)/i;
   end
   [temp user_index_case4(i)] = max(snr);
   % max-SNR 스케줄링은 SNR값이 가장 큰 사용자 선택

end

% 각 사용자별 전송속도 분포
% rate_dist: (행,열)=(사용자,해당 MCS의 분포확률)
for j=1:N_user
    rate_dist(j,:) = [sum(inst_rate(:,j)==TX_RATE(1)),
                      sum(inst_rate(:,j)==TX_RATE(2)),
                      sum(inst_rate(:,j)==TX_RATE(3)),
                      sum(inst_rate(:,j)==TX_RATE(4))]/N_slot;
end

rate_dist

% final throughput per user
% 사용자별 최종 처리율
final_per_user_th = [avg_th_case1(N_slot,:);
                     avg_th_case2(N_slot,:);
                     avg_th_case3(N_slot,:);
                     avg_th_case4(N_slot,:)]
final_total_th = sum(final_per_user_th')'

% scheduling probability
scheduling_prob = [counter_user_index_case1 / N_slot;
                   counter_user_index_case2 / N_slot;
                   counter_user_index_case3 / N_slot;
                   counter_user_index_case4 / N_slot]
figure;
subplot(2,2,1);
plot(avg_th_case1);
xlabel('time')
ylabel('throughput (bit/sec)');
title('Max-min scheduling');
grid; axis([0 N_slot 0 15]);

subplot(2,2,2);
plot(avg_th_case2);
xlabel('time')
ylabel('throughput (bit/sec)');
title('proportional fair scheduling');
grid; axis([0 N_slot 0 15]);

subplot(2,2,3);
plot(avg_th_case3);
xlabel('time')
ylabel('throughput (bit/sec)');
title('round-robin scheduling');
grid; axis([0 N_slot 0 15]);

subplot(2,2,4);
plot(avg_th_case4);
xlabel('time')
ylabel('throughput (bit/sec)');
title('max-SNR scheduling');
grid; axis([0 N_slot 0 15]);
```

SNR 값에 따른 AMC(Adaptive Modulation and Coding)을 고려한, 더욱 실제적인 환경으로 실험을 반복하였다.

채널 모델은 다음과 같다.

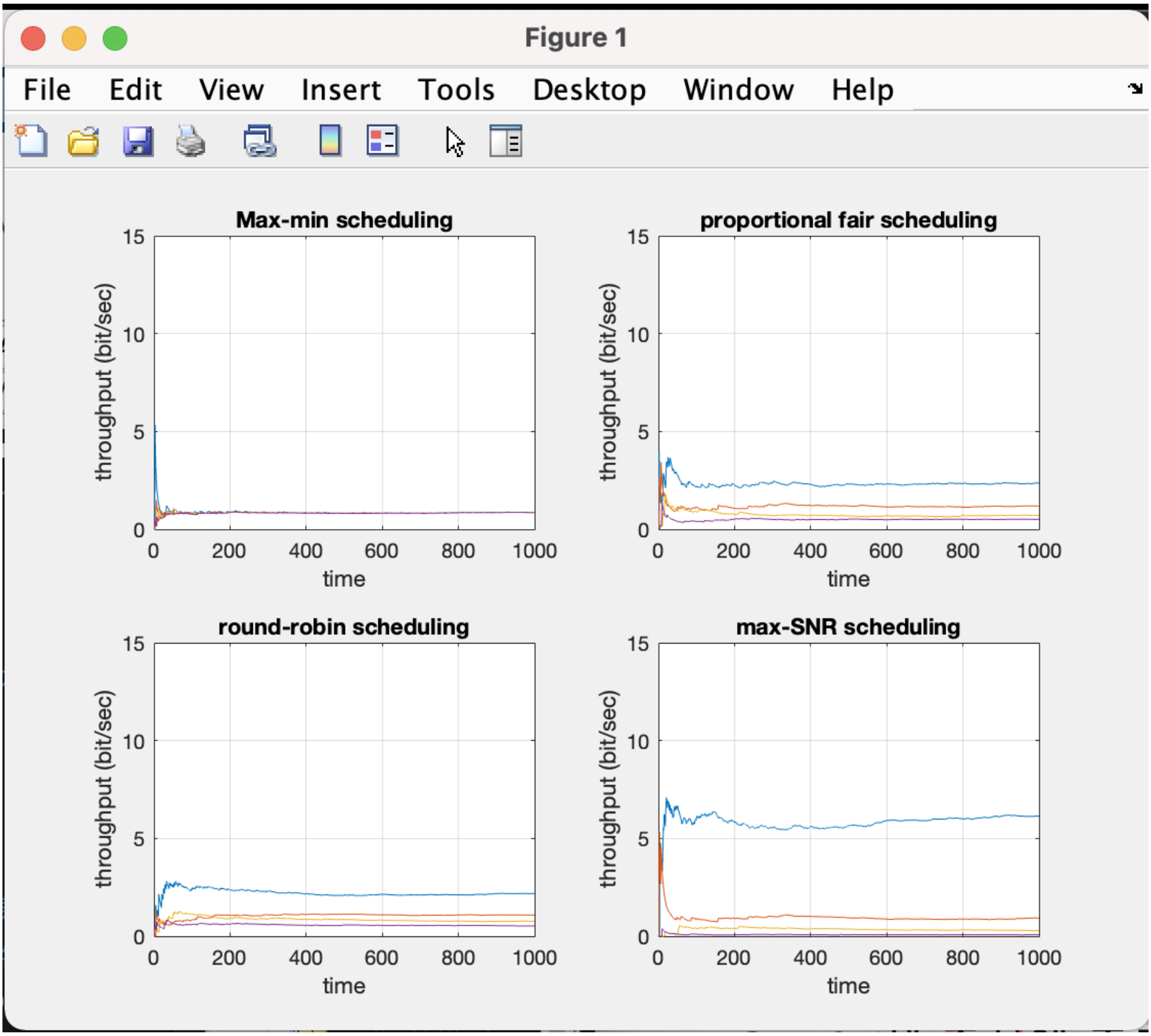
– Path-loss: 40 dB at d=10m, path-loss exponent: 3.5
– Shadowing: Log-normal distribution (mean = 0, STD = 10dB)
– RX power (dBm) = TX power (20dBm) – (path loss + N(0,10))
– Noise power = -80 dBm
– SNR (dB) = RX power – noise power

AMC는 다음과 같다.

MCS는 64QAM 2/3, 16QAM 1/2, QPSK 1/2, BPSK 1/4이다.

이들의 Data rate는 6*2/3=4, 4*1/2=2, 2*1/2=1, 1*1/4=1/4이므로 간단한 정수비로 나타내면 16, 8, 4, 1이다.

Case1 (사용자의 균일 분포)
– 사용자와 BS 거리 = 100, 200, 300, 400




