2018320022 이유섭

Programming Project #1, Spring 2019

1. 새로운 네트워크 생성

프로그래밍에 앞서 우리는 과제에서 제시한 아래 그림과 같이 실험을 위한 간단한 네트워크 토폴로지를 생성해야 한다.

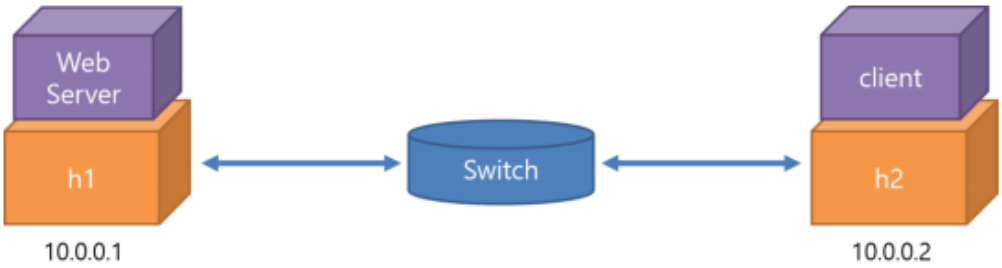


Figure 1. A simple network topology for experiments

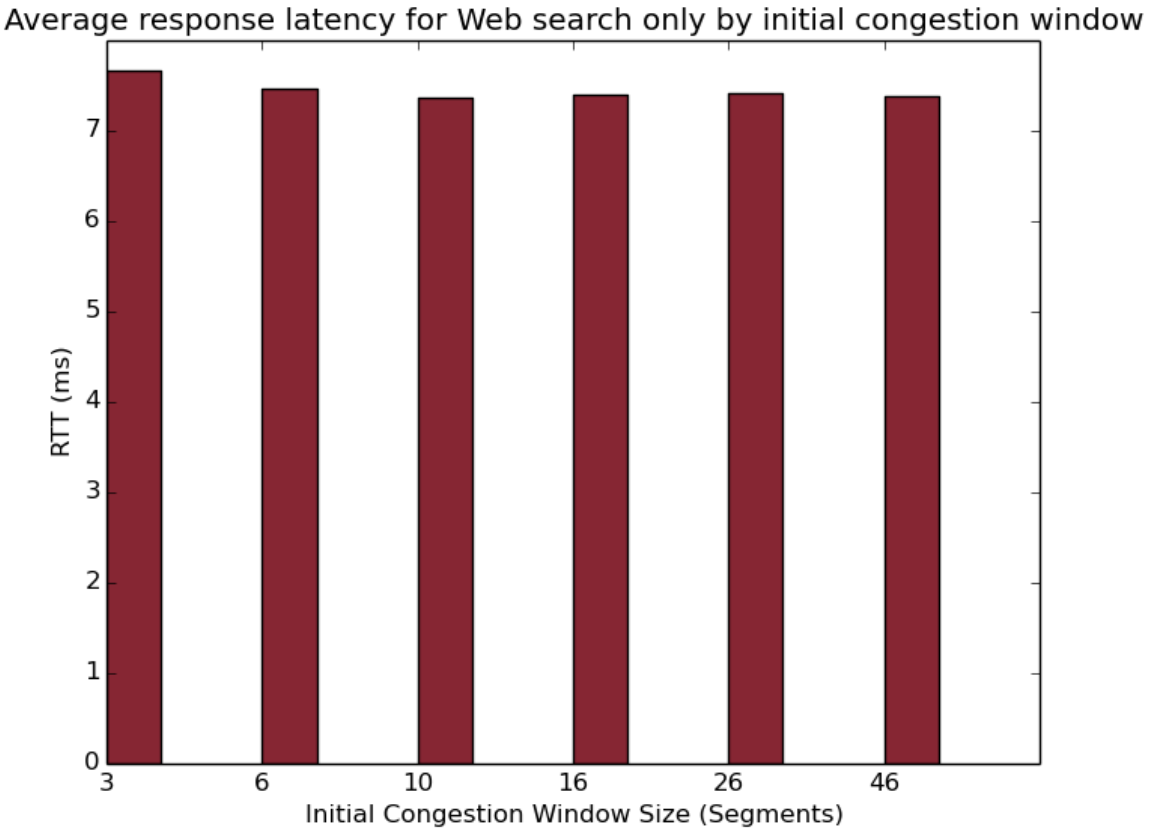
그림에서 보면 webserver node인 h1과 client h2가 있으며 그 사이가 Switch로 연결되어 있음을 알 수 있다. 이를 바탕으로 아래와 같이 host로 h1과 h2를 만들고, s1이라는 스위치를 생성하여 h1과 h2를 각각 s1에 연결해 두 host h1과 h2를 연결해 줄 수 있다.

```
switch = self.addSwitch('s1')

h1 = self.addHost('h1')
h2 = self.addHost('h2')

self.addLink(h1, switch, bw=bandwidth, delay='%dms' %(delay))
self.addLink(h2, switch, bw=bandwidth, delay='%dms' %(delay))
```

