문제 정의서

1. 연구의 필요성

가. TCP 및 UDP

QUIC은 TCP를 대체하기 위해 만들어지는 프로토콜이므로 먼저 TCP를 살펴 볼 필요성이 있다.

TCP는 이메일 전송, 파일 전송, HTTP 및 HTTPS 통신을 수행하는 등 신뢰성 있는 통신을 보장하는 국제 표준 프로토콜으로, IETF의 RFC 793에 등록되어 있다. 그러나 TCP는 보안을 생각하여 만들어진 프로토콜이 아니다. 현재 TCP로 HTTPS 보안 통신을 수행하기 위해서는 Three-way handshake 과정 및 TLS handshake 과정을 거쳐야 하는데, 이 과정에서 7회 이상의 패킷 왕복이 필요하기 때문에 매우 느린 연결 시간을 가지게 되고, 이런 문제 때문에 스트리밍 등의 서비스를 구현할 때는 TCP가 아니라 UDP 통신을 사용하는 것이 일반적이다.

UDP는 TCP보다 더 적은 헤더를 가지는 비연결형 프로토콜이다. 그래서 위의 문제점을 해결하기 위해 UDP를 사용할 수 있지만 이 경우에는 신뢰성 및 호환성에 대한 문제가 발생한다. UDP는 패킷 손실 및 혼잡 제어에 대한 대책을 가지지 않은 프로토콜이므로 유저 레벨에서 구현해주어야 하고, 기본적으로 http 및 https를 지원하지 않아서 UDP 통신만으로 모든 유형의 통신을 수행하기에는 무리가 있다.

나. QUIC 프로토콜의 장점

QUIC 프로토콜은 TCP 프로토콜에서 불필요한 패킷 왕복을 하나의 패킷으로 축소시켜 만든 UDP 기반의 프로토콜으로, TCP와 같은 정도의 신뢰성, TLS 암호화 및 UDP 통신만큼의 연결 속도를 가진다.

QUIC 프로토콜의 신뢰성은 TCP의 혼잡 제어 및 손실 복구 알고리즘을 거의 그대로 사용함으로서 기존 TCP의 신뢰성 및 호환성을 그대로 유지할 수 있다는 장점이 있다.

보안 연결을 위해 기존의 TCP 연결에서는 TLS를 사용한다. TLS 연결으로 인해 통신 내용 감청을 예방할 수 있게 되었으며, 이는 QUIC 프로토콜에서도 동일하다.

반면 TCP에서는 TLS를 사용하기 위해 몇 개의 추가적인 패킷을 필요로 하여 연결 수립에 필요한 시간이 늘어난 데 비해 QUIC 프로토콜에서는 이러한 암호화 연결을 단 하나의 패킷으로 대체하므로 0~1 RTT에 해당하는 시간만을 연결 수립에 사용한다.

QUIC은 또한 모바일 환경에 알맞게 네트워크 환경이 변화하더라도 빠르게 새로운 경로로 재연결할 수 있도록 하는 대응책도 포함하는 프로토콜이다.