```
rate_dist =

    0.3080    0.3820    0.2630    0.0460
    0.0540    0.2340    0.3730    0.3380
    0.0150    0.1050    0.3160    0.5630
    0.0040    0.0510    0.1940    0.7500


final_per_user_th =

    0.8850    0.8850    0.8820    0.8790
    2.2870    1.1540    0.6900    0.5400
    2.2080    1.1600    0.7250    0.5190
    6.5920    0.8510    0.1660    0.0780


final_total_th =

    3.5310
    4.6710
    4.6120
    7.6870


scheduling_prob =

    0.0920    0.1920    0.2900    0.4250
```

```
    0.2510    0.2550    0.2400    0.2530
    0.2490    0.2490    0.2500    0.2510
    0.7120    0.1860    0.0650    0.0360

>>
```

max-min

네 user의 tput은 모두 약 0.88bps로 수렴한다. 총 tput은 약 3.53bps이다. 모든 단말기의 처리율이 같아지도록 스케줄링한다. RR, PF, max-SNR 방식에 비해 처리율이 낮다.

proportional fair

user1의 tput은 약 2.29bps,

user2의 tput은 약 1.15bps,

user3의 tput은 약 0.69bps,

user4의 tput은 약 0.54bps로 수렴한다.

총 tput은 약 4.67bps이다. RR 방식 보다 전체 tput이 더 높다.

round-robin

user1의 tput은 약 2.21bps,

user2의 tput은 약 1.16bps,

user3의 tput은 약 0.73bps,

user4의 tput은 약 0.52bps로 수렴한다.

총 tput은 약 4.61bps이다.

max-SNR

user1의 tput은 약 6.59bps,

user2의 tput은 약 0.85bps,

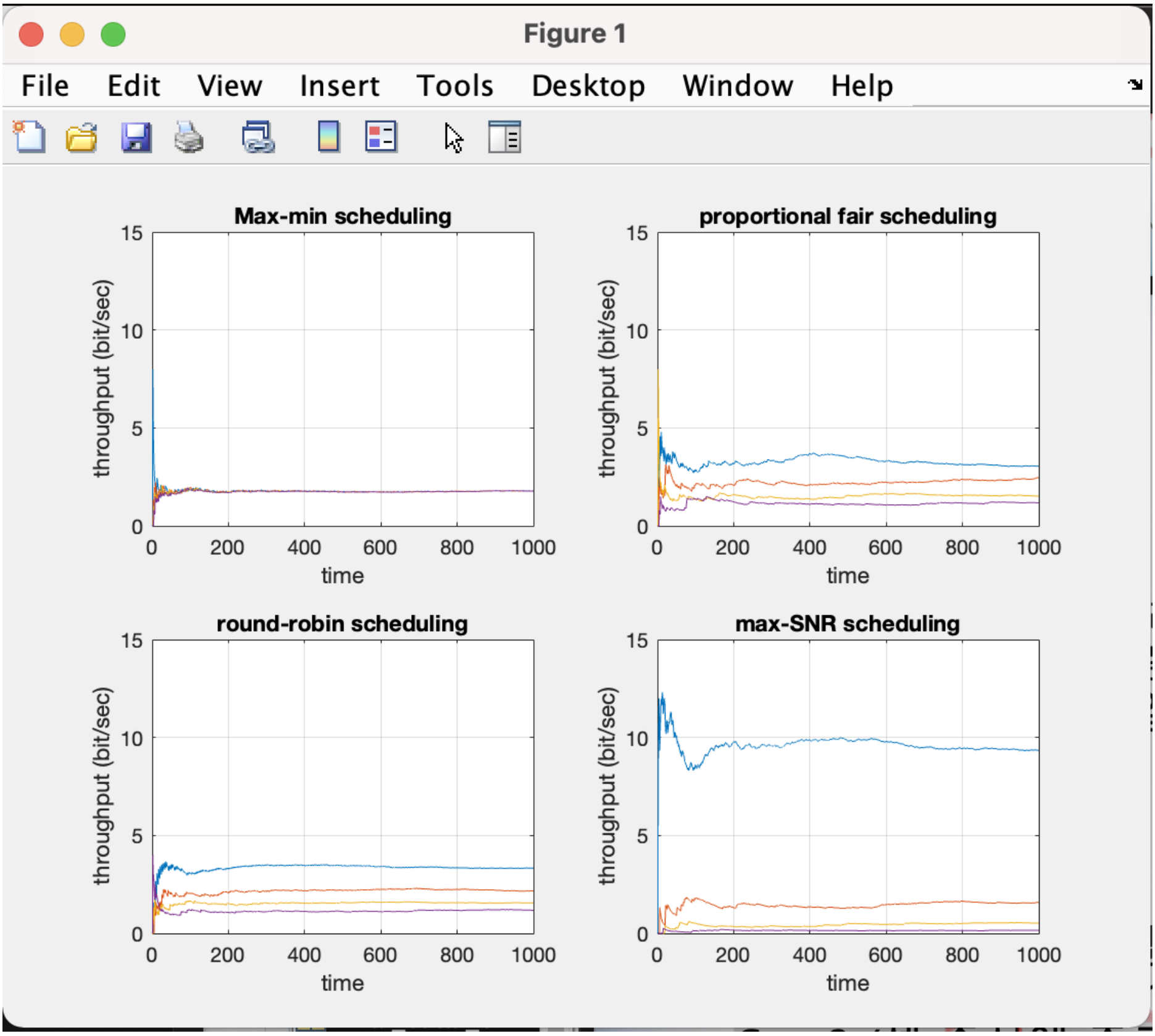
user3의 tput은 약 0.17bps,

user4의 tput은 약 0.08bps로 수렴한다.

총 tput은 약 7.69bps이다. max-SNR은 매 슬롯시간마다 가장 높은 SNR을 가진 사용자에게 할당하므로, 전체 처리율을 최대화하는 방향으로 스케줄링한다. 따라서 RR과 PF 방식보다 전체 처리율이 더 높다.

Case2 (셀 중심에 주로 분포)

– 사용자와 BS 거리 = 50, 100, 150, 200