이때, bw와 delay를 setup 시 할당 받은 bandwidth와 delay로 설정해야 한다.

2. init_cwnd_experiment() 함수 결과

init_cwnd_experiment() 는 이미 완벽히 작성된 함수이므로 아무런 수정 없이 실행을 시켜도 된다. 아래 그림은 init_cwnd_experiment() 함수를 실행시켰을 때 얻을 수 있는 이미지 파일이다.



init_cwnd_experiment() 함수는 같은 bandwidth, 같은 delay, 같은 segment 들을 바탕으로 initial congestion window 만 바꿔가며 측정한 결과이다.

Initial congestion window 값이 바뀌면서 RTT 값이 바뀌는 것을 볼 수 있으나 그 차이가 크지 않음을 알 수 있다. 때문에 다음에 진행되는 실험들에 대해서는 이 값들에 1000을 곱하여 가시적으로 보여줄 것이다.

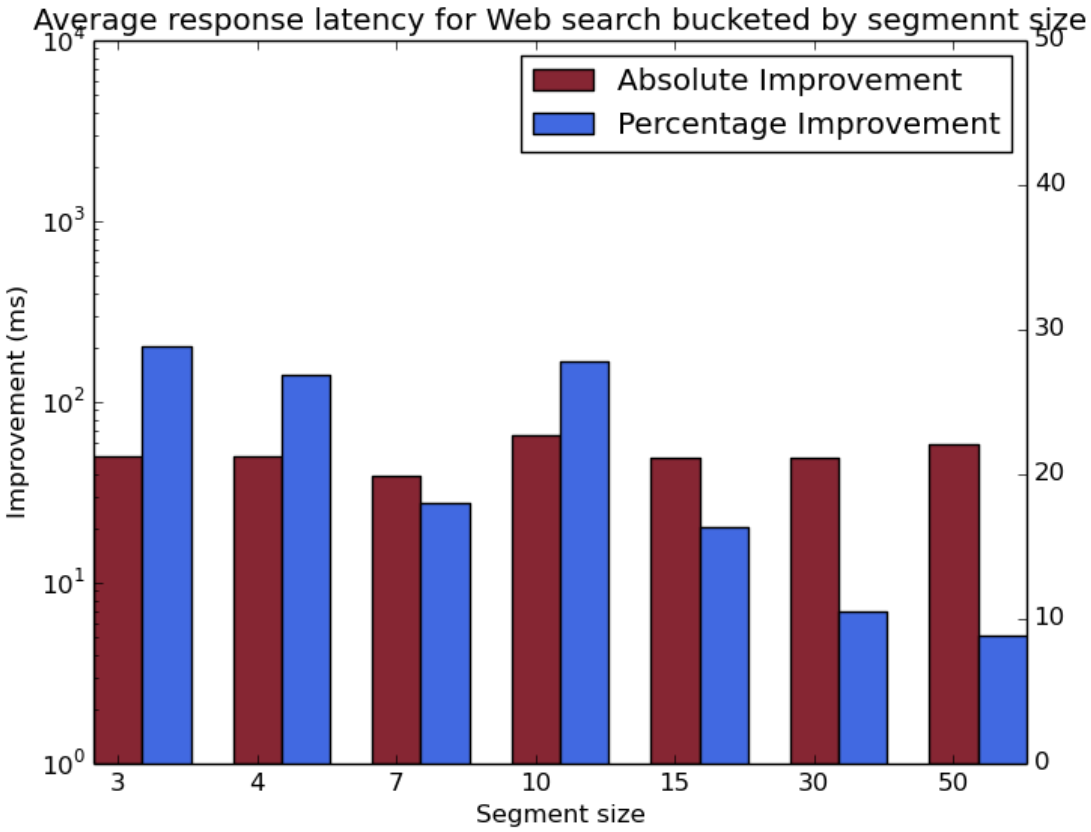
initial congestion window 크기를 키우면서 측정한 RTT 값이 초기 3번의 시행에서는 줄어드는 것이 보이나 그 이후에는 그 차이가 크지 않음을 알 수 있다. 때문에 우리는 이후 진행할 다양한 segment의 크기, bandwidth, network delay의 값의 상황에서 initial congestion window 크기에 따라 RTT 값이 얼마나 달라지는지 확인하는 실험에서 비교할 두 initial congestion window 크기를 3과 10으로 설정하였다.

