QUIC을 이용한 네트워크 성능 개선

이준성
데브시스터즈 개발팀

DEVSISTERS

이준성a.k.a 호떡

- KAIST 전산학과
- ACM-ICPC World Finalist
- Devsisters 에서 쿠키런 서버 만들고 있음

Why QUIC?

쿠키런 by 데브시스터즈

쿠키런 for Kakao

- 한국에서 2,600만 다운로드, 최고 300만 DAU 기록
- 한국의 탄탄한 모바일 망을 기반으로 안정적으로 서비스 중

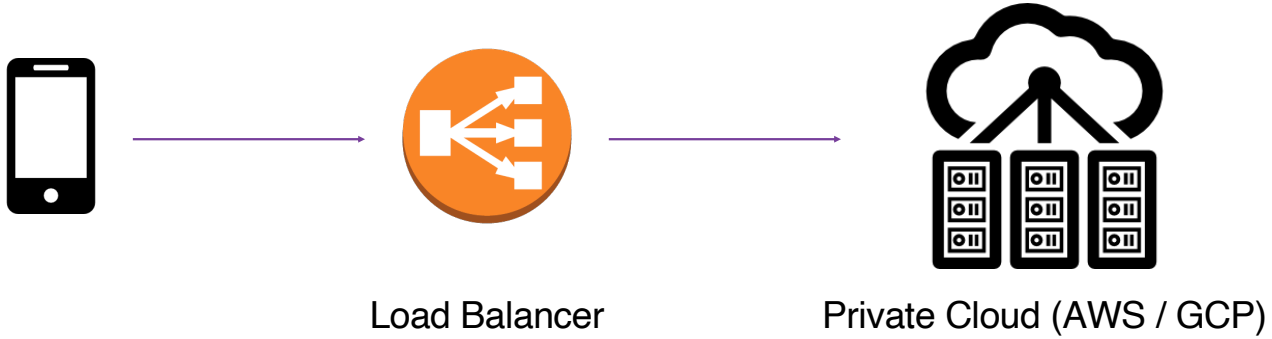
LINE COOKIERUN

- 일본, 태국, 대만 등에서 6000만 다운로드 돌파
- 네트워크 끊김에 대한 CS 꾸준히 접수

Cookie Run: OvenBreak

- 2016년 10월 27일 출시 예정
- 글로벌 원 서버 / 원 빌드

모바일 게임의 일반적인 구조

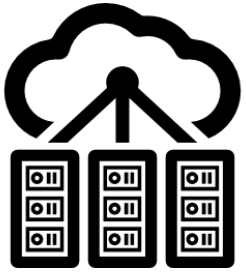


웹 서비스 + 모바일 게임의 흔한 구조

모바일 게임의 일반적인 구조

Load Balancer

신뢰 가능한 구간
상대적으로 빠르고 안정적이다
(물론 여전히 많은 문제가 있지만)

