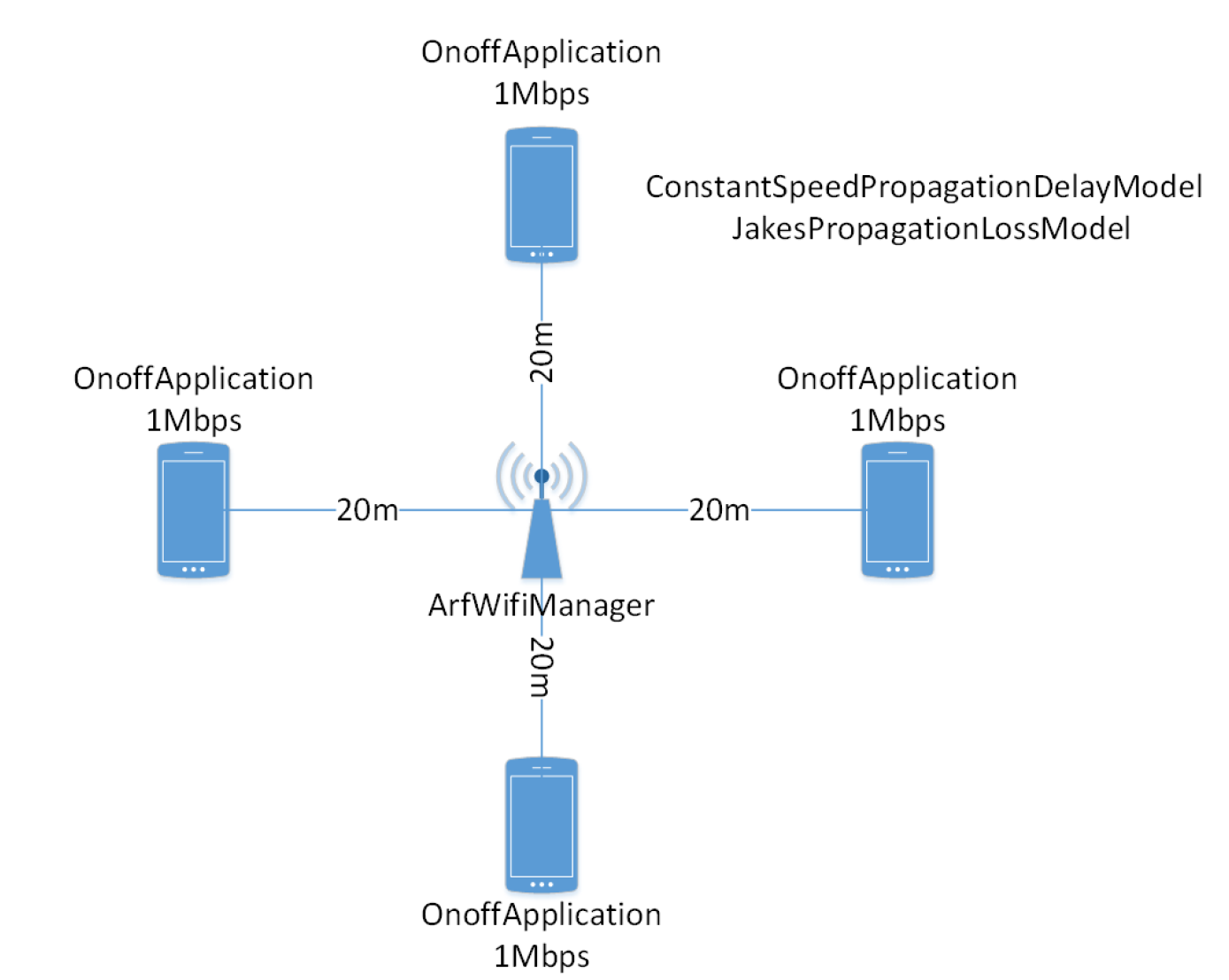
**네트워크 프로토콜 설계 및 실습 과제 6**

전기정보공학부 2016-10769 이권형

**1. Objective of experiment**

WIFI protocol에 대해, AP node에 Packet Sink application을 설치하고 각각 2개와 8개의 station에 OnOff application을 설치하여 traffic을 관찰한다. 이때 rate adaption algorithm을 ARF, CARA, Ideal하게 설정하여 비교하고, threshold를 0과 2346bytes로 설정하여 비교한다. 이를 ns-3를 이용한 simulation의 결과에서 wireshark를 이용해 total throughput, Physical layer data rate등을 관찰할 것이다. 이로써 ARF, CARA algorithm의 상황에 따른 장단점과 rate의 변화 stream을 확인할 수 있다.