3. segment_size_experiment() 함수 작성

이 함수는 segment 크기를 달리하였을 때, initial congestion window에 따라 RTT 값이 얼마나 달라지는지 실험해보는 함수로써 다음과 같이 두 cwnd에 대해서 segment의 크기를 달리하여 실험하는 코드이다. (segment 크기가 점점 커지도록 설정하였다.)

```
for r in range(N):
    for i in range(M):
        initcwnd, initrwnd = MODE[i]
        results[r, i] = experiment(bandwidth,
                                   delay,
                                   initcwnd,
                                   initrwnd,
                                   file=["%s.html" %(SEG[r])])
        print results
```

아래의 표는 segment_size_experiment() 함수를 실행시켜 얻은 결과이다.



위의 표를 보면 cwnd 값이 커짐에 따라 RTT 값이 줄어든 것을 볼 수 있으나 segment의 크기에 RTT의 절대적인 차는 큰 차이가 없음을 알 수 있다. 그러나 상대적으로 segment의 크기가 커짐에 따라 RTT에 따른 RTT차의 비율이 줄어드는 것으로 보아 segment 의 값이 커질수록 전체적인 RTT 값이 증가하고 있음을 예측할 수 있다.

4. bandwidth_experiment() 함수 작성

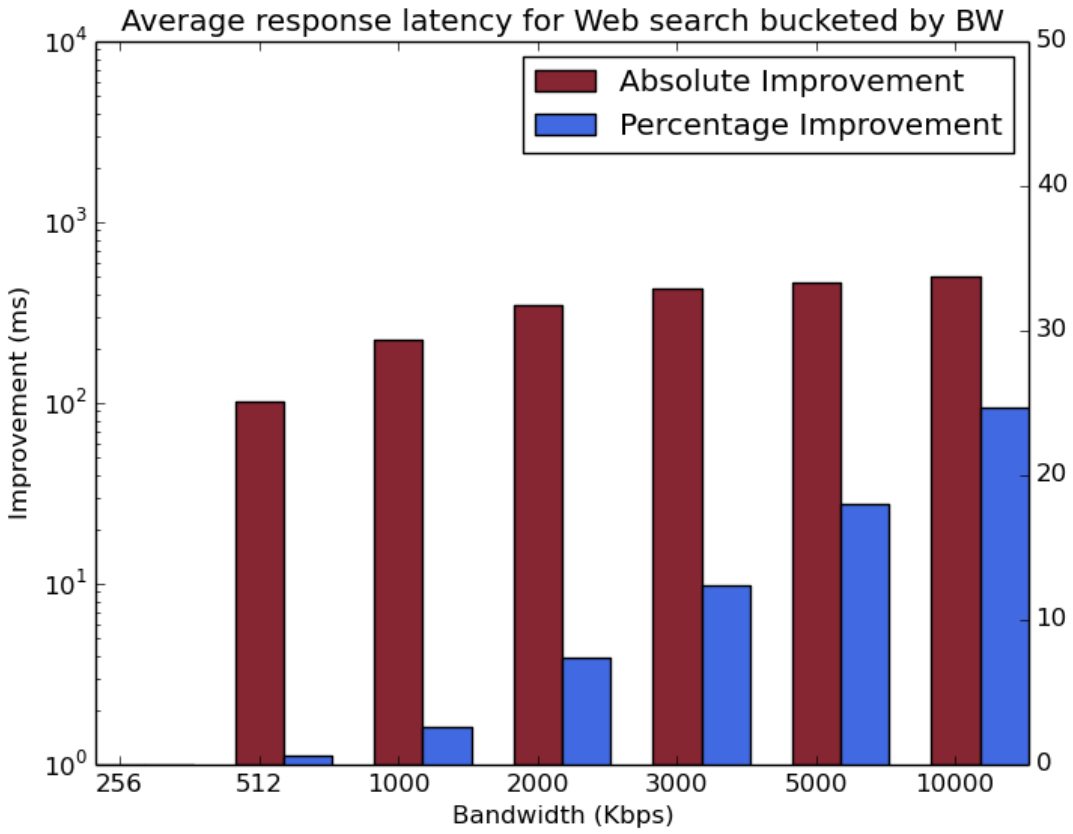
이 함수는 bandwidth를 달리하였을 때, initial congestion window에 따라 RTT 값이 얼마나 달라지

는지 실험해보는 함수로써 다음과 같이 두 cwnd에 대해서 bandwidth를 달리하여 실험하는 코드이다. (bandwidth의 값이 점점 커지도록 설정하였다.)

```
for r in range(N):
    for i in range(M):
        initcwnd, initrwnd = MODE[i]
        results[r, i] = experiment(BW[r]/1000.0,
                                   delay,
                                   initcwnd,
                                   initrwnd)
        print results
```

여기서 bandwidth 자리에 단순히 BW[r] 이라고 쓰지 않고 1000.0으로 나눈 이유는 기존에 bandwidth의 값으로 설정할 때 가정한 단위와 BW tuple에 값을 넣었을 때 가정한 단위가 다르기 때문이다.

아래의 표는 segment_size_experiment() 함수를 실행시켜 얻은 결과이다.