```
rate_dist =

    0.7060    0.2210    0.0650    0.0070
    0.3190    0.3790    0.2320    0.0690
    0.1410    0.3070    0.3630    0.1880
    0.0690    0.2310    0.3950    0.3040


final_per_user_th =

    1.8400    1.8340    1.8260    1.8340
    3.0230    2.3930    1.4710    1.2110
    3.3330    2.1840    1.4720    1.1720
    9.1230    1.8330    0.3710    0.1860


final_total_th =

    7.3340
    8.0980
    8.1610
   11.5130


scheduling_prob =

    0.1500    0.1920    0.3010    0.3560
    0.2280    0.2650    0.2520    0.2540
    0.2500    0.2490    0.2500    0.2500
    0.6850    0.1990    0.0710    0.0440

>>
```

max-min

네 user의 tput은 모두 약 1.83bps로 수렴한다. 총 tput은 약 7.3340bps이다. 모든 단말기의 처리율이 같아지도록 스케줄링한다. RR, PF, max-SNR 방식에 비해 처리율이 낮다.

proportional fair

user1의 tput은 약 3.02bps,

user2의 tput은 약 2.39bps,

user3의 tput은 약 1.47bps,

user4의 tput은 약 1.21bps로 수렴한다.

총 tput은 약 8.10bps이다.

round-robin

user1의 tput은 약 3.33bps,

user2의 tput은 약 2.18bps,

user3의 tput은 약 1.47bps,

user4의 tput은 약 1.17bps로 수렴한다.

총 tput은 약 8.16bps이다.

max-SNR

user1의 tput은 약 9.12bps,

user2의 tput은 약 1.83bps,

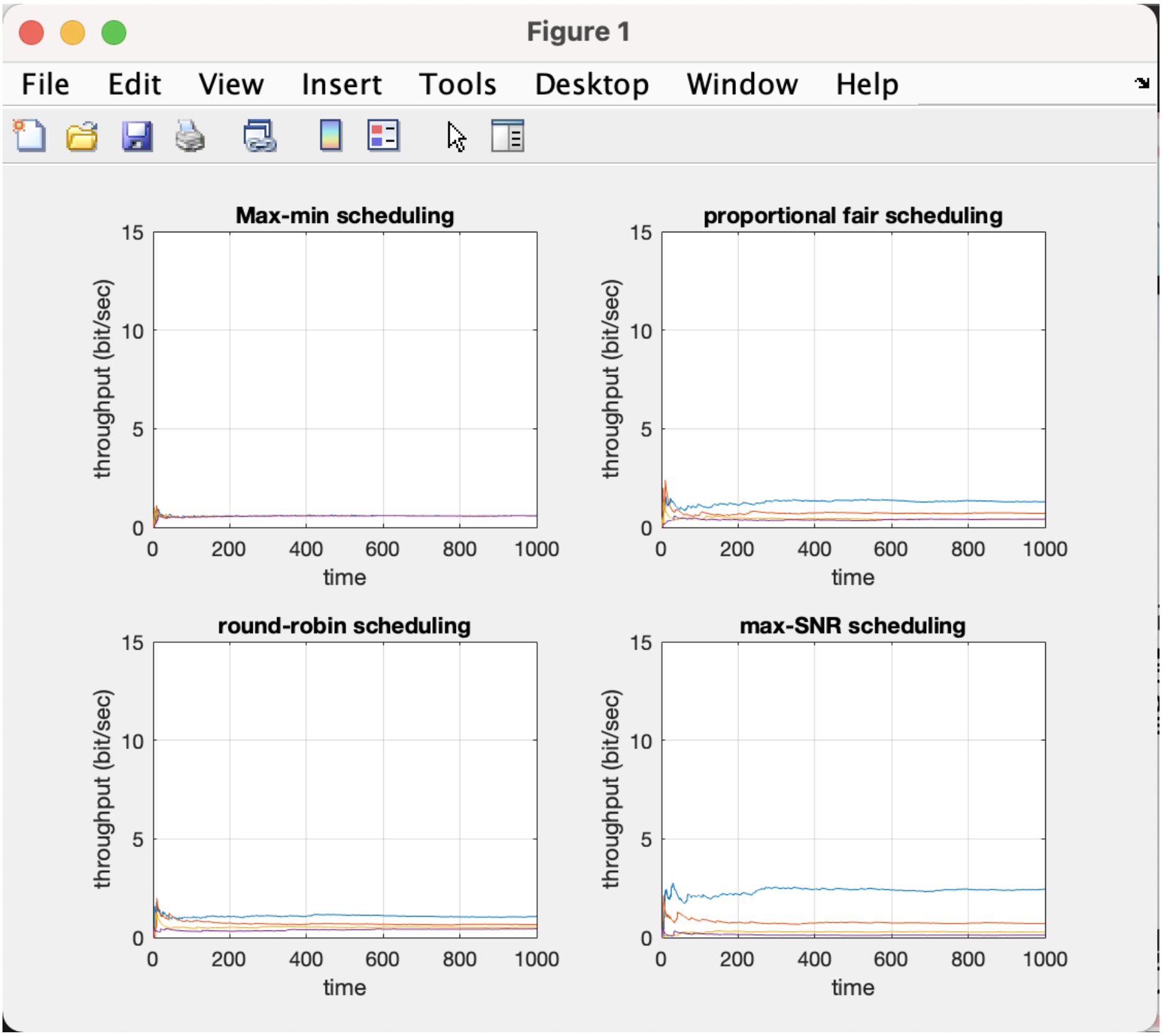
user3의 tput은 약 0.37bps,

user4의 tput은 약 0.19bps로 수렴한다.

총 tput은 약 11.51bps이다.

max-SNR은 매 슬롯시간마다 가장 높은 SNR을 가진 사용자에게 할당하므로, 전체 처리율을 최대화하는 방향으로 스케줄링한다. 따라서 RR과 PF 방식보다 전체 처리율이 더 높다.

Case3 (셀 외곽에 주로 분포)
– 사용자와 BS 거리 = 200, 300, 400, 500