**2. Topology & Realization**



**그림1.** week2.c의 topology

기본적으로 위의 topology와 같은 구조를 가졌다. 하나의 SinkApplication이 AP node가 되고, 다른 node들이 station node가 되어 OnOff application을 이용한다. 이때 OnOff application에서의 data rate은 30Mbps를 사용하고 packet size는 400bytes로 하였다. 배치는 위와 같지 않고 AP를 중심으로 하는 30m의 disc상에서 random하게 배치되며, RandomDirection2d의 mobility를 이용하여 이동한다. bound는 -30~30까지의 rectangle을 이용한다. 이 때 nWifi를 이용하여 station node의 개수를 2개와 8개로 조정할 수 있다. UDP/IP protocol을 이용하고 routing helper를 통해 이를 구현했다.

이를 구현하기 위한 code에 대한 대략적인 설명은 다음과 같다. 자세한 사항은 code의 주석에 표시되어 있다.

protocol은 WIFI를 사용하고, AP node 하나와 Station node를 nWifi개만큼 만든다. nWifi는 cmdline argument로, 기본적으로는 2개로 setting되어 있다. YansWifiChannelHelper를 통해 조건에 맞도록 channel configure를 하고, 이후 YansWifiPhyHelper를 통해 physical layer를 만든다. ssid는 ns-3-ssid를 이용하고, type을 AP WIFI MAC으로 mac을 설정한다. 이후 algo라는 argument에 따라 rate algorithm을 case문을 이용하여 설정한다. 각각 0부터 2까지, ARF, CARA, IDEAL로 rate algorithm을 설정한다. Mobility는 AP node의 경우 Constant position으로, 위치를 원점으로 하여 설정한다. STA는 모두 Random direction 2d model로 설정한다. Bound를 사각형으로 한 변의 길이가 60이 되도록 설정하고, 각 node의 위치를 Uniform Disc Position으로 rho가 30m 이내에 있는 disc위에 random하게 배치한다. OnOff application과 PacketSink application을 각각 AP node와 station nodes에 설치한 뒤, AP node는 1초부터 10초까지, 나머지 station node들은 2초에서 5초까지 작동한다. Enable Pcap으로 packet을 capture하고, simulation을 12초에 종료하도록 시작한다.

**3. Result & Discussion**

1) Compare the total throughput for 12 different cases

station node의 개수와 rate algorithm, Threshold를 바꾸어 가며 12개의 결과를 얻었다. 각각은 최대 throughput으로 하였다. 이는 I/O graph에서 눈금을 통해 측정할 수 있었으며, 다소 오차가 있을 수 있다.

|  |  |  |  |
| --- | --- | --- | --- |
| Threshold (bytes) | ARF | CARA | IDEAL |
| 0 | 14 | 14 | 15.5 |
| 2346 | 18 | 17.2 | 18 |

**표1.** nWifi=2에서의 throughput (Mbps)

|  |  |  |  |
| --- | --- | --- | --- |
| Threshold (bytes) | ARF | CARA | IDEAL |
| 0 | 14 | 14 | 15.5 |
| 2346 | 5 | 15.75 | 17.2 |

**표2.** nWifi=8에서의 throughput (Mbps)

표1을 보면, station이 2개일 때, ARF와 CARA는 threshold가 0일 때 같은 throughput을 나타냄을 알 수 있다. 또한 Ideal의 경우, 모든 경우에서 가장 높은 throughput을 보유했다.

threshold가 0bytes인 경우, 모든 경우에서 RTS/CTS를 사용하기 때문에 CARA algorithm의 경우에서도 ARF algorithm과 같이 RTS/CTS만을 사용하게 되었다. 따라서 위 경우 두 algorithm은 동일한 결과를 보인다. 이를 통해 threshold가 0이 되어 항상 RTS/CTS를 보내게 될 경우 CARA가 channel error와 collision error를 구분하지 않고, ARF와 같이 rate를 적용하는 것임을 확인할 수 있다.

threshold가 2346 bytes인 경우에서는 위와는 다르게 나타난다. ARF와 CARA에서 모두 threshold가 0일 때보다 throughput이 늘어났음을 확인할 수 있다. 또한, 그 정도가 CARA보다 ARF가 높게 나타났다. threshold가 앞에서 보다 높기 때문에, ARF에서는 RTS/CTS를 사용하지 않기 때문에 threshold가 0bytes일 때보다 throughput이 높아졌다. 이에 반해 CARA에서는 packet loss가 발생할 경우, RTS/CTS를 사용하기 때문에 ARF보다 RTS/CTS로 인한 overhead가 크게 측정된 것이다. 또한, RTS/CTS를 사용하였음에도 packet loss가 발생하고, 이를 channel error로 인지하여 rate을 낮추기 때문에 이로 인해서도 throughput이 감소한다. 따라서, threshold가 높아짐에 따라 RTS/CTS를 사용하지 않게 되고, 전체적으로 throughput이 높게 측정된 것이다.

