11.25 목 - [MATLAB] random backoff lab report

실습 1.1 - Contention window의 영향 분석

Nuser = 10

CWmin = 16

```
total_th =

   12.4488

fairness_index =

    0.9995

collision_prob =

    0.6744

utilization =

    0.5135

>>
```

CWmin = 32

```
total_th =

   16.9128

fairness_index =

    0.9997

collision_prob =

    0.4288

utilization =

    0.6977

>>
```

CWmin = 64

```
total_th =

   18.8280

fairness_index =
```

```
    0.9999


collision_prob =

    0.2470


utilization =

    0.7767

>>
```

CWmin = 128

```
total_th =

   18.6616


fairness_index =

    0.9998


collision_prob =

    0.1304


utilization =

    0.7698

>>
```

CWmin = 256

```
total_th =

   16.6464


fairness_index =

    0.9999


collision_prob =

    0.0686


utilization =

    0.6867

>>
```

CWmin = 512

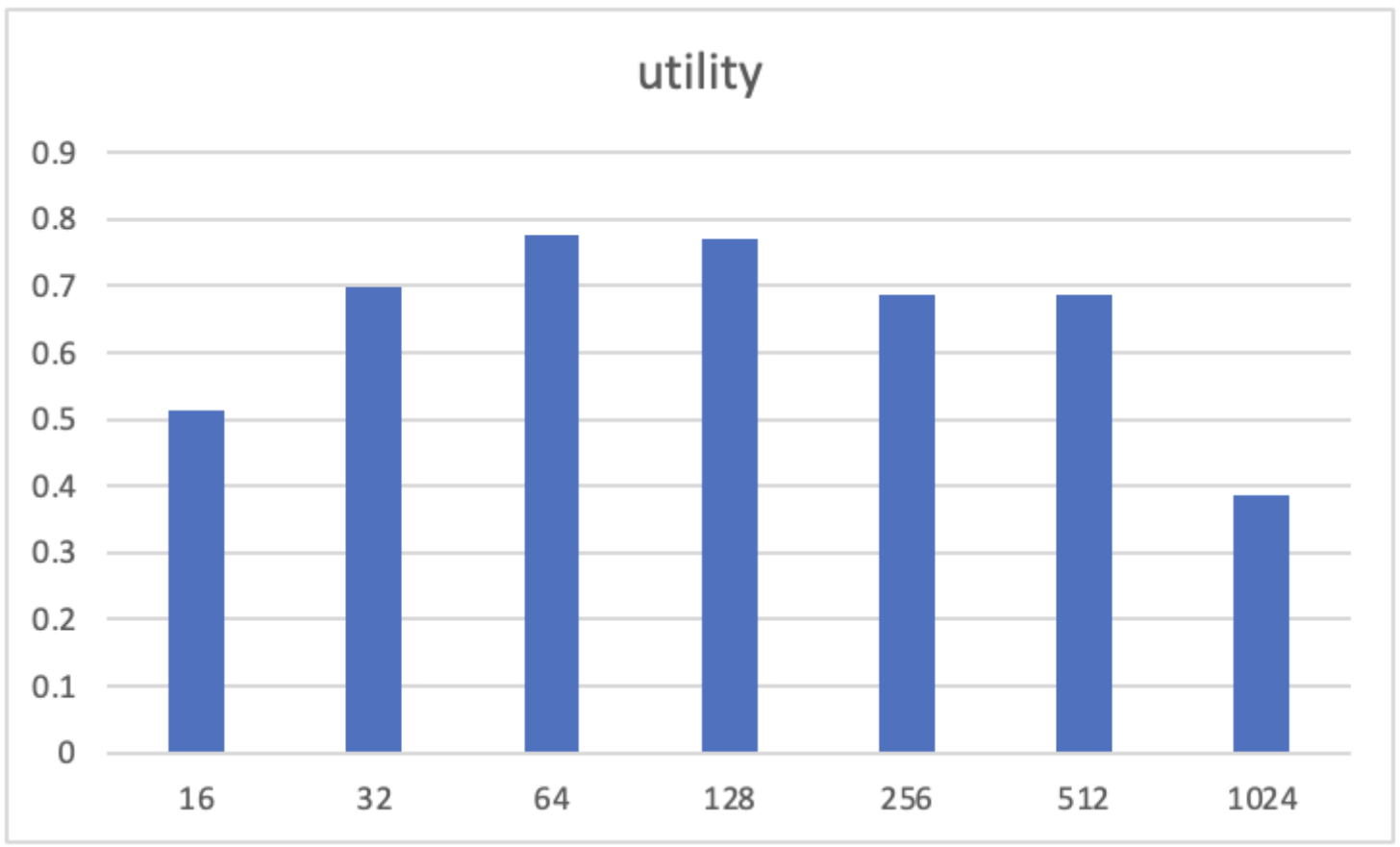
```
total_th =

   16.6464


fairness_index =

    0.9999
```

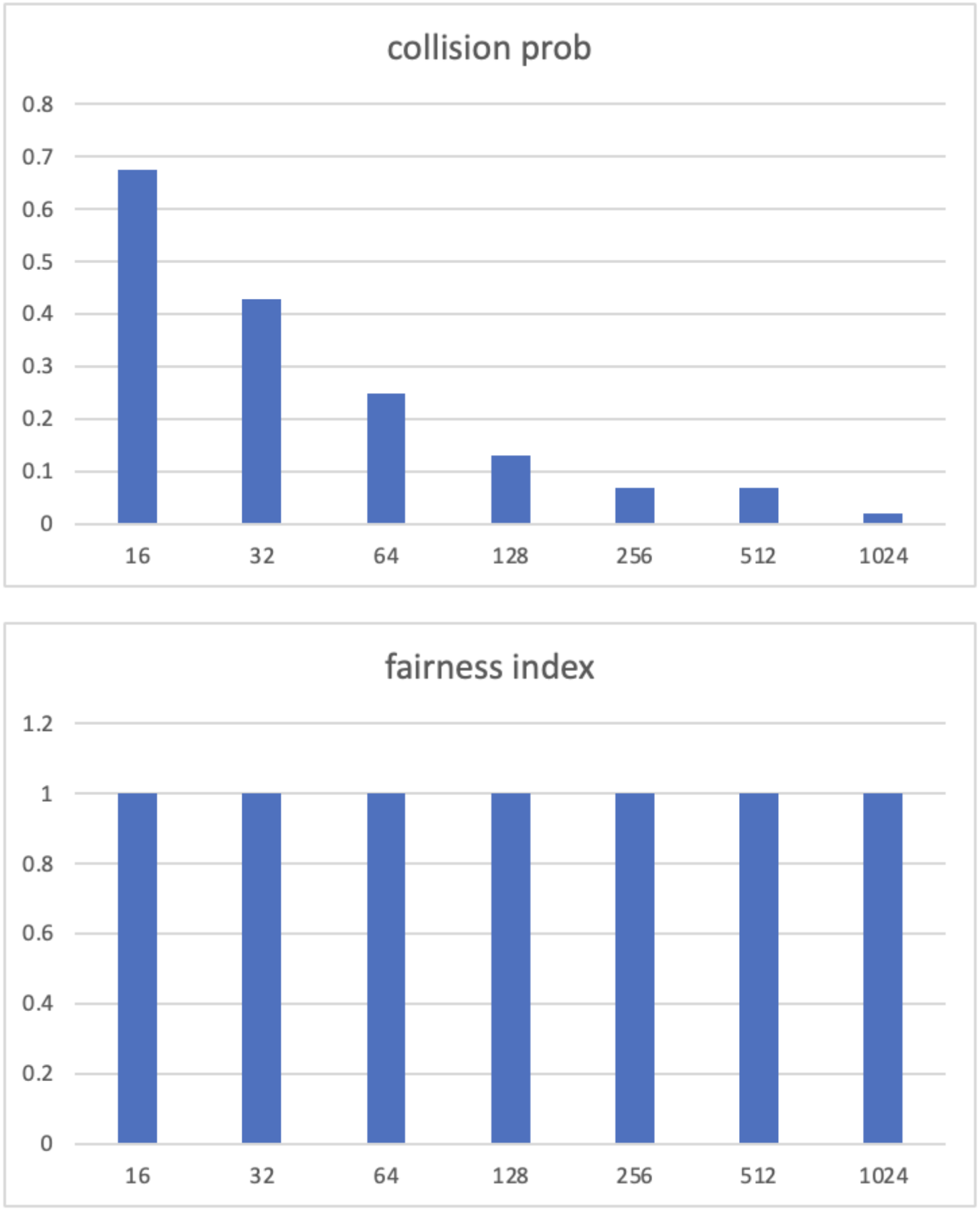
```
collision_prob =

    0.0686


utilization =

    0.6867

>>
```

CWmin = 1024

```
total_th =

    9.3720


fairness_index =

    0.9998


collision_prob =

    0.0194


utilization =

    0.3866

>>
```

전체 네트워크의 utilization, collision prob, fairness index 성능 지표를 관찰





Nuser = 5

CWmin = 16

```
total_th =

   17.4568


fairness_index =

    0.9999


collision_prob =

    0.3929


utilization =

    0.7201
```

```
>>
```

CWmin = 32

```
total_th =

   19.2008


fairness_index =

    0.9999


collision_prob =

    0.2182


utilization =

    0.7920

>>
```