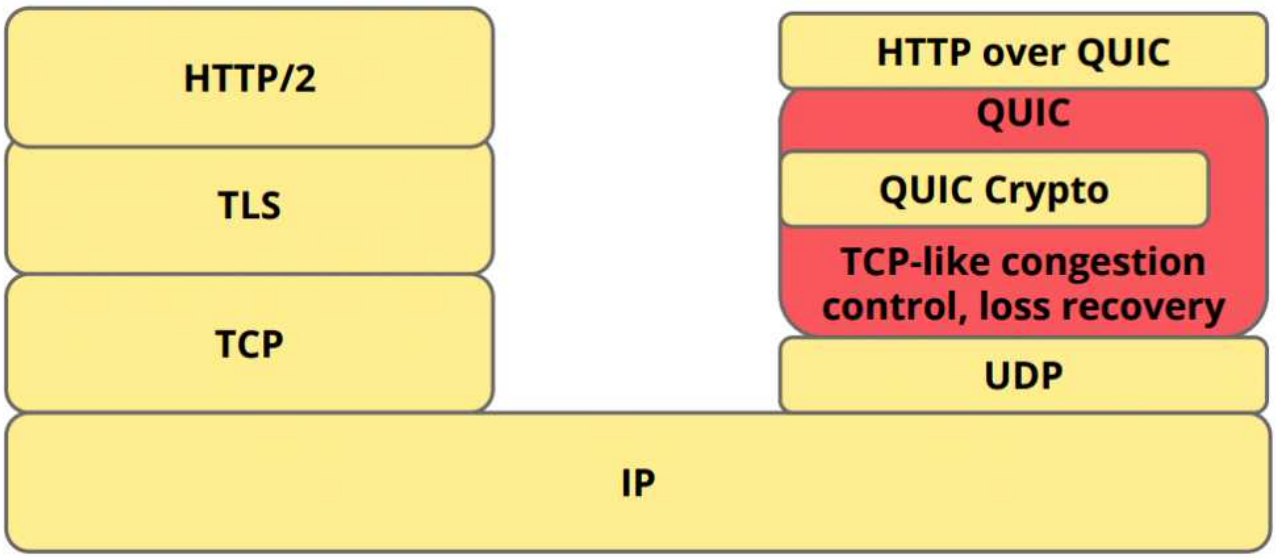

그림 1 QUIC 프로토콜 헤더

다. 국내/외에서의 QUIC 프로토콜

QUIC 프로토콜은 현재 Chrome 브라우저 및 Google, Youtube 등 구글에서 운영하는 페이지들에 사용되고 있다. 이번 프로젝트에서도 사용하게 될 Chromium 오픈소스에 QUIC을 구현하기 위한 튜토리얼이 포함되어 있으며, 이를 통해 구글을 중심으로 한 각종 포럼에서 활발하게 연구가 진행되고 있다.

구글은 QUIC 프로토콜을 국제 표준으로 IETF(국제 인터넷 표준화 기구)에 제안하였으며, RFC 파일이 IETF 홈페이지에 Proposed Standard(표준으로 제안됨) 상태로 올라와 있는 것을 확인할 수 있다.

국내 게임 개발사인 데브시스터즈의 Github 리포지토리에서도 QUIC을 GO 언어로 구현한 것을 확인할 수 있다. 이는 TCP에 비해 QUIC이 연결 수립에 소요되는 시간이 짧아 더 빠른 게임 접속 및 끊김없는 플레이를 가능하게 하기 때문이라고 한다.

2. 연구의 범위 및 목표

1. 연구 목표

이 연구의 목표는 Java 언어를 이용하여 UDP를 기반으로 한 QUIC 서버 및 클라이언트를 제작하여 실제로 두 지점 사이에 통신을 수행하고 TCP보다 나은 RTT 횟수를 확보하는 데 있다.

1-1. 세부 목표

- QUIC 패킷을 패킹하고, 언패킹하는 기능을 구현하여 QUIC 패킷을 가지고 통신할 수 있도록 동작하는 코드를 구현한다.
- QUIC 패킷에 대하여 혼잡 제어를 수행하여 TCP 통신처럼 네트워크 혼잡이 발생할 시 정상 동작하도록 구현한다.
- QUIC 패킷에 대하여 손실 복구 기능을 구현하여 패킷이 손실되었을 경우 손실 패킷을 다시 전송하도록 하는 기능을 구현한다.
- TLS 암호화를 이용하여 패킷 감청에 대해 안전한 프로토콜을 구현한다.
- SPDY 프로토콜의 동기식 패킷 송수신과 비교하여 한 개의 Connection에서 에러가 발생하더라도 다른 Connection은 계속 동작할 수 있도록 각 Connection들 간에 영향을 주고받지 않도록 구현한다.
- QUIC 프로토콜을 구현하여 TCP 연결 속도와 비교해 보고 더 나은 연결 속도를 확인한다.
- 이 연구를 통해 만들어지는 Java QUIC 서버/클라이언트 코드를 라이브러리 형태 혹은 타 오픈소스에 포함하는 형태로 배포하여 QUIC 프로토콜이 많이 사용될 수 있도록 한다.

2. 연구 범위

- Chromium 코드 중 QUIC 관련 부분 분석 및 내용 정리
- Linux Kernel 코드 중 TCP 관련 부분 분석 및 내용 정리
- Java로 QUIC 패킷 패킹, 언패킹 코드 구현
- Java로 TCP Cubic 구현
- Java로 Packet loss 복원 기능 구현

3. 연구 내용

☞ 작성내용
1) 연구개발의 구체적 내용
2) 적용하게 될 연구방법

이 연구는 구글 크로미엄 및 리눅스 소스 코드를 분석하여 QUIC 프로토콜을 자바로 구현하는 목표를 가지고 있다.

