1

안녕하세요 팀 11입니다 발표 시작하겠습니다.

저희는 유사 영상 스트리밍 모델을 만들었습니다. 시나리오 파일도 만들어 봤고 프로토콜도 만들고… 그런 활동을 하고 그랬어요. 그럼 한번 프로토콜부터 보도록 하죠

.

4

저희 팀에서는 오디오나 비디오 데이터의 스트리밍을 위해 사용되는 Real time protocol, 즉 RTP라는 프로토콜에서 아이디어를 얻어서 리얼타임 스트리밍을 위한 간단한 프로토콜을 추가했습니다.

.

7.

먼저 기존의 RTP를 보게 되면, UDP 위에서 동작하고, RTP Packet Header를 통해 Sequence number와 Timestamp 등의 정보를 함께 전송해 UDP에서 발생할 수 있는 Packet loss에 대해서 빠르게 Retransmit을 할 수 있도록 지원하고 있습니다.

8.

저희가 구현한 RTP 헤더는 기존의 헤더에서는 저희가 제작한 비디오 스트리밍 어플리케이션에 필수적인 Sequence number만 남겨두고 각 프레임을 구분할 수 있도록 이번 프레임의 마지막 시퀀스 넘버를 알려주는 값을 추가해 주었습니다. 그리고 Sequence number의 크기를 기존의 16비트에서 32비트로 확장해 충분히 큰 Sequence number도 스트리밍 어플리케이션에서 사용할 수 있도록 해 주었습니다.

이렇게 구현된 헤더는 기존의 12바이트에서 4바이트 축소된 8바이트의 헤더 크기를 갖습니다.

9.

지금 보시는 코드들은 구현된 RTP 헤더 소스 코드의 일부인데요, 각 필드는 32비트의 unsigned integer 값으로 구현했고, 재전송이 필요 없을 때 sequence number를 0으로 설정해 보내주는 방식으로 사용되는 것을 확인할 수 있습니다.

10.

앞서 말한 rtp를 이용하여 저희 팀은 reliable streaming을 구현해 보았습니다. 실은 video 라기보단 사진을 이용한 건데... 함 보시죠

11

일단, 사진이 있습니다. 오른쪽에 스윙스랑 삼장이 보이는데 얘네 두장으로 진행했습니다. 총 12장이 있고, 각각 6개씩 고화질부터 저화질의 사진이 input으로 받아집니다.

그리고 frameFile 이란 이름으로 두 사진들의 이름이 input에 들어오는데, - 예를 들어 스윙스 삼장 스윙스 스윙스 삼장 - frameRate가 25면, 이 25줄이 그게 1초짜리 영상이 되는 겁니다.

12.

즉, 이렇게 25frame에 1초 영상으로 취급하는 겁니다.

마치 옛날에 디즈니 애니메이션처럼 말이죠. 1초 동안 25번 두 사진이 파라라락 바뀌는 영상인 거예요.

저희 model은 이 영상을 서버에서 클라로 전달하는 것을 목적으로 합니다.

13.

클라는 frame을 쌓아 놓다가 25개 이상이 되면 1초 분량의 사진 25개를 -원래는 화면에 보여줘야 하겠지만- 파일로 저장합니다.

14.

이때 클라이언트가 판단을 합니다. 적절한 양의 frame이 오면 딱 좋겠죠. 그치만 아닐 수도 있습니다. 여기서 영상이 buffering이 많이 걸리면, 즉 frameRate만큼의 이미지를 1초에 못 받아서 출력을 못하면, 좀 더 저화질로 보면은 더 끊김 없이 볼 수 있으니까 용량이 작은 걸로 달라고 서버에 말을 해줍니다.

반면 막 버퍼에 5초 이상의 영상이 쌓여 있으면 솔직히 좀 더 고화질로 보아도 괜찮다는 이야기니까 큰 파일, 즉 고화질의 영상을 달라고 하고요.

15.

자 이제 frame은 어떻게 보내느냐의 문제입니다.

보통은 이미지 크기가 packet 크기보다 큽니다. 그렇기 때문에

16.

이렇게! 패킷 여러개에 나눠 보내야 하는데, 그때 rtp헤더를 써서 reliable하게 구현합니다.

17.

frame 각각에 seq번호를 부여합니다. 그리고 그 정보를 rtp 헤더에 그 정보를 써 놓는 거예요. 예를 들어 이 패킷은 1 - 8 seq 인 frame의 2번째 seq 인 패킷이다. 이런 느낌으로요. (이게 괄호가 있는 건 아-까 전에 마지막 seq 넘버를 써 놓기 때문이에요. 즉 실은 8 하나밖에 안 써 놓습니다)

18.