CWmin = 64

```
total_th =

   18.7872


fairness_index =

    0.9999


collision_prob =

    0.1187


utilization =

    0.7750

>>
```

CWmin = 128

```
total_th =

   16.7736


fairness_index =

    1.0000


collision_prob =

    0.0584


utilization =

    0.6919
```

```
>>
```

CWmin = 256

```
total_th =

   13.3648


fairness_index =

    0.9999


collision_prob =

    0.0291


utilization =

    0.5513

>>
```

CWmin = 512

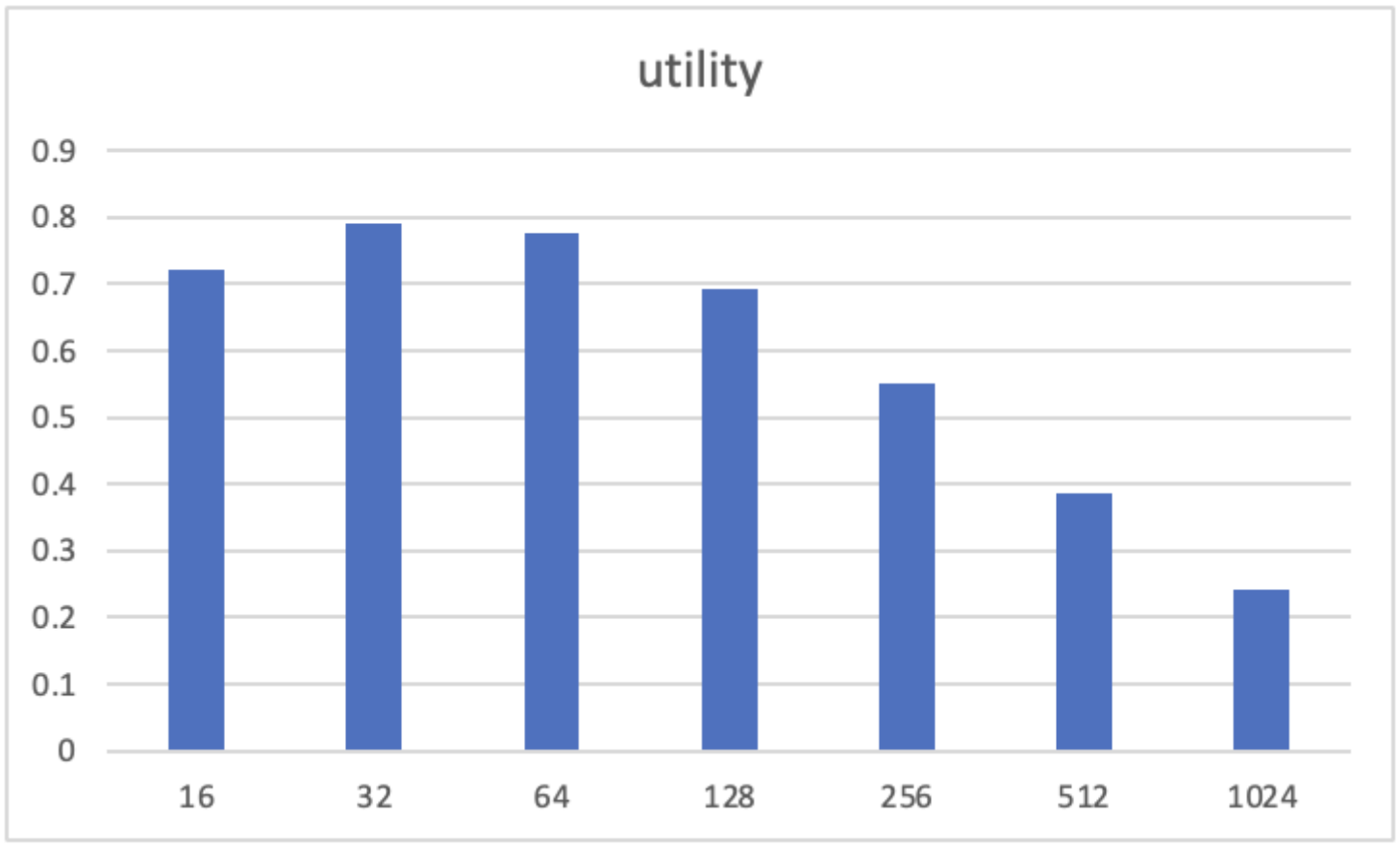
```
total_th =

    9.3360


fairness_index =

    0.9999


collision_prob =

    0.0182


utilization =

    0.3851

>>
```