```
rate_dist =

    0.0530    0.2460    0.3750    0.3250
    0.0150    0.1120    0.3360    0.5360
    0.0020    0.0480    0.1960    0.7530
    0.0010    0.0300    0.1440    0.8240


final_per_user_th =

    0.6110    0.5970    0.5970    0.5970
    1.3530    0.7730    0.3950    0.3680
    1.0550    0.7150    0.4610    0.4040
    2.4650    0.8290    0.1790    0.1310


final_total_th =

    2.4020
    2.8890
    2.6350
    3.6040


scheduling_prob =

    0.1240    0.1920    0.3210    0.3620
    0.2910    0.2610    0.2150    0.2320
    0.2500    0.2490    0.2500    0.2500
    0.5390    0.2750    0.1090    0.0760

>>
```

max-min

세 user의 tput은 모두 약 0.60bps로 수렴한다. 총 tput은 약 2.40bps이다. 이는 RR 방식과 PF 방식의 절반 정도의 처리율이다. 모든 단말기의 처리율이 같아지도록 스케줄링한다.

proportional fair

user1의 tput은 약 1.35bps,

user2는 약 0.77bps,

user3는 약 0.40bps,

user4는 약 0.37bps로 수렴한다.

총 tput은 약 2.8890bps이다.

round-robin

user1의 tput은 약 1.06bps,

user2는 약 0.72bps,

user3는 약 0.46bps,

user4는 약 0.40bps로 수렴한다.

총 tput은 약 2.64bps이다.

max-SNR

user1의 tput은 약 2.47bps,

user2는 약 0.83bps,

user3는 약 0.18bps,

user4는 약 0.13bps로 수렴한다.

총 tput은 약 3.60bps이다. max-SNR은 매 슬롯시간마다 가장 높은 SNR을 가진 사용자에게 할당하므로, 전체 처리율을 최대화하는 방향으로 스케줄링한다.

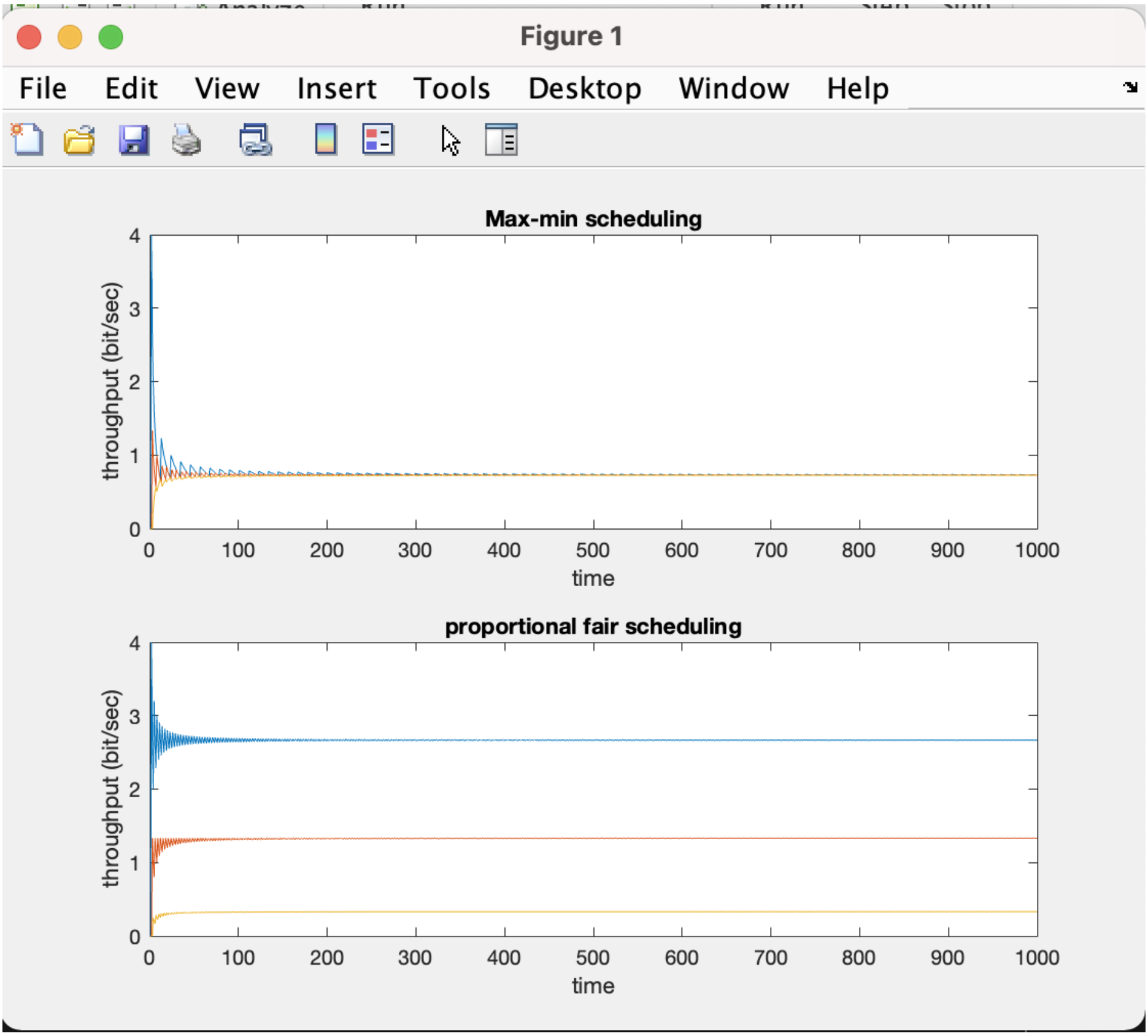
scheduling_lab4.m - QoS