그러면 클라는 8 값을 저장해 놓고 후에 8이나 그보다 큰 패킷이 오면. 오.. 내가 frame을 구성할 1-8의 패킷이 전부 있나? 를 검사하는 거죠. 근데 예를 들어 5가 없어요? 그럼 서버한테 달라고 합니다.

다만 여기서 '패킷이 올때', frame 구성을 진행하기 때문에 막판에 패킷이 전부 다 우수수 떨어지면 frame 구성을 안해서, 초반 부 영상만 남는 경우도 생기는데, 어차피 streaming이라 저희 예제에서는 그렇게 큰 문제가 생기지 않을 것 같습니다. 끊임없이 영상이 오니까요. 원래..

19.

추가적으로 서버는 만약에, 갑자기 클라가 죽으면, 즉 신호가 없으면 시간을 셌다가 그 클라에게의 전송을 멈추는 기능도 가지고 있어요. 이건 RTP랑 크게 관련 없는 부가적인 기능입니다.

20

이제 저희 시나리오 파일 이야기를 좀 할게요..

22.

라우팅 알고리즘은 다음과 같이 다양한 종류가 있는데, 저희는 이 중에서 Shortest path routing의 대표적인 예시인 다익스트라 알고리즘을 선택하여 구현해보았습니다.

23.

다익스트라 알고리즘은 음수 가중치를 갖지 않는 그래프에서 주어진 출발점과 도착점 사이의 최단 경로문제를 푸는 알고리즘입니다. 출발지로부터 최단 경로를 갖는 점들을 차례대로 찾아가면서 경로를 탐색합니다. 네트워크 토폴리지와 링크 비용이 모든 노드에 알려진다는 특징이 있습니다. 즉 모드들이 같은 정보를 가지고 있다고 가정합니다.

24.

자 이게 Input 파일과 output 경로입니다.

프로그램에 텍스트 형식으로 topology 정보가 입력이 들어옵니다. 첫 줄에 node의 개수(nodenum)와 간선의 개수(bridgenum)를 의미하는 두 개의 정수가 주어집니다. 그 다음 간선의 개수(bridgenum)만큼 간선에 대한 정보가 한 줄에 하나씩 주어지며, 각 줄은 시작점, 끝점, 간선의 cost를 의미하는 세 개의 정수가 주어집니다.

프로그램 내부에서 이 정보를 토대로 다익스트라 알고리즘으로 최단 경로와 길이를 구한 후 출발지로부터 목적지까지의 최단 거리와 경로를 이차원 vector 형식으로 나타내도록 했습니다.

이러한 경로를 바탕으로 네트워크 시뮬레이션이 돌아가도록 구현하였습니다.

25

이번 프로젝트에서 구성한 topology에 대해 설명하겠습니다.

26

client와 server는 각각 1대 씩 설치되고 router topology는 다익스트라 알고리즘을 통해 추출된 최적 경로를 통해 구성합니다.

27

topology를 연결하는 방식 중 wifi에 대한 부분의 핵심 코드를 설명하면 다음과 같습니다.

router는 packet을 보내는 기능과 받는 기능이 동시에 필요합니다.

이를 구현하기 위해 router를 표현하는 하나의 node에는 client와 server가 동시에 설치됩니다.

2번째 for문에서는 다익스트라 알고리즘을 통해 추출된 경로에 따라 client를 설치합니다.

28

p2p 연결 또한 전체적인 내용은 wifi의 경우와 비슷합니다.

하지만, 노드와 노드 사이의 연결을 일일이 선언하고 설정해줘야 하기 때문에,

container 및 helper를 벡터로 선언하여 각각 담아주었습니다.

현재 보고 계시는 슬라이드는 그 예시중 하나인 p2pPointHelper이며

input으로 받은 간선 별 거리의 cost를 delay 속성으로 반영하여 값을 넣어주었습니다.

29.

다음은 각 연결에 주소를 깔아주는 코드입니다.

연결 별로 세 번째 자리수를 다르게 하여 ip주소가 중복으로 배정되는 오류를 방지했습니다.

30.

p2p 형식으로 연결한 여러 노드에 클라이언트 및 서버 어플리케이션을 설치하는 코드는

wifi와 같은 방식을 사용했습니다. 이상으로 topology에 대한 설명을 마치겠습니다.

31.

그래서 결과가 이렇게 나옵니다. 이게 wiFI 고, 보시면 1초 동영상이 제대로 나온 걸 확인할 수 있고,

32.

P2P에서는 4초에 해당하는 영상이 제대로 video파일에 저장된 걸 확인할 수 있습니다.

(한참 후)

33.

감사합니다.