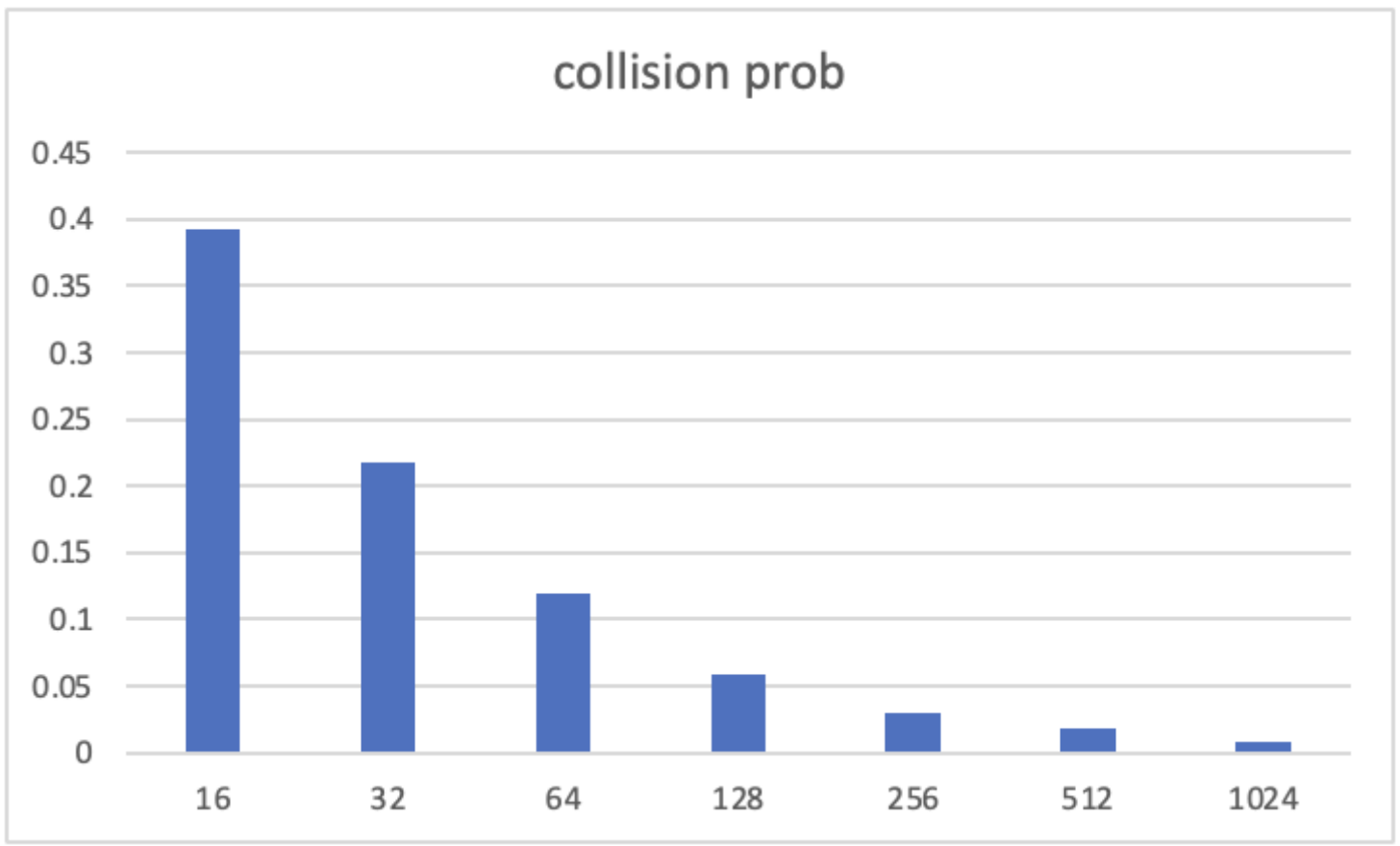
CWmin = 1024

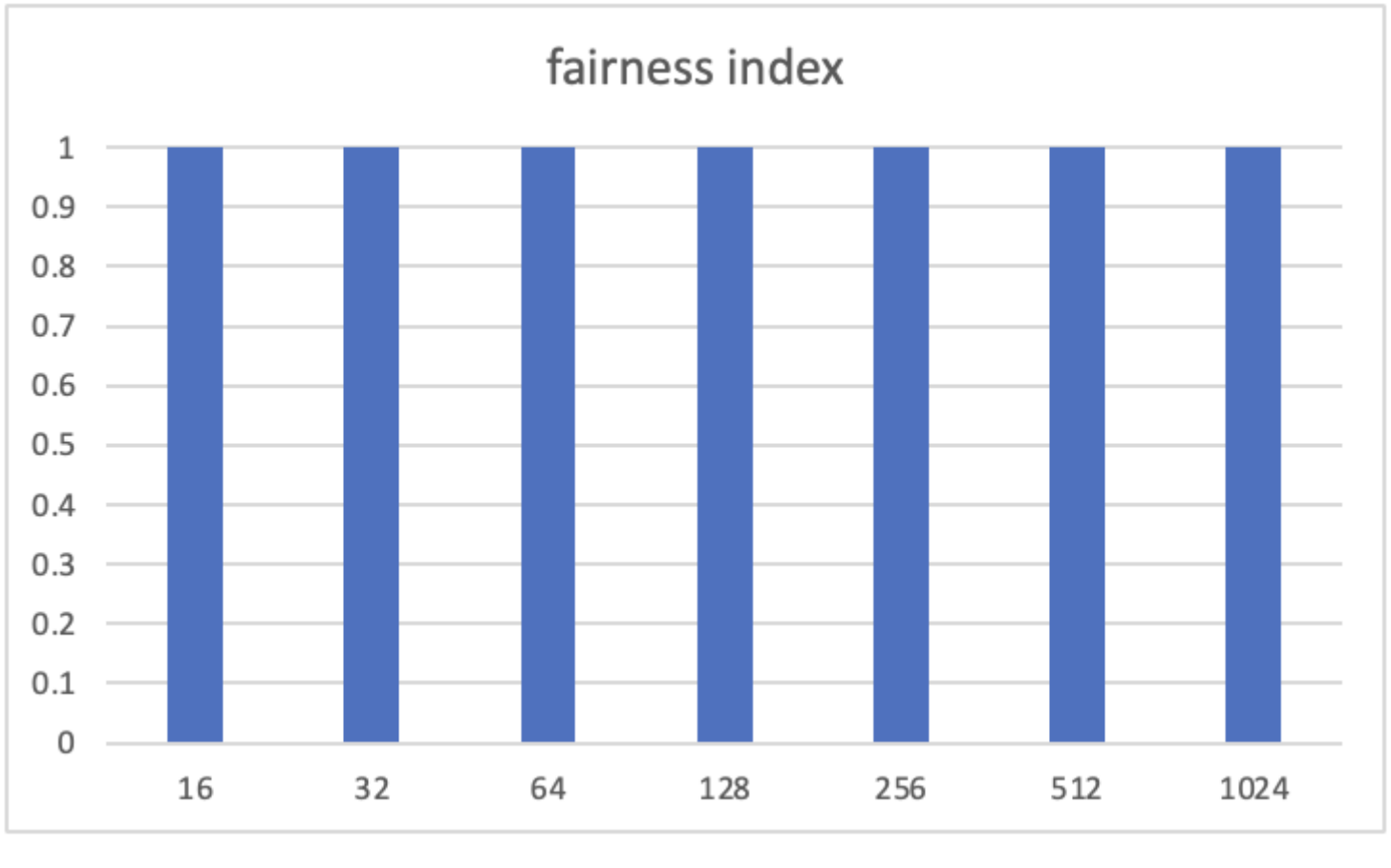
```
total_th =

    5.8384


fairness_index =

    0.9997


collision_prob =

    0.0084


utilization =

    0.2408
```

```
>>
```

전체 네트워크의 utilization, collision prob, fairness index 성능 지표를 관찰







Nuser = 20

CWmin = 16

```
total_th =

   17.4280


fairness_index =

    0.9999


collision_prob =

    0.3946


utilization =

    0.7189

>>
```

CWmin = 32

```
total_th =

   19.1160


fairness_index =

    0.9999


collision_prob =

    0.2237


utilization =

    0.7885

>>
```

CWmin = 64

```
total_th =

   18.7680


fairness_index =

    1.0000


collision_prob =

    0.1206


utilization =

    0.7742

>>
```

CWmin = 128

```
total_th =

   16.7224


fairness_index =

    0.9999


collision_prob =

    0.0612


utilization =

    0.6898

>>
```

CWmin = 256

```
total_th =

   13.3104


fairness_index =

    0.9998


collision_prob =

    0.0311


utilization =

    0.5491

>>
```

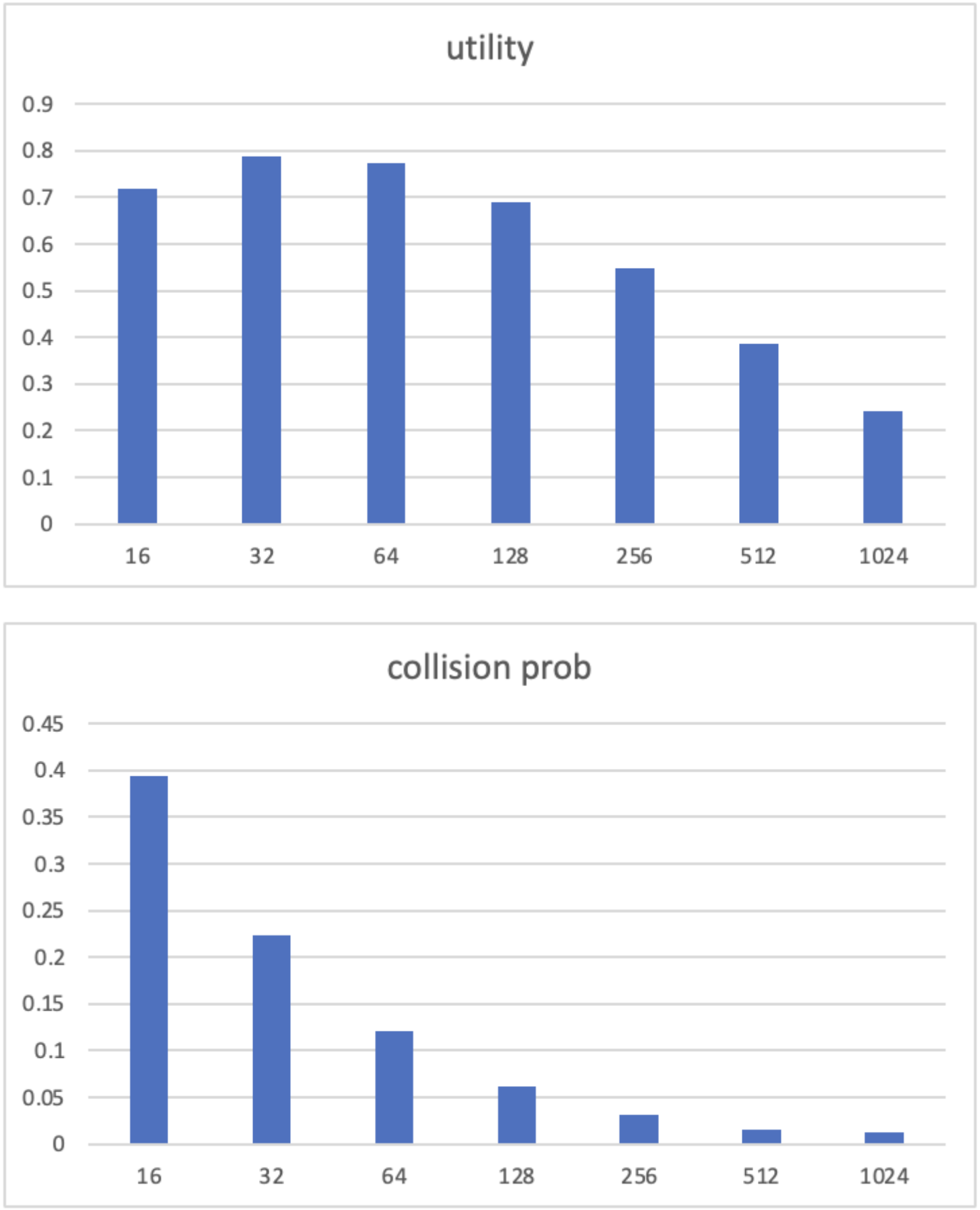
CWmin = 512

```
total_th =

    9.3456


fairness_index =

    0.9999


collision_prob =

    0.0153


utilization =

    0.3855

>>
```

CWmin = 1024

```
total_th =

    5.8400


fairness_index =

    0.9998


collision_prob =

    0.0118


utilization =

    0.2409

>>
```

전체 네트워크의 utilization, collision prob, fairness index 성능 지표를 관찰


utility
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
16
32
64
128
256
512
1024
collision prob
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
16
32
64
128
256
512
1024
fairness index
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
16
32
64
128
256
512
1024

실습 1.2 - 각 단말별 CWmin의 차별화 영향

case1: CWmin = (16, 16, 16, 16)

```
n_access =

        8472        8398        8418        8427


n_success =

        5791        5706        5740        5735


collision_prob =

    0.3186

>>
```

CWmin이 모두 16으로 같으므로, n_access와 n_success는 모두 유사하게 발생했다.

CWmin이 모두 같으므로, collision_prob는 약 32%로 충돌 확률이 높다.

case2: CWmin = (16, 16, 16, 32)

```
n_access =

        9214        9289        9230        4745


n_success =

        6709        6859        6794        3263


collision_prob =

    0.2726

>>
```

CWmin = (16, 16, 16, 32)

CWmin이 16인 단말들은 32인 단말보다 n_access가 약 2배 더 많다.