```
%-----------------------------------------
% INC4103: 스케줄링 실습4
% 사용자별 최소 요구 전송 속도가 있는 경우
%-----------------------------------------

% 3가지 스케줄링 정책 고려함
% case1: max-min
% case2: proportional fair

% 하나의 시간 슬롯은 한 명의 사용자에게 할당된다고 가정

clear;
close all;
clc;


N_slot = 1000;    % 전체 슬롯 수
N_user = 3;       % 사용자 수

% 변수 선언 & 초기화
% 각 스케줄링 기법별 사용자별 누적 전송 데이터 양
bytes_sent_case1 = zeros(1,N_user);
bytes_sent_case2 = zeros(1,N_user);

% 각 스케줄링 기법별 사용자별 평균 처리율
% (행,열)=(시간,사용자)
avg_th_case1 = zeros(N_slot,N_user);
avg_th_case2 = zeros(N_slot,N_user);

% 매 슬롯별 스케줄링 결과 선택된 사용자 인덱스
user_index_case1 = zeros(N_slot,1);
user_index_case2 = zeros(N_slot,1);
```

```
% 초기값은 random하게 설정함
user_index_case1(1) = ceil(rand*N_user);
user_index_case2(1) = ceil(rand*N_user);

% 사용자별 할당된 (스케줄링된) 슬롯의 합계
counter_user_index_case1 = zeros(1,N_user);
counter_user_index_case2 = zeros(1,N_user);

% 전송 속도는 채널 상태에 따라 다음중 하나로 결정됨
TX_RATE = [8, 4, 1];
% 사용자1: 채널 상태 좋음
% 사용자2&3: 채널 상태 보통
% 사용자4: 채널 상태 나쁨
channel_quality_prob = [1.0   0.0   0.0;  % 사용자1
                        0.0   1.0   0.0;  % 사용자2
                        0.0   0.0   1.0]; % 사용자3


% minimum required tput
% R_min = [0 0 0];
% R_min = [0.5 0.5 0.5];
R_min = [0.8 0.4 0.1];
% R_min = [1 1 1];

for i = 2:N_slot

 % 각 사용자별 MCS 결정
 % 매 슬롯시간마다 확률적으로 변경된다고 가정함
 for j=1:N_user
     p = rand;
     if (p < channel_quality_prob(j,1))
         inst_rate(j) = TX_RATE(1);
     elseif (p < channel_quality_prob(j,1)+channel_quality_prob(j,2))
         inst_rate(j) = TX_RATE(2);
     else
         inst_rate(j) = TX_RATE(3);
     end
 end

% case1: max-min
for j=1:N_user
    if (j==user_index_case1(i-1))
        % 스케줄링 결과 슬롯을 할당받은 사용자
        % 전송 데이터 양을 전송속도만큼 증가시키고
        % 할당받은 슬롯수를 1 증가
        bytes_sent_case1(j) =  bytes_sent_case1(j)+inst_rate(j);
        counter_user_index_case1(j) = counter_user_index_case1(j)+1;
    end
    % 사용자별 처리율 업데이트
    avg_th_case1(i,:) = (bytes_sent_case1)/i;
    % 다음 슬롯 시간의 스케줄링 사용자 결정
    [temp user_index_case1(i)] = min(avg_th_case1(i,:)-R_min);
    % max-min은 평균 처리율이 가장 낮은 사용자를 선택함
    % 만약, 같은 최소값을 가진다면 그 중 첫번째
 end