QUIC 구현을 위해 분석해야 할 코드는 크로미엄 QUIC 코드, Devsisters QUIC 코드, 리눅스 TCP 코드이다. 이 중 크로미엄 코드는 C++로 구현되었고, Devsisters 코드는 GO 언어이며, 리눅스는 C를 사용하고 있다. 따라서 Java로 이 코드를 구현하기 위해 이 언어들에 친숙해질 필요가 있다.

구현을 위해서는 Agile Model에서 추구하는 개발 방법론을 최대한 따라서 가장 작은 목표로 쪼개서 각 단계마다 가치 있는 결과물을 산출하기 위해, 이 문단 다음부터 추가된 내용들은 각각의 단계가 중간 완성품이 될 예정이다.

먼저 구현해야 할 것은 QUIC 패킷을 만들고, 분해하는 코드(패킹/언패킹)이다. 이를 위해서 크로미엄 혹은 Devsisters의 QUIC 코드를 분석해야 하며, QUIC 패킷에 필요한 헤더 및 암호화를 추가하여 QUIC 패킷을 로컬에서 성공적으로 만들고, 분해하는 것이 이 단계의 최종 목표이다.

QUIC 패킷 패킹/언패킹 단계가 완료되었다면 QUIC 패킷을 보낼 수 있다. 하지만 QUIC은 TCP처럼 Connection 기반의 통신 프로토콜이므로 클라이언트에서 서버로 보내는 첫 패킷 및 서버에서 클라이언트로 되돌려주는 패킷을 만들어야 한다.

QUIC 프로토콜에서는 TCP에서 사용하는 손실 복구 및 혼잡 제어 알고리즘을 그대로 사용한다. 혼잡 제어 알고리즘은 TCP에서 사용하던 알고리즘 중 하나를 선택할 예정이다.

손실 복구 알고리즘은 FEC를 사용한다. QUIC의 FEC는 모든 패킷이 도착하지 않았을 경우 여분의 바이트를 전송하여 redundancy를 제공한다. XOR 기반 시스템을 사용하므로, 해당 XOR FEC 패킷의 그룹에 속해 있는 패킷이 도착하지 않았을 시 한 개의 패킷이 손실되었을 경우에 한해 복구할 수 있다. 이는 패킷 손실 시 재전송으로 인해 발생하는 blocking을 방지할 수 있어 네트워크의 성능이 하락하는 현상을 방지할 수 있다.

최종적으로 구현된 QUIC 서버, 클라이언트를 해외 서버 및 국내 서버(혹은 개인 컴퓨터)에서 통신 서비스를 수행하여 작은 데이터, 중간 데이터, 큰 데이터에 대하여 전송 시간을 비교하여 구현된 QUIC 프로토콜 서버/클라이언트가 네트워크 성능을 개선할 수 있다는 점을 확인한다.

4. 연구 팀의 구성 및 과제 추진 일정

☞ 작성내용
1) 연구진 구성 및 역할에 대하여 기술함
2) 추진일정 (1학기, 2학기 모두 포함한 전체 일정과 1학기 상세 일정 분리하여 작성)

1. 연구진 구성

- 정진욱 : 팀장, 서버 및 클라이언트 구현, 발표 및 문서 제작
- 신동령 : 서버 및 클라이언트 구현(종합설계 안 들음)

2. 추진일정

2-1. 전체 일정

	3월	4월	5월	6월	7월	8월	9월	10월	11월
Google Chromium 코드 분석	o	o	o	o	o				
Linux Kernel 코드 분석		o	o	o	o	o			
QUIC 패킷 패킹/언패킹 코드 작성		o	o	o	o				
혼잡 제어 코드 작성			o	o	o	o			
손실 복구 코드 작성			o	o	o	o			
추가 구현 및 에러 처리						o	o		
대회 참여, 보고서 작성							o	o	o

2-2. 1학기 일정

3월 18일 : 문제 정의서 및 요구사항 명세서 발표

4월 8일 : 유스케이스 모델링 발표

4월 22일 : 시퀀스 다이어그램 모델링

6월 3일 : 클래스 다이어그램 모델링

※ 반드시 5쪽 이상 10쪽 이내로 작성