따라서 충돌도 많이 발생하고, tput이 높다.

case1과 비교하였을 때 한 단말의 CWmin이 32로 커졌으므로, case1보다 collision_prob이 줄어들었다.

case3: CWmin = (16, 16, 32, 64)

```
n_access =

       11064       11109        5652        2923


n_success =

        8920        8924        4313        2127


collision_prob =

    0.2102
```

```
>>
```

CWmin = (16, 16, 32, 64)

n_access는 CWmin과 반비례한다.

n_success는 n_access와 대체로 비례한다.

case2와 비교하였을 때 한 단말의 CWmin이 64로 커졌으므로, case2보다 collision_prob이 줄어들었다.

case4: CWmin = (16, 32, 64, 128)

```
n_access =

       14855        7683        3941        1934


n_success =

       13322        6506        3186        1556


collision_prob =

    0.1353

>>
```

CWmin = (16, 32, 64, 128)

n_access는 CWmin과 반비례한다.

n_success는 n_access와 대체로 비례한다.

CWmin 값이 네 단말 모두 다르므로, collision_prob는 약 14%로 충돌 확률이 낮다.

실습 2.1 - 각 단말별 전송 프레임 크기가 다른 경우 Fairness

case1: packet length = (1000, 1000, 1000, 1000) Bytes

```
n_access =

        8384        8402        8422        8390


n_success =

        5772        5807        5796        5774


per_user_th =

    4.6176    4.6456    4.6368    4.6192


total_th =

   18.5192

>>
```

네 단말의 프레임 크기가 모두 1000bytes로 같이 때문에 n_access와 n_success의 값은 서로 유사하다.

네 단말은 성공 시 1000bytes의 프레임을 전송하고, 전체 tput은 18.5192이다.

case2: packet length = (1000, 1000, 1000, 500) Bytes

```
n_access =

        9281        9288        9301        9364


n_success =

        6425        6418        6397        6492


per_user_th =

    5.1400    5.1344    5.1176    2.5968


total_th =

   17.9888

>>
```

case1과 비교했을 때, 한 단말의 프레임 크기가 500으로 작아졌다.

n_access는 네 단말 모두 비슷하고, case1 보다 값이 커졌다.

n_success는 네 단말 모두 비슷하다.

세 단말은 성공 시 1000bytes를 전송하고, 다른 한 단말은 성공 시 500bytes를 전송한다. 따라서 tput은 약 2배 차이가 난다.

case1과 비교했을 때, 한 단말의 프레임의 크기가 500으로 작아졌다. 따라서 전체 tput은 17.9888으로, case1보다 작다.

case3: packet length = (1000, 1000, 500, 500) Bytes

```
n_access =

       10527       10625       10650       10835


n_success =

        7271        7257        7266        7405


per_user_th =

    5.8168    5.8056    2.9064    2.9620


total_th =

   17.4908

>>
```

case2와 비교했을 때, 한 단말의 프레임 크기가 500으로 작아졌다.

n_access는 네 단말 모두 비슷하고, case2 보다 값이 커졌다.

n_success는 네 단말 모두 비슷하다.

두 단말은 성공 시 1000bytes를 전송하고, 다른 두 단말은 성공 시 500bytes를 전송한다. 따라서 tput은 약 2배 차이가 난다.

case2와 비교했을 때, 한 단말의 프레임의 크기가 500으로 작아졌다. 따라서 전체 tput은 17.4908으로, case2보다 작다.

case4: packet length = (2000, 1000, 500, 100) Bytes

```
n_access =

        8450        8514        8468        8430
```

```
n_success =

        5768        5857        5778        5784


per_user_th =

    9.2288    4.6856    2.3112    0.4627


total_th =

   16.6883

>>
```

네 단말의 n_access는 모두 비슷하다.

n_success의 값도 모두 비슷하다.

각 단말의 tput은 해당 단말이 전송하는 프레임의 크기와 비례한다.

실습 2.2 - 각 단말별 전송 속도가 다른 경우 fairness

case1: Tx_rate = (24, 24, 24, 24) Mb

```
n_access =

        8412        8414        8464        8371


n_collision =

        2676        2616        2659        2637


n_success =

        5736        5798        5805        5734


per_user_th =

    4.5888    4.6384    4.6440    4.5872


total_th =

   18.4584


fairness_index =

    1.0000


collision_prob =

    0.3145


utilization =

    0.7614

>>
```

case2: Tx_rate = (24, 24, 24, 12) Mb/s

```
n_access =
```

```
        6567        6569        6516        6629


n_collision =

        2028        2027        2063        2123


n_success =

        4539        4542        4453        4506


per_user_th =

    3.6312    3.6336    3.5624    3.6048


total_th =

   14.4320


fairness_index =

    0.9999


collision_prob =

    0.3136


utilization =

    0.7440

>>
```

case1과 비교했을 때, 한 단말의 전송 속도가 12mbps로 감소했다.

case1에서 per_user_th = [4.5888 4.6384 4.6440 4.5872] 였지만,

case2에서 per_user_th = [3.6312 3.6336 3.5624 3.6048] 이다.

전체적으로 tput이 감소하였다.

프레임의 크기는 1000bytes이고, 한 단말의 전송 속도가 절반인 12mbps로 줄었으므로 이 단말의 전송 시간은 2배로 증가하였다. 따라서 n_access가 감소했고, 전체적으로 tput도 감소하였다.

case3: Tx_rate = (24, 24, 12, 12) Mb/s

```
n_access =

        5504        5524        5530        5501


n_collision =

        1742        1746        1691        1713


n_success =

        3762        3778        3839        3788


per_user_th =

    3.0096    3.0224    3.0712    3.0304


total_th =

   12.1336


fairness_index =

    0.9999
```

```
collision_prob =

    0.3124


utilization =

    0.7522

>>
```

case2와 비교했을 때, 한 단말의 전송 속도가 12mbps로 감소했다.

case2에서 per_user_th = [3.6312 3.6336 3.5624 3.6048] 였지만,

case3에서 per_user_th = [3.0096 3.0224 3.0712 3.0304] 이다.

전체적으로 tput이 감소하였다.

프레임의 크기는 1000bytes이고, 한 단말의 전송 속도가 절반인 12mbps로 줄었으므로 이 단말의 전송 시간은 2배로 증가하였다. 따라서 n_access가 감소했고, 전체적으로 tput도 감소하였다.

case4: Tx_rate = (48, 24, 12, 6) Mb/s

```
n_access =

        4284        4255        4202        4227


n_collision =

        1350        1354        1319        1330


n_success =

        2934        2901        2883        2897


per_user_th =

    2.3472    2.3208    2.3064    2.3176


total_th =

    9.2920


fairness_index =

    1.0000


collision_prob =

    0.3155


utilization =

    0.7183

>>
```

case3와 비교했을 때, 한 단말의 전송 속도는 6mbps로 절반 감소했고, 다른 단말의 전송 속도는 48mbps로 두 배 증가했다.

case3에서 per_user_th = [3.0096 3.0224 3.0712 3.0304] 였지만,

case4에서 per_user_th = [2.3472 2.3208 2.3064 2.3176] 이다.

전체적으로 tput이 감소하였다.

실습 3.1 - BEB의 영향 이해

Nuser = 20

case1: BEB 해제

```
total_th =

    5.5312


fairness_index =

    0.9970


collision_prob =

    0.9083


utilization =

    0.2282

>>
```

case2: BEB 설정

```
total_th =

   16.2360


fairness_index =

    0.9911


collision_prob =

    0.4767


utilization =

    0.6697

>>
```

BEB를 설정 후 BEB 해제 시 대비 utilization 증가, collision prob 감소, fairness index 감소, total tput 증가의 변화가 있었다.

전송 실패 시 CWmin의 크기를 두 배 증가시키고, 전송 성공 시 CWmin의 크기를 초기값 16으로 되돌리는 BEB의 메커니즘 때문이다.

Nuser = 5

case1: BEB 해제

```
per_user_th =

    3.4840    3.4760    3.4880    3.4712    3.5232


total_th =

   17.4424
```

```
fairness_index =

    1.0000


collision_prob =

    0.3940


utilization =

    0.7195

>>
```

case2: BEB 설정

```
per_user_th =

    3.7520    3.7784    3.8776    3.7104    3.7104


total_th =

   18.8288


fairness_index =

    0.9997


collision_prob =

    0.2711


utilization =

    0.7767

>>
```

BER을 설정 후 BER 해제 시 대비 utilization 증가, collision prob 감소, fairness index 감소, total tput 증가의 변화가 있었다.

전송 실패 시 CWmin의 크기를 두 배 증가시키고, 전송 성공 시 CWmin의 크기를 초기값 16으로 되돌리는 BER의 메커니즘 때문이다.