% case2: proportional fair
for j=1:N_user
    if (j==user_index_case2(i-1))
        bytes_sent_case2(j) =  bytes_sent_case2(j)+inst_rate(j);
        counter_user_index_case2(j) = counter_user_index_case2(j)+1;
      end
      avg_th_case2(i,:) = max((bytes_sent_case2)/i,eps);
      % 평균 처리율이 0이면 나눗셈 연산시 오류가 발생하여
      % 아주 작은값 (eps)로 바꿈
   end
   [temp user_index_case2(i)] = max(inst_rate./max((avg_th_case2(i,:)-R_min), eps));
   % PF 스케줄링은 전송속도/평균처리율값이 가장 큰 사용자 선택
end

% 사용자별 최종 처리율
final_per_user_th = [avg_th_case1(N_slot,:);
                     avg_th_case2(N_slot,:)]
final_total_th = sum(final_per_user_th')'

% scheduling probability
scheduling_prob = [counter_user_index_case1 / N_slot;
                   counter_user_index_case2 / N_slot]

figure;
subplot(2,1,1)
plot(avg_th_case1);
xlabel('time')
ylabel('throughput (bit/sec)');
title('Max-min scheduling');
```

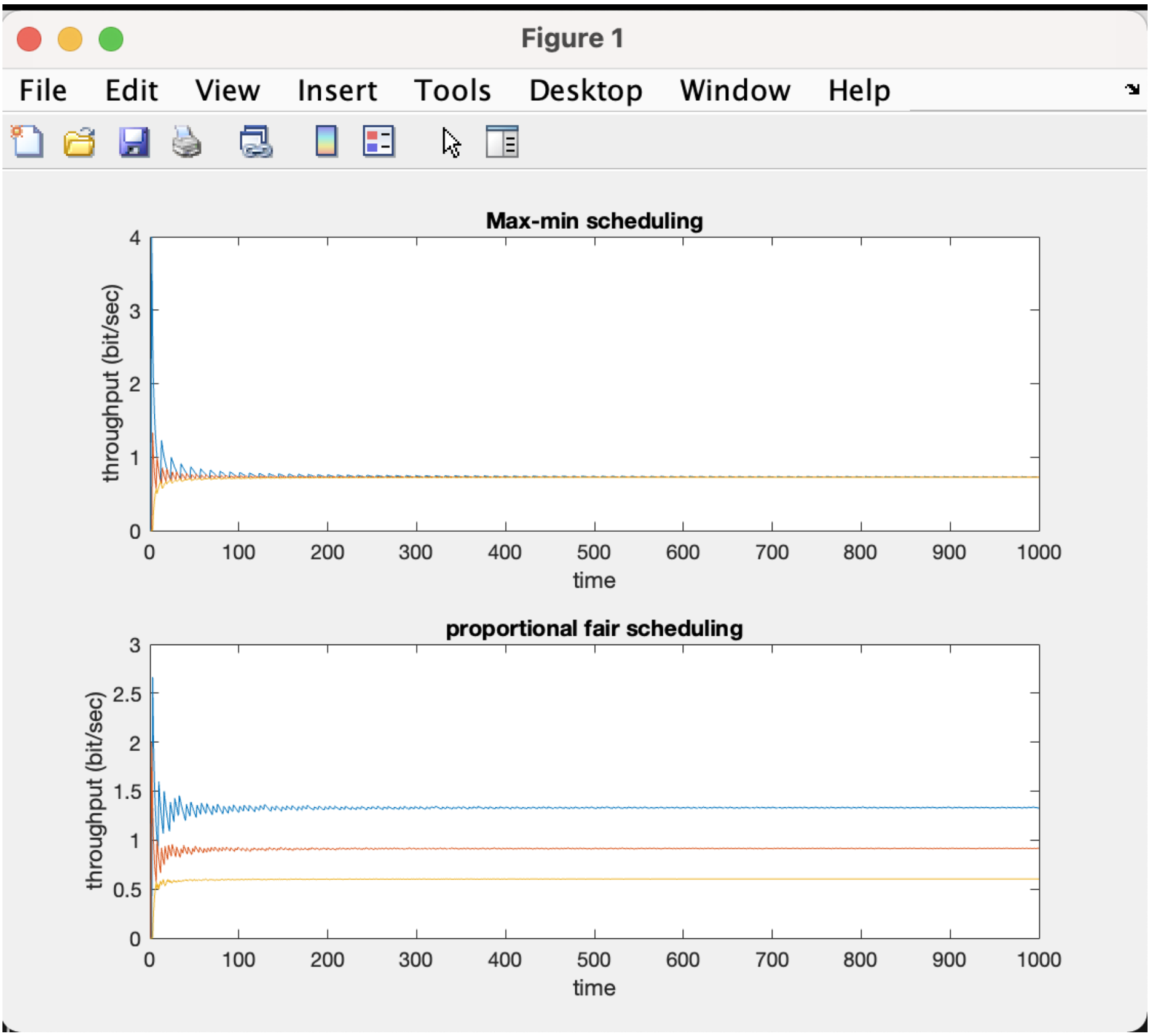
```
subplot(2,1,2)
plot(avg_th_case2);
xlabel('time')
ylabel('throughput (bit/sec)');
title('proportional fair scheduling');
```

case1: R_min = [0 0 0]



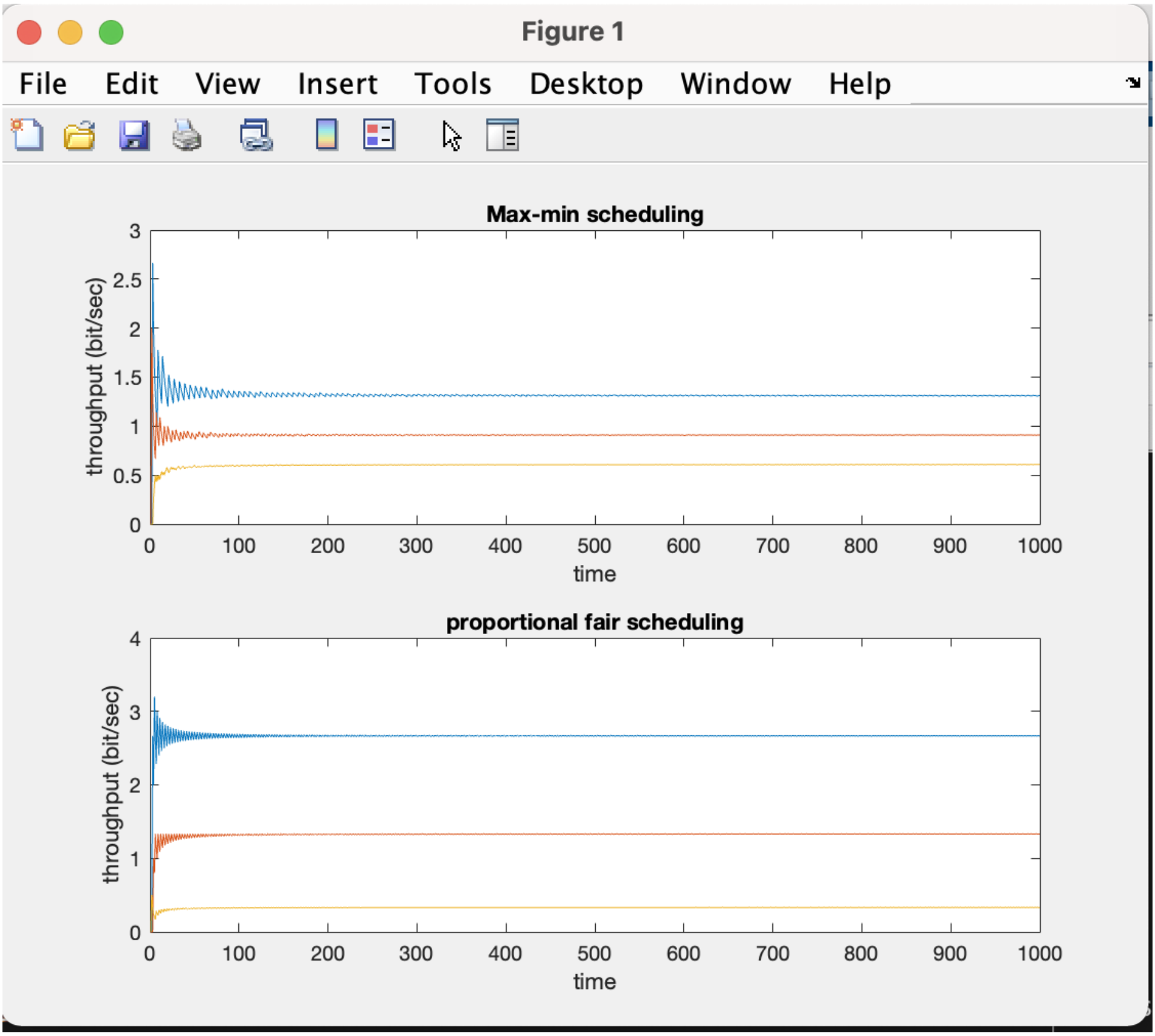