표2를 보면, nWifi가 2일때와는 다른 특징이 관찰된다. ARF와 CARA에서, threshold가 0byte일 때는 nWifi가 2일 때와 같은 throughput을 보인다. Ideal은 여전히 가장 높은 throughput을 보유했다.

그러나 ARF의 경우 threshold가 2346bytes로 높아짐에 따라 throughput이 급격하게 감소한다. 이는 collision에 의한 것으로 추정되는데, ARF의 경우 packet loss가 발생하는 경우 이를 channel error와 collision으로 구분하지 않고 data rate을 감소시킨다. 따라서 station node가 8개일 때, threshold가 2346bytes일 때에는 RTS/CTS를 사용하지 않아 더 자주 collision이 발생하게 되고, ARF의 경우 이를 channel error로 인한 것으로 인식하여 data rate을 급격하게 낮추어 throughput이 감소하는 것이다. 이에 반해 CARA의 경우는 channel error와 collision error를 구분하기 위해, packet loss에서 CTS/RTS를 사용한다. 따라서 CARA는 collision이 자주 발생하는 상황에서 계속하여 CTS/RTS를 사용하면서 data rate을 유지하기 때문에 위와 같이 ARF에 비해 throughput이 높게 나타나는 것이다.

station node의 개수가 2개일 때보다 8개일 때 CARA에서의 throughput 증가량이 줄었다. 이는 collision에 의한 것으로 추정되는데, CARA에서는 packet loss가 일어날 때마다 RTS/CTS를 사용하므로 collision이 더 자주 발생함에 따라 RTS/CTS의 overhead가 증가할 것이다. channel error rate가 일정할 때, 위와 같이 collision이 증가하여도 throughput이 낮아지는 것을 확인할 수 있다.

IDEAL algorithm은 ns-3에서 구현된 것으로, packet의 SNR을 모두 알며, threshold의 value를 target SNR/BER curve에 따라 data rate을 조절한다. 따라서 이 경우, packet loss의 최소화를 위해 scheduling이 효율적으로 이루어지며, collision의 빈도가 낮게 나타나서 throughput이 높게 측정되는 것이다.

2) Compare the ratio of PHY rate for 8 different cases

다음은 capture된 packet에서, radiotap.dtarate filter를 이용하여 각 data rate마다 packet의 개수를 이용해 data rate ratio를 측정하여 표로 나타낸 것이다. packet의 개수는 filter를 이용하여 displayed된 개수를 status bar를 통해 확인한 것이다. 단위는 모두 %이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threshold (bytes) | 0 | | 2346 | |
| Data rate (Mbps) | ARF | CARA | ARF | CARA |
| 6 | 50.3 | 50.3 | 0.8 | 11.6 |
| 9 | 0.1 | 0.1 | 0.1 | 0.1 |
| 12 | 0.2 | 0.2 | 0.2 | 0.2 |
| 18 | 0.1 | 0.1 | 0.1 | 0.1 |
| 24 | 24.7 | 24.7 | 49.6 | 44.1 |
| 36 | 0.1 | 0.1 | 1.1 | 0.1 |
| 48 | 0.1 | 0.1 | 6.2 | 0.1 |
| 54 | 24.5 | 24.5 | 41.8 | 43.8 |

**표3.** nWifi=2에서의 data rate ratio 측정 [단위: %]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threshold (bytes) | 0 | | 2346 | |
| Data rate (Mbps) | ARF | CARA | ARF | CARA |
| 6 | 50.9 | 50.9 | 88.1 | 27.8 |
| 9 | 0.2 | 0.2 | 7.7 | 0.3 |
| 12 | 0.6 | 0.6 | 3.1 | 0.8 |
| 18 | 0.2 | 0.2 | 0.2 | 0.3 |
| 24 | 24.3 | 24.3 | 0.7 | 35.7 |
| 36 | 0.2 | 0.2 | 0.1 | 0.3 |
| 48 | 0.2 | 0.2 | 0 | 0.3 |
| 54 | 23.4 | 23.4 | 0 | 34.6 |

**표4.** nWifi=8에서의 data rate ratio 측정 [단위: %]

위 결과는 각각 station node를 2, 8개에서 algorithm과 threshold를 변화시키며 capture packet에서 data rate의 ratio를 측정한 것이다. 전체적으로, 특정 구간에 data rate이 집중되어있음을 확인할 수 있다.