실습 3.2 - BEB의 영향 이해

case1: BEB 해제하고 CWmin = 16

Nuser = 5

```
total_th =

   17.5224


fairness_index =

    0.9999


collision_prob =

    0.3889
```

```
utilization =

    0.7228

>>
```

Nuser = 10

```
total_th =

   12.4504


fairness_index =

    0.9995


collision_prob =

    0.6741


utilization =

    0.5136

>>
```

Nuser = 20

```
total_th =

    5.5160


fairness_index =

    0.9966


collision_prob =

    0.9084


utilization =

    0.2275

>>
```

Nuser = 30

```
total_th =

    2.2824


fairness_index =

    0.9887


collision_prob =

    0.9731
```

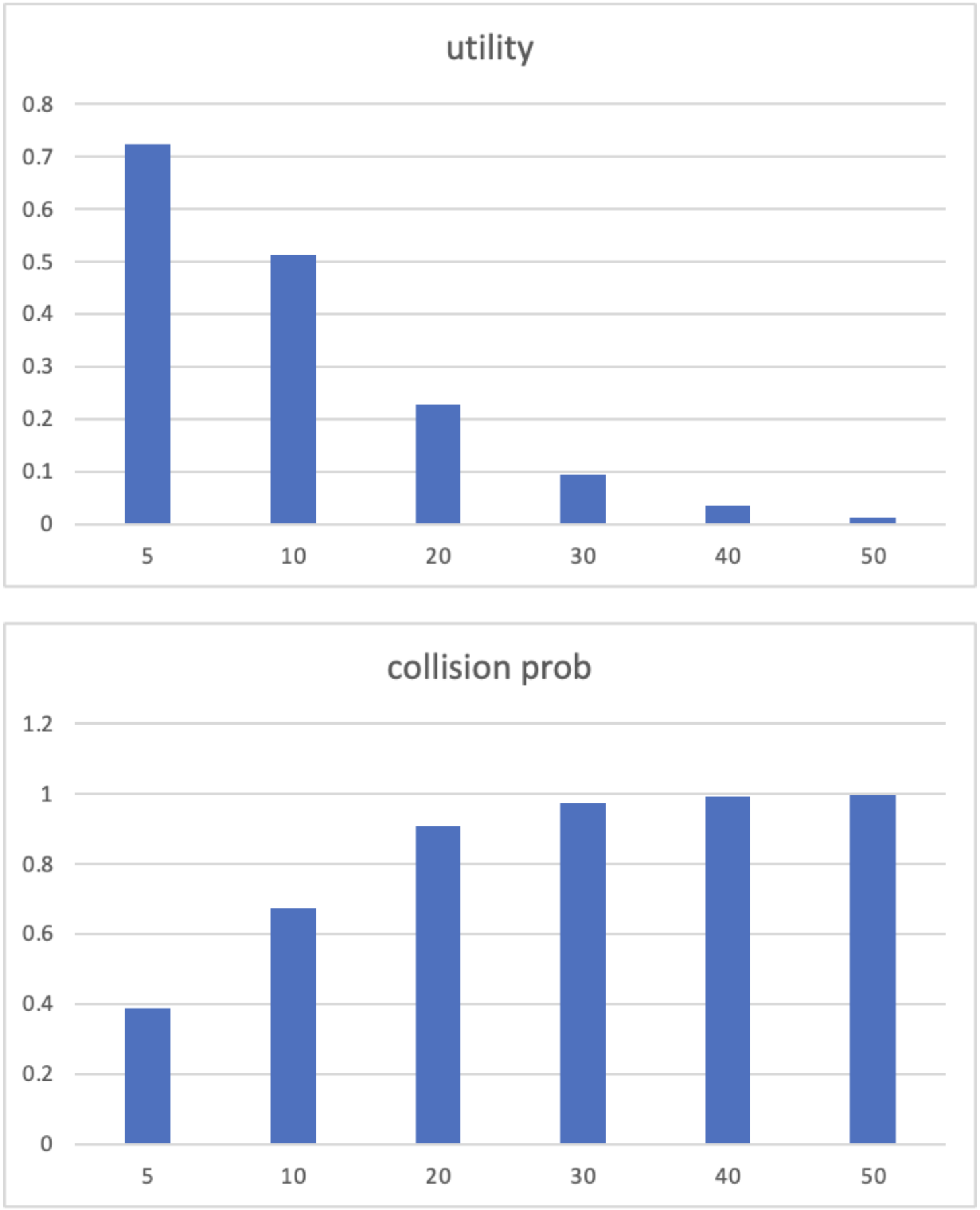
```
utilization =

    0.0941

>>
```

Nuser = 40

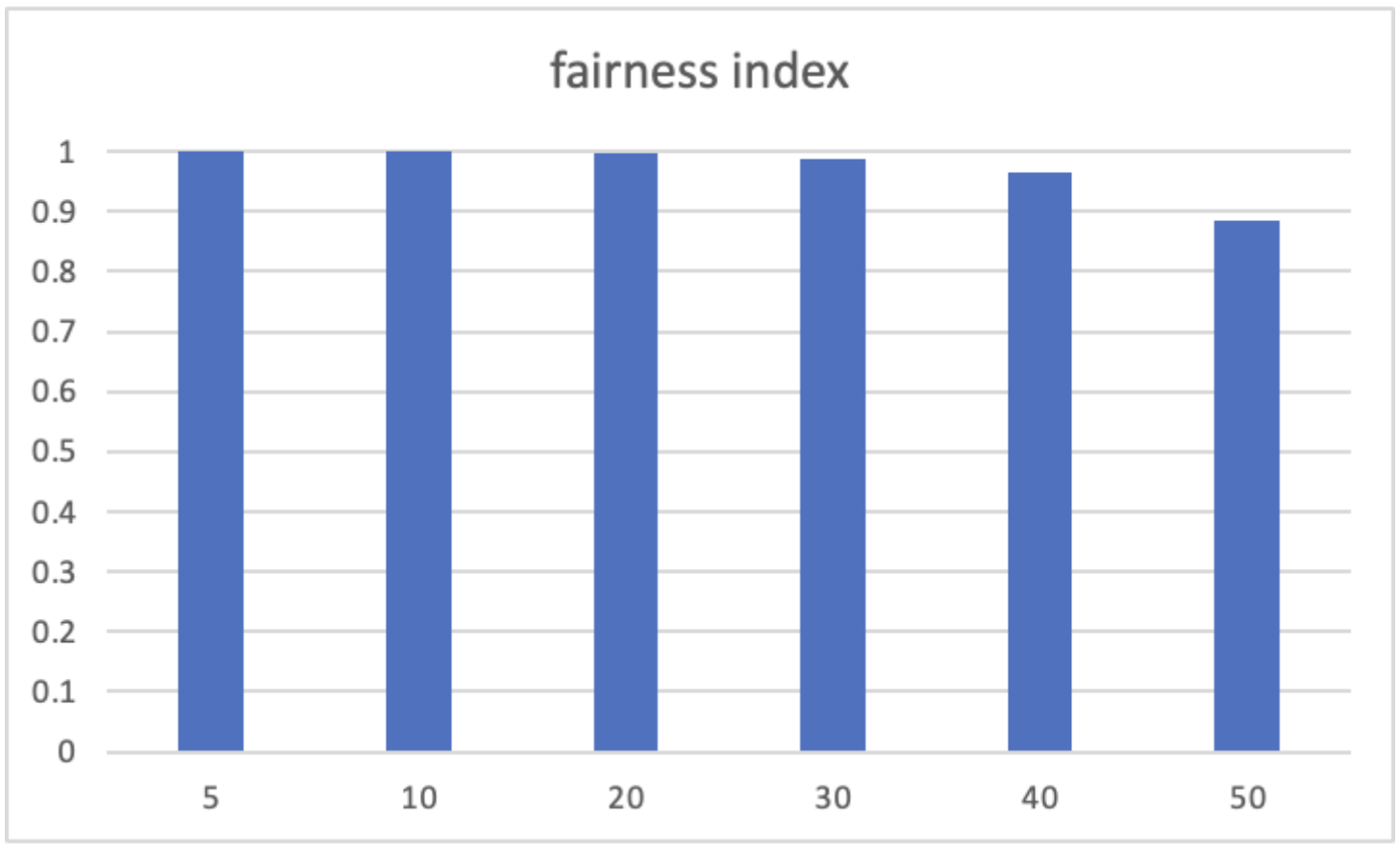
```
total_th =

    0.8272


fairness_index =

    0.9645


collision_prob =

    0.9926


utilization =

    0.0341

>>
```

Nuser = 50

```
total_th =

    0.2888


fairness_index =

    0.8838


collision_prob =

    0.9979


utilization =

    0.0119

>>
```


utility
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
20
30
40
50

collision prob
1.2
1
0.8
0.6
0.4
0.2
0
5
10
20
30
40
50


fairness index
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
20
30
40
50

case2: BEB 해제하고 CWmin = 64

Nuser = 5

```
total_th =

   18.8232


fairness_index =

    0.9999


collision_prob =

    0.1158


utilization =

    0.7765

>>
```

Nuser = 10

```
total_th =

   18.8760


fairness_index =

    0.9999


collision_prob =

    0.2429


utilization =

    0.7786

>>
```

Nuser = 20

```
total_th =

   16.6784


fairness_index =

    0.9996


collision_prob =

    0.4455


utilization =

    0.6880
```

```
>>
```

Nuser = 30

```
total_th =

   14.1336


fairness_index =

    0.9989


collision_prob =

    0.5969


utilization =

    0.5830

>>
```