```
final_per_user_th =

    0.7280    0.7280    0.7260
    % (8bit*1)/(1+2+8)s = (4bit*2)/(1+2+8)s = (1bit*8)/(1+2+8)s = 0.7273bps
    2.6640    1.3320    0.3330
    % [8 4 1]/3 = [2.6667  1.333  0.333] = 8:4:1


final_total_th =

    2.1820
    4.3290


scheduling_prob =

    0.0910    0.1820    0.7260 % 1/8:1/4:1/1 = 1:2:8
    0.3330    0.3330    0.3330 % 1:1:1
```

case2: R_min = [0.5 0.5 0.5]




```
final_per_user_th =

    0.7280    0.7280    0.7260
    % (8bit*1)/(1+2+8)s = (4bit*2)/(1+2+8)s = (1bit*8)/(1+2+8)s = 0.7273bps
    1.3337    0.9169    0.6041
    % [1.3337 0.9169 0.6041] - 0.5 = [0.8337 0.4169 0.1041] => 8:4:1
    % 0.5씩을 보장해준 뒤 tput이 8:4:1로 분배됨
    % user1, user2의 할당이 적어졌음


final_total_th =

    2.1820
    2.8547 % user1, user2의 할당이 적어졌기 때문에 tput 감소


scheduling_prob =

    0.0910    0.1820    0.7260
    % 1/8:1/4:1/1 = 1:2:8
    0.1667    0.2292    0.6041
    % 초반에 0.5씩 보장해준 뒤 교대로 할당
    % user1, user2의 할당이 적어짐
```

case3: R_min = [0.8 0.4 0.1]