우선 threshold가 0byte일 경우 모든 경우에서 rate ratio가 거의 동일하게 측정되었다. 이는 앞서 보았듯, threshold가 0byte이므로 RTS/CTS를 사용하게 되어 CARA와 ARF가 동일하게 적용되는 것이다. 또한, data rate이 6Mbps인 packet들은 대부분 RTS/CTS인 것을 확인할 수 있었다. 즉, RTS/CTS는 6Mbps만을 사용하고, 이로 인해 한번의 transmission에서 1 pair의 RTS/CTS가 필요하기 때문에 개수의 ratio가 50%정도로 나타나는 것이다. 이외의 50%는 packet과 ACK일 것이다.

또한, 24Mbps의 rate을 보면 대부분의 packet이 ACK임을 확인할 수 있었다. 따라서 나머지 50%중 ACK과 실제 transmission을 위해 사용된 packet으로 나뉘는데, ACK은 24Mbps에서, transmission에 사용된 UDP packet은 54Mbps에서 전송되었을 것이다. station의 개수가 8개인 경우에도 마찬가지로, collision을 방지하기 위해 CTS/RTS가 50%, ACK이 25%, 나머지 UDP packet이 25%정도를 사용한다.

threshold가 2346일 때는 위와 다른 결과를 확인할 수 있다. station의 개수가 2개인 경우, ARF와 CARA가 비슷한 결과를 보여주는데, 6Mbps에서의 packet 전송이 사라진 것을 알 수 있다. 즉, 이는 RTS/CTS가 거의 사용되지 않았음을 의미한다. 따라서 대부분의 packet이 UDP data와 ACK이며, 각각 50%정도로 24, 54Mbps에 분포해 있다. CRAR의 경우 6Mbps가 11%정도로, CTS/RTS가 조금 사용되었다. 이는 CARA에서 packet loss가 발생할 때 CTS/RTS를 사용해 collision에 의한 것인지 channel error에 의한 것인지 확인하기 위함으로, 1)에서의 discussion과 같이, 이로 인한 overhead가 발생하여 throughput이 나타났을 것이다.

station node의 개수가 8개일 때, ARF의 경우 대부분의 packet이 6Mbps의 rate에 분포해 있음을 확인할 수 있다. 이는 UDP data 또한 6Mbps로 전송되었음을 의미한다. 계속되는 collision에서, ARF의 경우 packet loss가 2번 일어날 때마다 rate을 줄이므로 계속하여 rate가 오르지 못하는 것이다. 반면에 CARA의 경우 collision이 일어날 때 RTS/CTS를 이용하고, channel error임이 아님을 알게 되어 data rate을 유지하므로 54Mbps로 전송되는 비율이 높다. collision이 발생할 때마다 RTS/CTS를 사용하고, 이로 인해 data rate은 54Mbps로 유지되지만 CTS/RTS에 의한 overhead가 발생함을 확인할 수 있다.

3) Tracing PHY rate of ArfWifiManger

**그림2.** UDP data에 대한 rate trace graph. x축은 time(sec)이고, 가로축은 rate(Mbps)이다.

**그림3.** ACK에 대한 rate trace graph. x축은 time(sec)이고, 가로축은 rate(Mbps)이다.

그림2와 3은 각각 UDP data와 ACK의 rate을 측정하여 시간에 대해 그래프를 그린 것이다. 위의 실험과 같은 상황에서 nWifi을 4로 하고, ARF algorithm을 사용하여 측정하였다.

앞에서 서술했듯, UDP data rate은 6~54Mbps에 고루 퍼져 있으며, ACK data의 경우 대부분이 6Mbps에, 나머지는 12, 24Mbps에 분포해 있다. 이는 collision에 의한 것으로, station node가 2개일 때 54Mbps에서 대부분이 전송되고, 8개일 때는 6Mbps에서 대부분이 전송되던 것과 다른 결과를 보여준다. 적당히 data rate을 조절하면서 전송하고 있음을 알 수 있다. ARF algorithm의 경우 두번의 연속된 miss에서 data rate을 줄이고, 10개의 성공적인 ACK 이후에 data rate을 증가시킨다. 직접 data를 조사한 결과, 하나의 rate은 최소한 두 개의 packet에 대해 유지되었고, 일정 개수의 packet전송 이후에 rate이 증가함을 확인할 수 있었다.

ACK의 경우 6, 12, 24Mbps에만 분포하는 것으로 보아, 그 크기가 작아 rate이 높지 않은 것이 안전하므로 protocol에서 그렇게 설정된 것으로 보인다. ACK 또한 channel의 상태에 따라 증가하고 감소하는 것을 확인할 수 있다.

이로써 ARF algorithm에서의 rate조절을 graph를 통해 볼 수 있었다.

**4. Reference**

- Kurose, J. F. and K. W. Ross (2012). Computer Networking: A Top-Down Approach (6th Edition).

- ns3 network 2주차 강의 자료.

- ns-3 Doxygen. <https://www.nsnam.org/doxygen/>