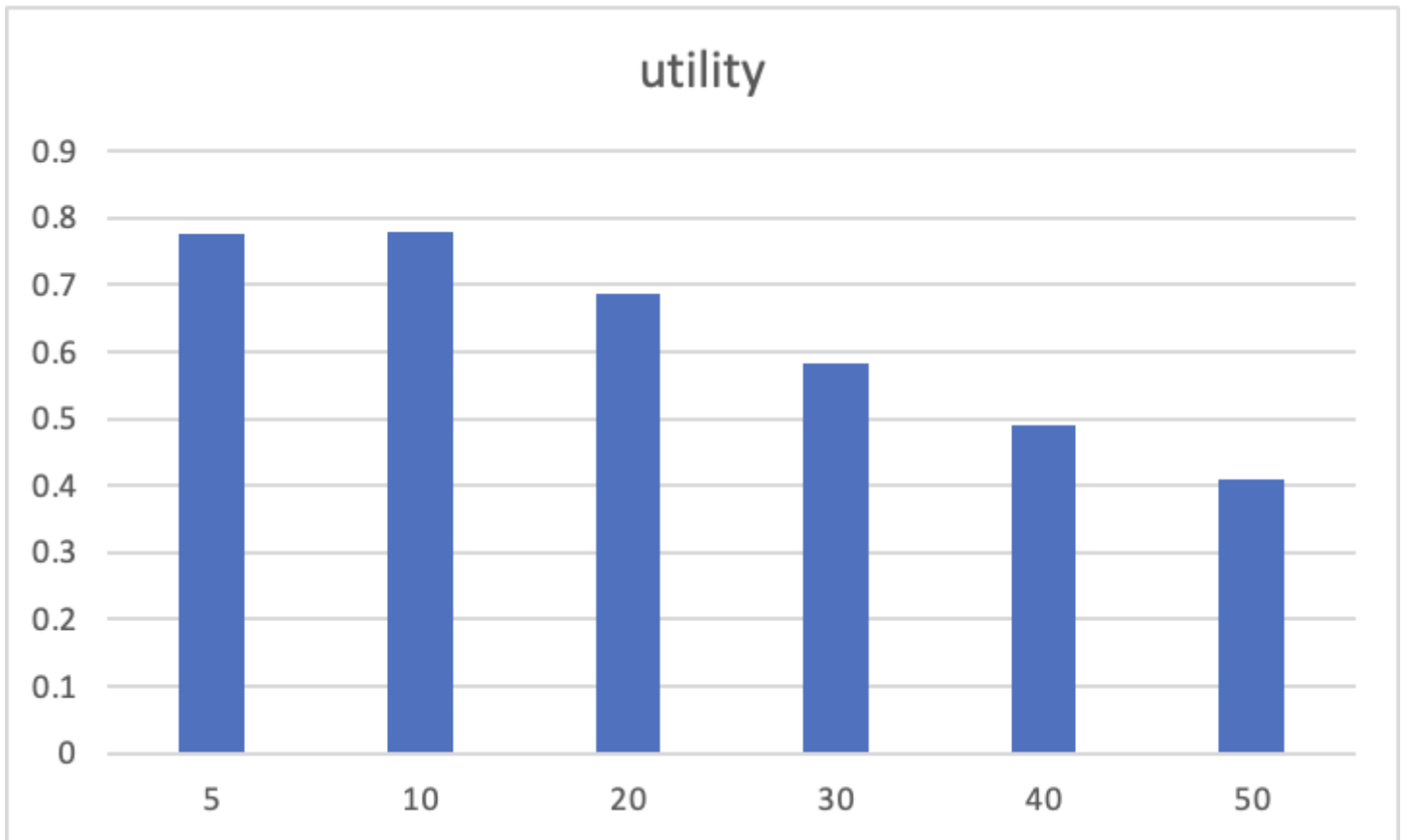
Nuser = 40

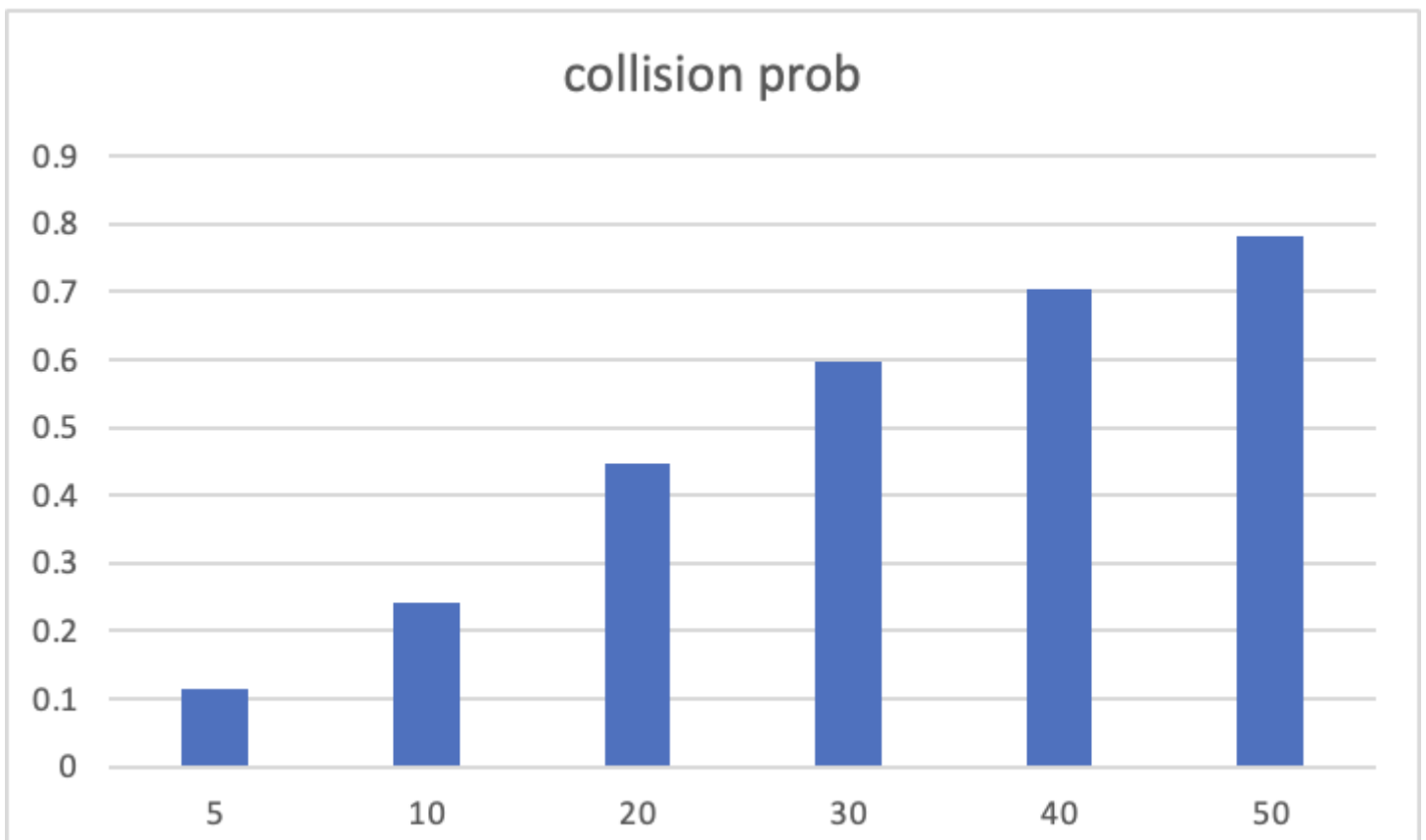
```
total_th =

   11.9152


fairness_index =

    0.9979


collision_prob =

    0.7028


utilization =

    0.4915

>>
```