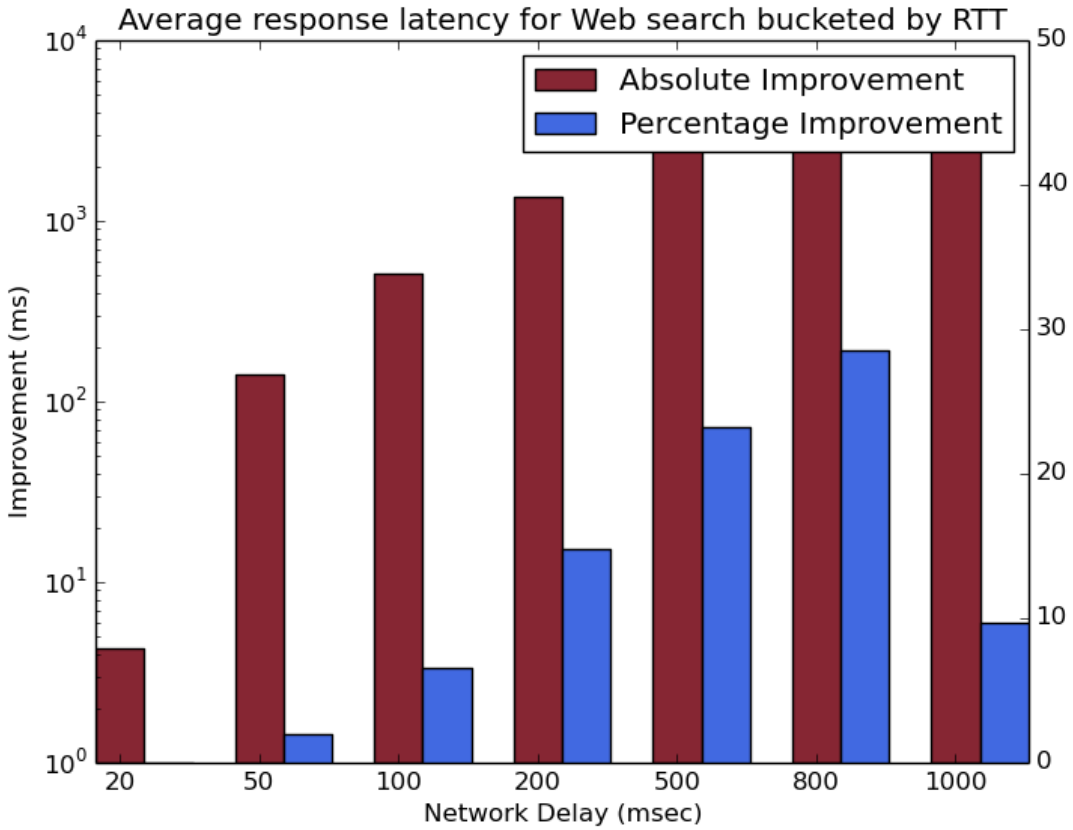
위의 표를 보면 bandwidth가 커질수록 cwnd에 의한 RTT 차가 커지는 것을 볼 수 있다. 또한 위의 표에서 RTT의 차가 선형으로 증가하는 반면 RTT 차의 비가 지수적으로 증가하는 것으로 보아 전체적인 RTT 값은 줄어들고 있음을 알 수 있다.

5. network_delay_experiment() 함수 작성

이 함수는 network delay를 달리하였을 때, initial congestion window에 따라 RTT 값이 얼마나 달라지는지 실험해보는 함수로써 다음과 같이 두 cwnd에 대해서 network delay를 달리하여 실험하는 코드이다. (network delay의 값이 점점 커지도록 설정하였다.)

```
for r in range(N):
    for i in range(M):
        initcwnd, initrwnd = MODE[i]
        results[r, i] = experiment(bandwidth,
                                   delay[r],
                                   initcwnd,
                                   initrwnd)
        print results
```

아래의 표는 segment_size_experiment() 함수를 실행시켜 얻은 결과이다.



위의 표를 보면 bandwidth가 커질수록 cwnd에 의한 RTT 차가 크게 증가하는 것을 볼 수 있다. 또한 위의 표에서 RTT의 차와 같이 RTT 차의 비가 함께 선형으로 증가하는 것으로 보아 전체적인 RTT 차이는 크지 않음을 알 수 있다. RTT의 차가 계속 선형으로 증가하는 반면에 RTT 차의 비

가 마지막에 급격히 떨어지는 모습을 볼 수 있는데, 이는 network delay의 증가에 따라 RTT가 크게 변하지는 않지만 어느 순간을 기점으로 RTT값이 크게 증가하지 구간이 있을 수 있다는 것을 예상해볼 수 있다.

6. generate_final_figure() 함수 작성

이 함수는 앞서 설명한, 3, 4, 5 함수를 실행했을 때 나온 결과를 한 눈에 살펴보고 이들을 비교하기 위해 boxplot 형태로 표를 작성해 이미지 파일로 저장하는 함수이다.