```
final_per_user_th =

    1.3097    0.9093    0.6090
    % [1.3097 0.9093 0.6090] - [0.8 0.4 0.1] = [0.5 0.5 0.5] => 1:1:1
    % 초반에 R_min = [0.8 0.4 0.1]을 보장해준 뒤 균등 분배(1:1:1)
    2.6665    1.3333    0.3334
    % [8 4 1]/3 = [2.6667  1.333  0.333]


final_total_th =

    2.8280
    4.3332


scheduling_prob =

    0.1637    0.2273    0.6090
    0.3333    0.3333    0.3333 % 1:1:1
```

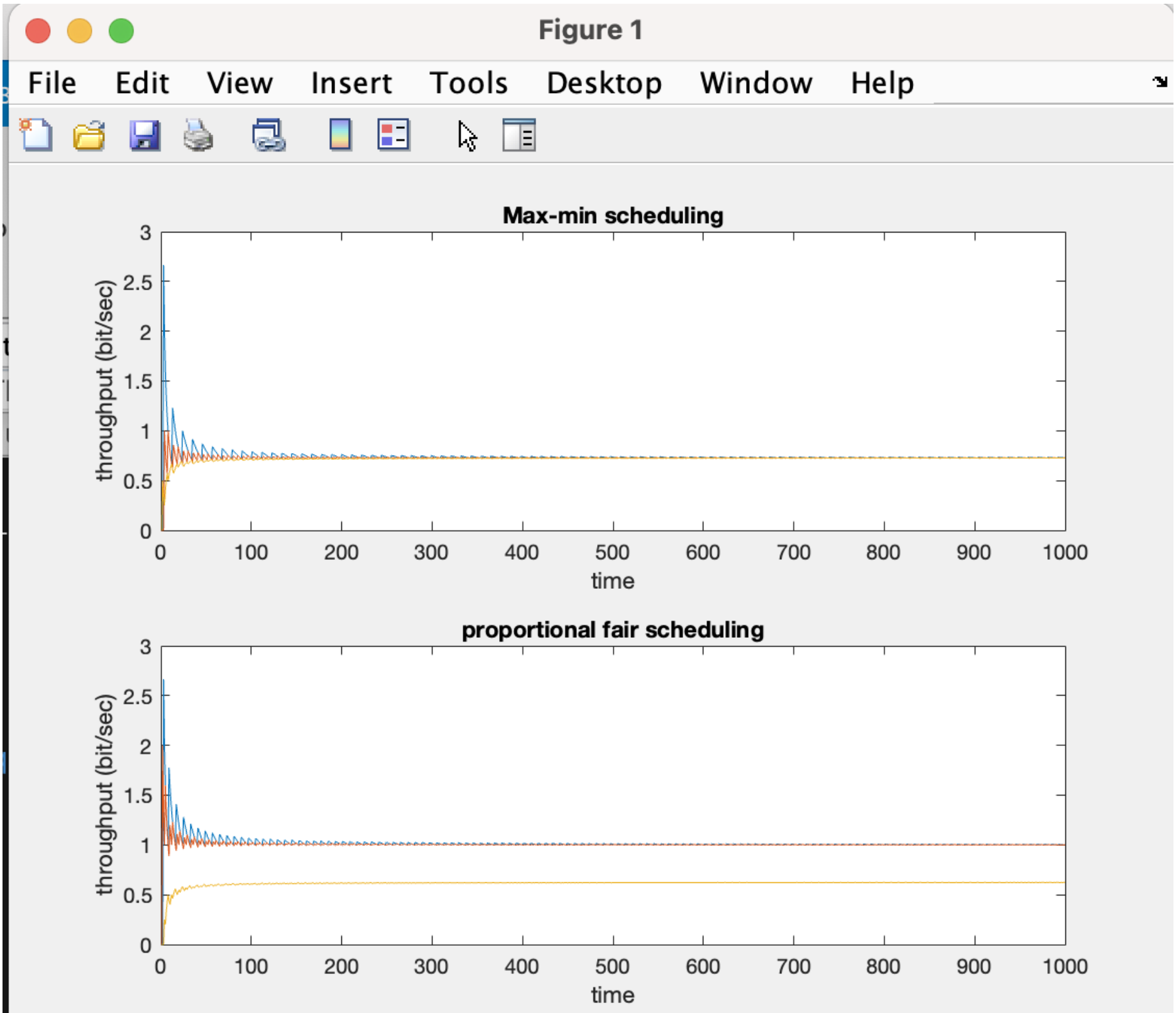
case4: R_min = [1 1 1]

사용자의 요구 전송 속도가 없을 때

max-min에서 세 user의 tput은 [0.7 0.7 0.7] 이였고,

proportional fair에서 세 user의 tput은 [2.6 1.3 0.3]이였음.

따라서 R_min = [1 1 1]일 때 일부 사용자의 요구 전송 속도를 만족할 수 없는 경우도 발생함.




```
final_per_user_th =

    0.7280    0.7280    0.7260
    % (8bit*1)/(1+2+8)s = (4bit*2)/(1+2+8)s = (1bit*8)/(1+2+8)s = 0.7273bps
    1.0000    1.0000    0.6240
    % user1과 user2는 요구량 충족했지만, user3은 충족시키지 못했음

final_total_th =

    2.1820
    2.6240


scheduling_prob =

    0.0910    0.1820    0.7260 % 1/8:1/4:1/1 = 1:2:8
    0.1250    0.2500    0.6240
    % user1,2,3가 교대로 할당받다가 user1이 충족되면 user2,3가 교대로 할당 받고, user2도 충족된 이후 user3가 할당되었음
```