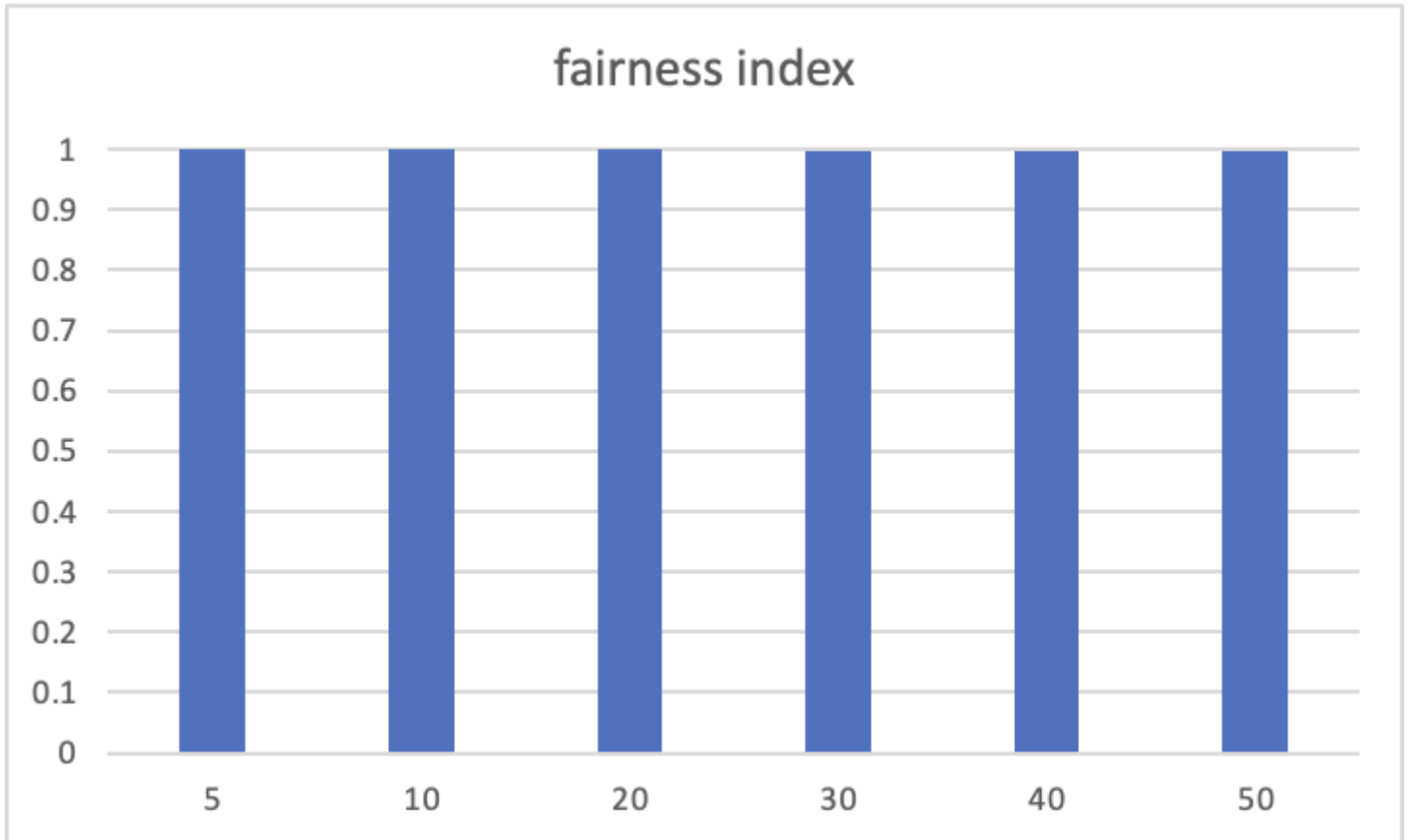
Nuser = 50

```
total_th =

    9.9024


fairness_index =

    0.9965


collision_prob =

    0.7819


utilization =

    0.4085
```

```
>>
```


utility
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
20
30
40
50


collision prob
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
20
30
40
50



case3: BEB 설정하고 CWmin = 16

Nuser = 5

```
total_th =

   18.8336


fairness_index =

    0.9994


collision_prob =

    0.2720


utilization =

    0.7769

>>
```

Nuser = 10

```
total_th =

   17.5992


fairness_index =

    0.9921


collision_prob =

    0.3792


utilization =

    0.7260
```

```
>>
```

Nuser = 20

```
total_th =

   16.2536


fairness_index =

    0.9799


collision_prob =

    0.4732


utilization =

    0.6705

>>
```

Nuser = 30

```
total_th =

   15.3560


fairness_index =

    0.9763


collision_prob =

    0.5299


utilization =

    0.6334

>>
```

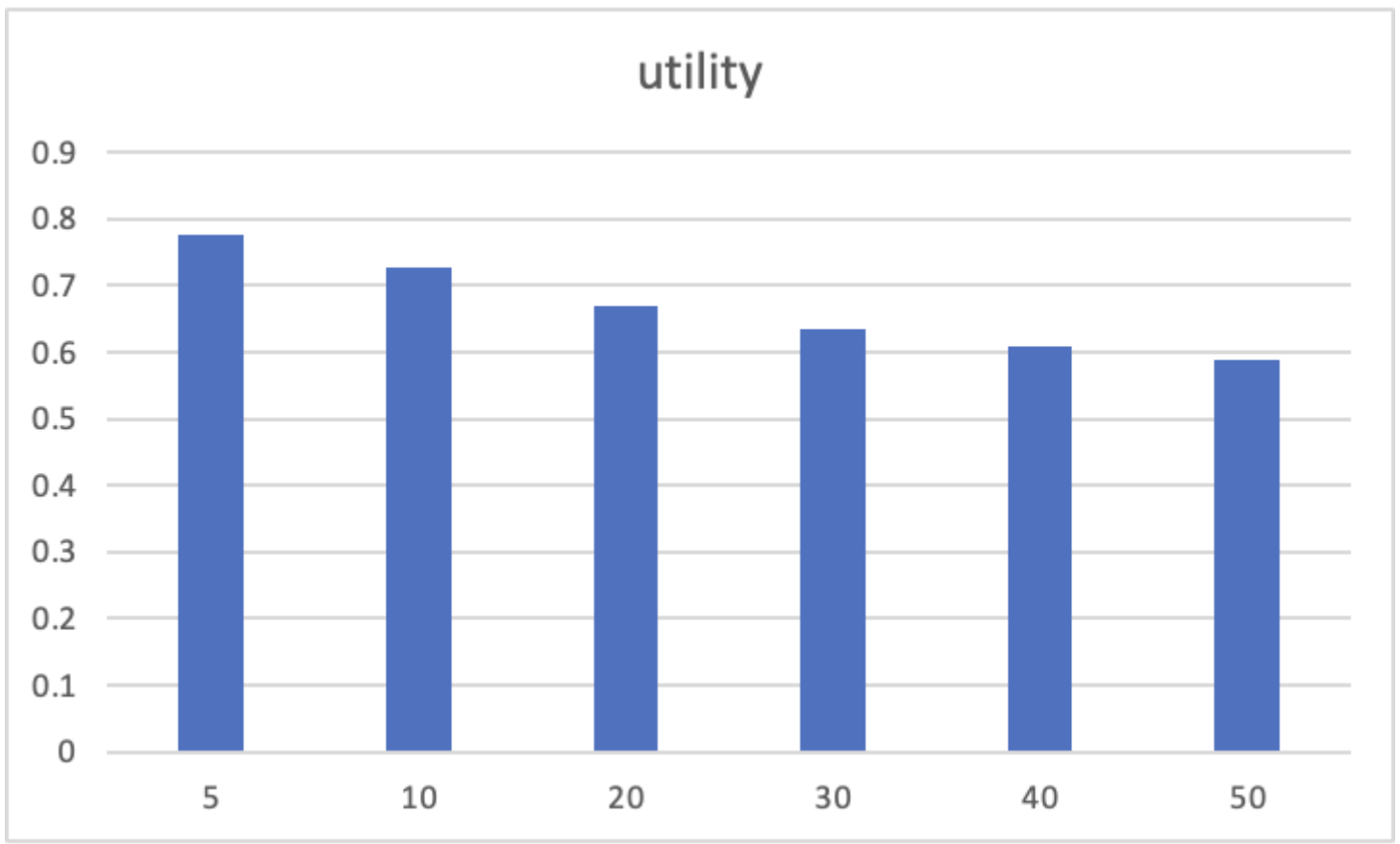
Nuser = 40

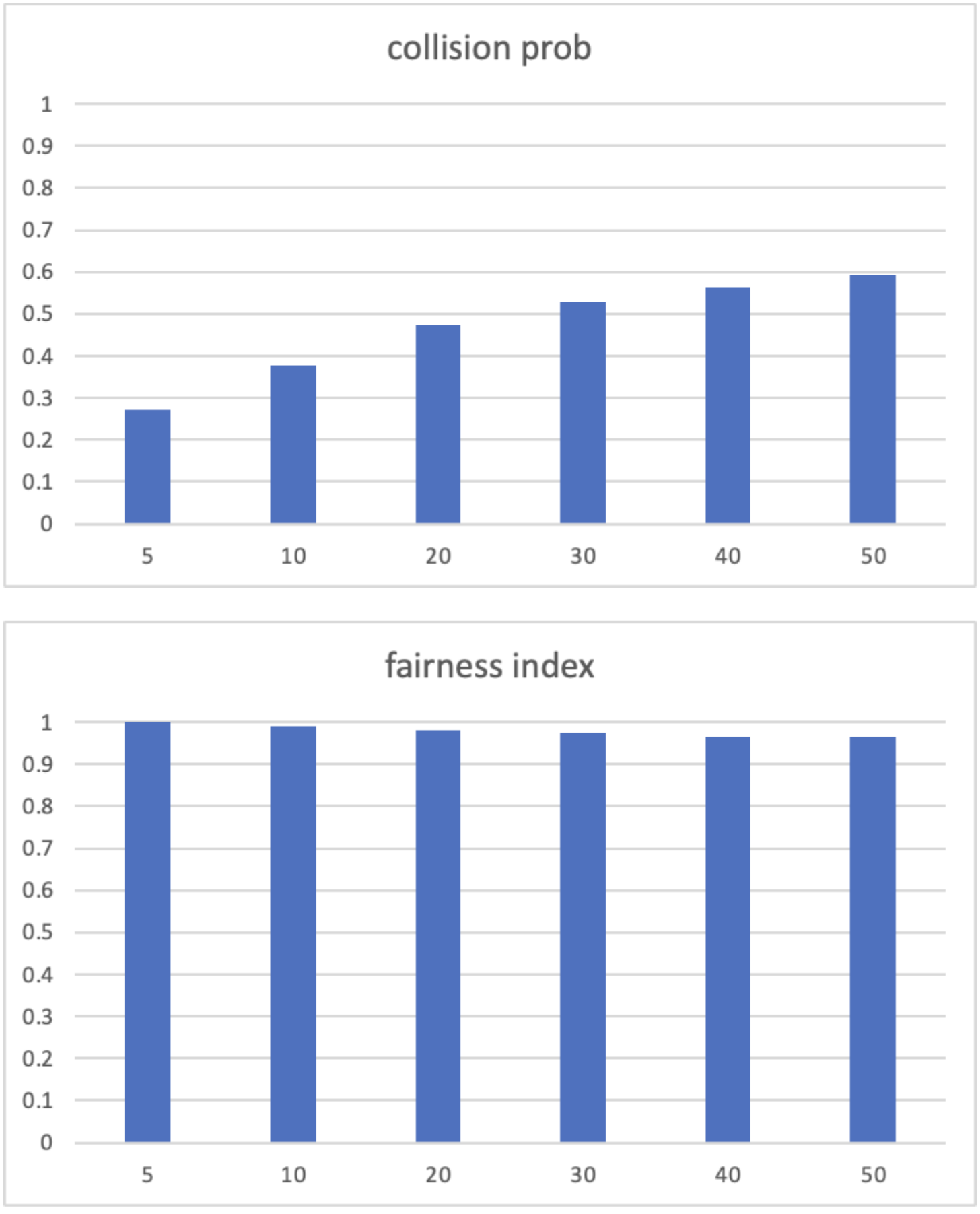
```
total_th =

   14.7360


fairness_index =

    0.9658


collision_prob =

    0.5640


utilization =

    0.6079
```

```
>>
```