3, 4, 5 함수의 결과를 표로 나타내기 위해선 3, 4, 5 함수를 실행한 결과를 가져와야 한다. 이는 다양한 방법이 존재하나 여기서는 각 함수들을 호출했을 때, 그 결과를 반환하여 그 결과를 직접 함수의 인자로 전달하는 방식을 사용하였다. 이를 위해 generate_final_figure() 함수를 다음과 같이 수정하였다.

```
def generate_final_figure(sresult, bresult, nresult):
```

또한 3, 4, 5 함수를 호출할 시 그에 대한 반환 값이 존재하기 때문에 3, 4, 5 함수를 호출하는 코드도 다음과 같이 수정하였고, generate_final_figure() 함수가 달라짐에 따라 이 함수를 호출하는 코드도 달리하였다.

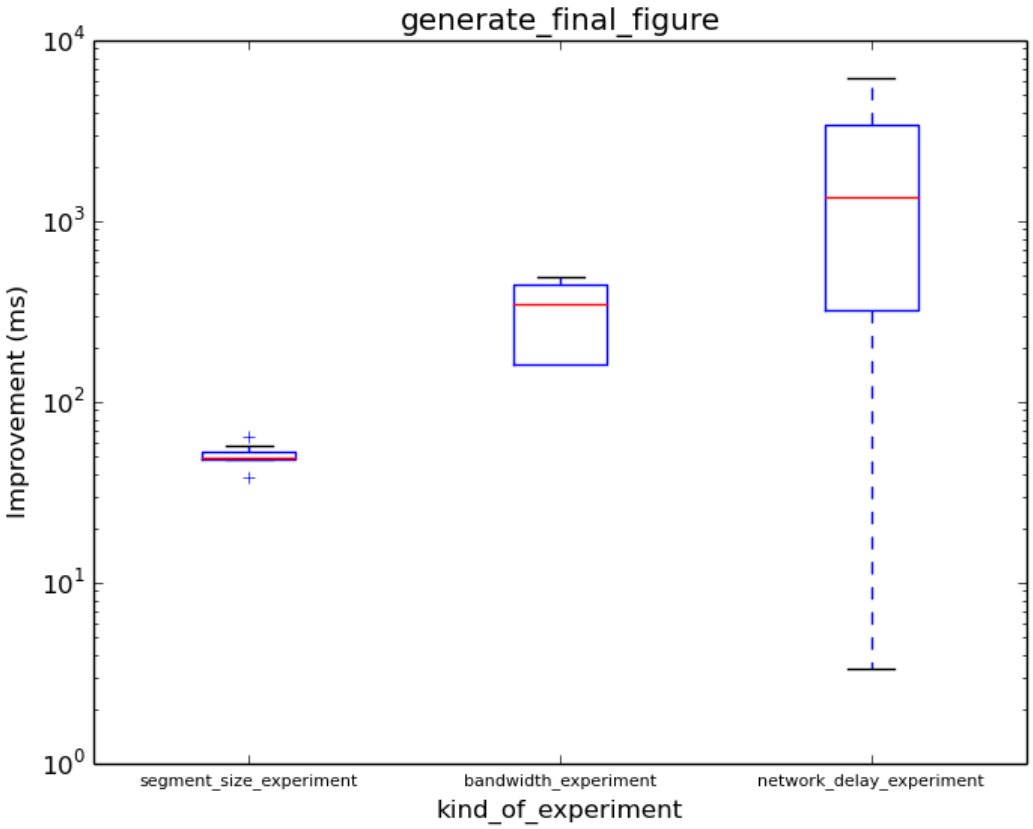
```
sresults = segment_size_experiment()
bresults = bandwidth_experiment()
nresults = network_delay_experiment()
generate_final_figure(sresults, bresults, nresults)
```

앞서 말했듯이 각 함수들을 호출 했을 때, 그 결과를 반환하는 식으로 수정했다고 하였는데, 이는 generate_figures () 함수와 3, 4, 5 각각의 함수에서 다음과 같이 수정하였다.

```
return abs_im
```

```
return generate_figures("Figure_Segment_Size_Experiment",
```

아래의 표는 segment_size_experiment() 함수를 실행시켜 얻은 결과이다.


generate_final_figure
Improvement (ms)
10^4
10^3
10^2
10^1
10^0
segment_size_experiment
bandwidth_experiment
network_delay_experiment
kind_of_experiment