Nuser = 50

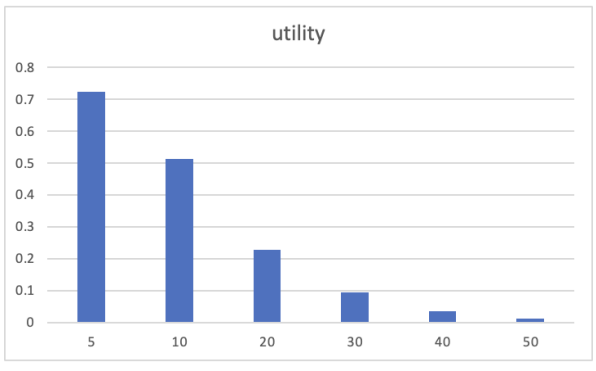
```
total_th =

   14.2368


fairness_index =

    0.9658


collision_prob =

    0.5916


utilization =

    0.5873

>>
```

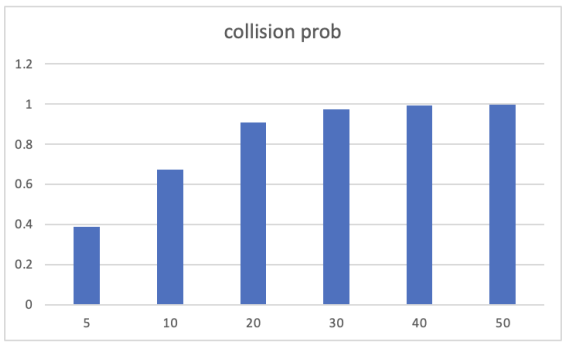


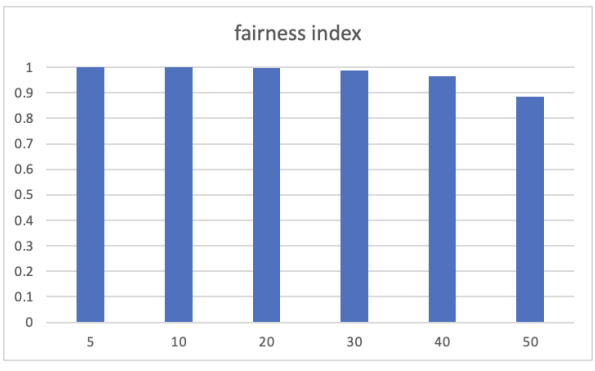


BEB 설정하지 않고 고정된 CWmin 값을 사용하는 경우에 비해 BEB 메커니즘의 장점을 설명

case1: BER 해제, CWmin = 16

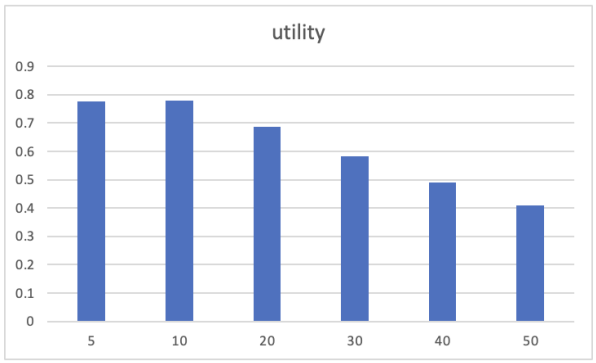


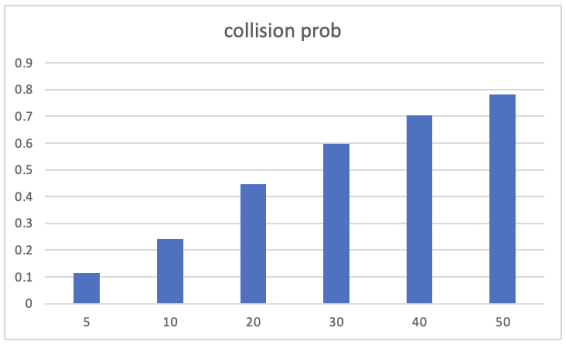


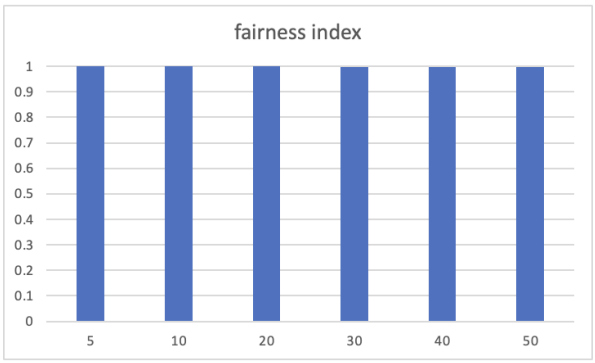


CWmin이 16으로 고정되어있고, BER이 해제되어있으므로 collision prob이 대체적으로 높게 발생했고, utility는 낮다.

case2: BER 해제, CWmin = 64

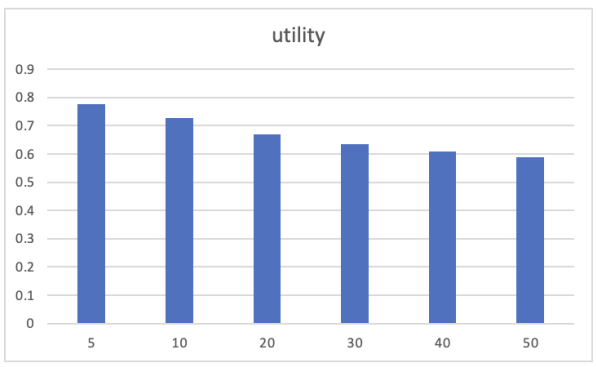


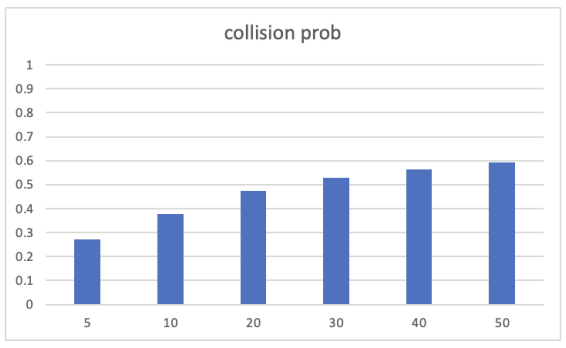


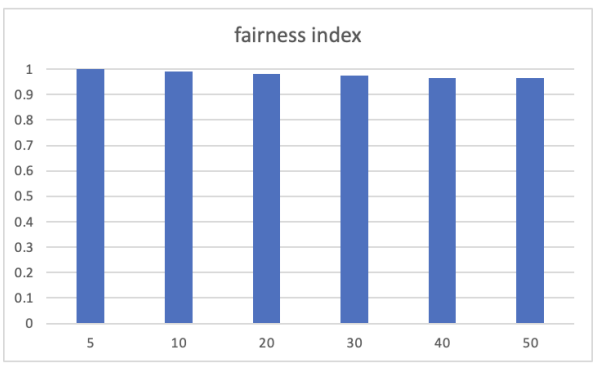


CWmin을 case1 대비 4배 증가시켜 64로 고정되어있다. collision prob이 case1 대비 일정 부분 감소하였고, utility도 대체적으로 높아졌다. 하지만, 여전히 user가 많을 때 utility가 높지 않다.

case3: BEB 설정, CWmin = 16







CWmin이 가변적으로 변하는 BEB를 설정하였다.

collision prob이 user가 많은 상황에서 크게 감소하였고, utility도 증가하였다.

user가 많아지면 충돌 확률이 증가하고, 이에 optimal CW size가 큰 것이 요구된다.

BEB는 충돌 발생 시 CWmin의 크기를 2배 증가시킴으로써 CW size를 가변적으로 조정하여 user의 수에 대응한다.