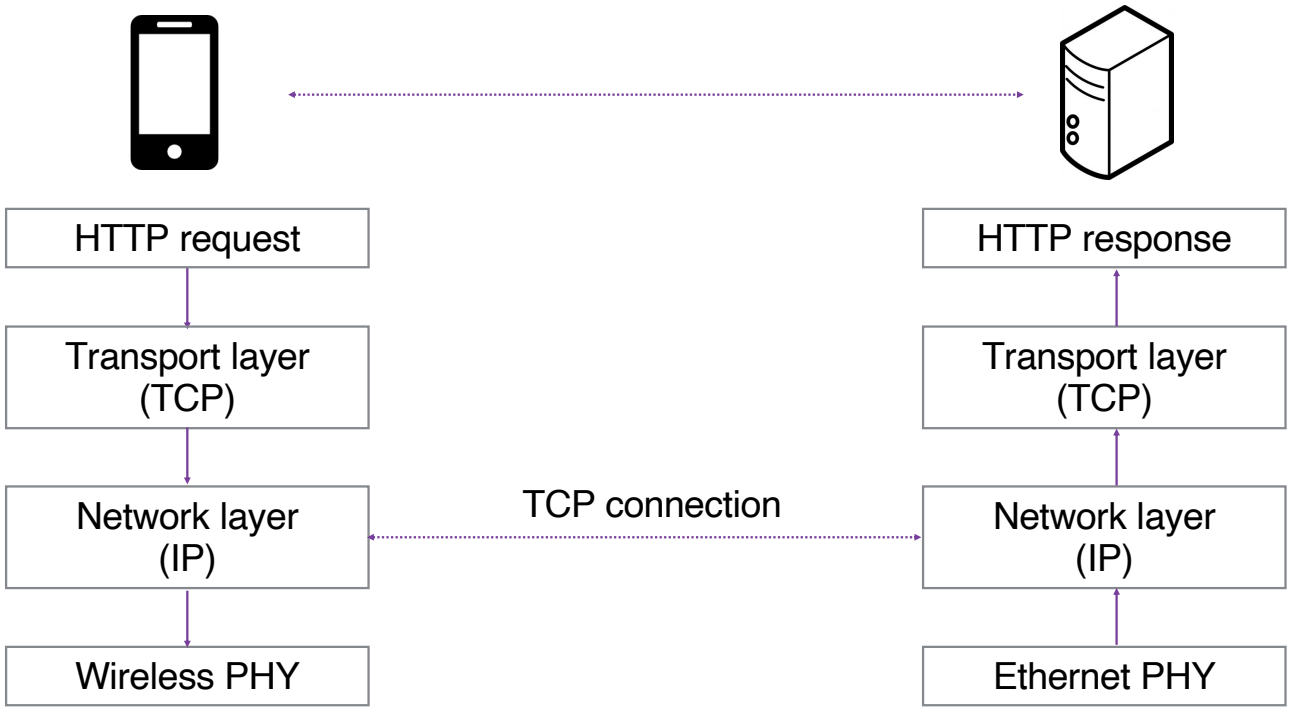


Private Cloud (AWS / GCP)

모바일 게임의 일반적인 구조

무슨 일이 일어날 지 모른다
매우 느리거나 자주 끊길 수 있다
전국 어디서나 LTE 뻥뻥 터지는 나라는 세상에 몇 개 없다
(한국 일본 싱가포르 홍콩?)

모바일 게임의 일반적인 구조



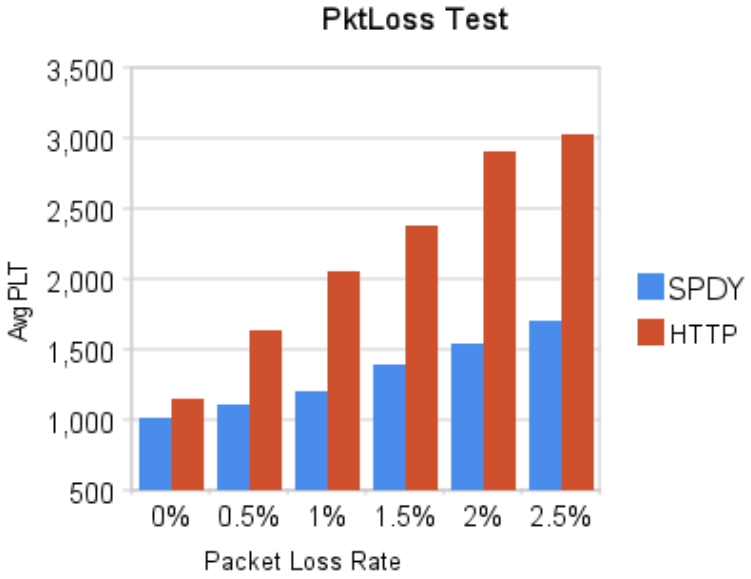
Issues in HTTP/1.1

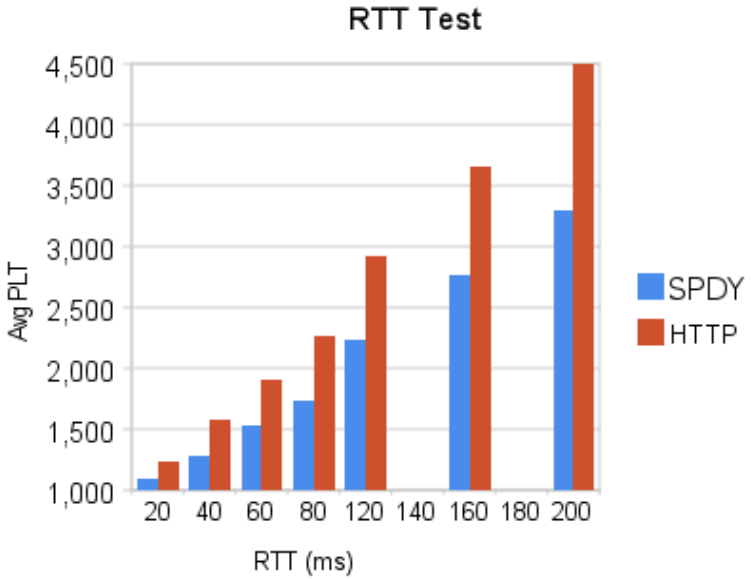
- Head-of-line blocking (HoL)
  - Single request per connection
  - Request must be responded in order (FIFO)
- Not optimized well
  - Uncompressed headers
  - Redundant headers
- Lack of Server-side push

SPDY & HTTP2

- Multiplexed streams
  - Multiple streams over single TCP connection
- Request Prioritization
- HTTP header compression
  - HPACK format
    - Huffman code
- Server Push

SPDY & HTTP2





https://www.chromium.org/spdy/spdy-data

모바일에서는?

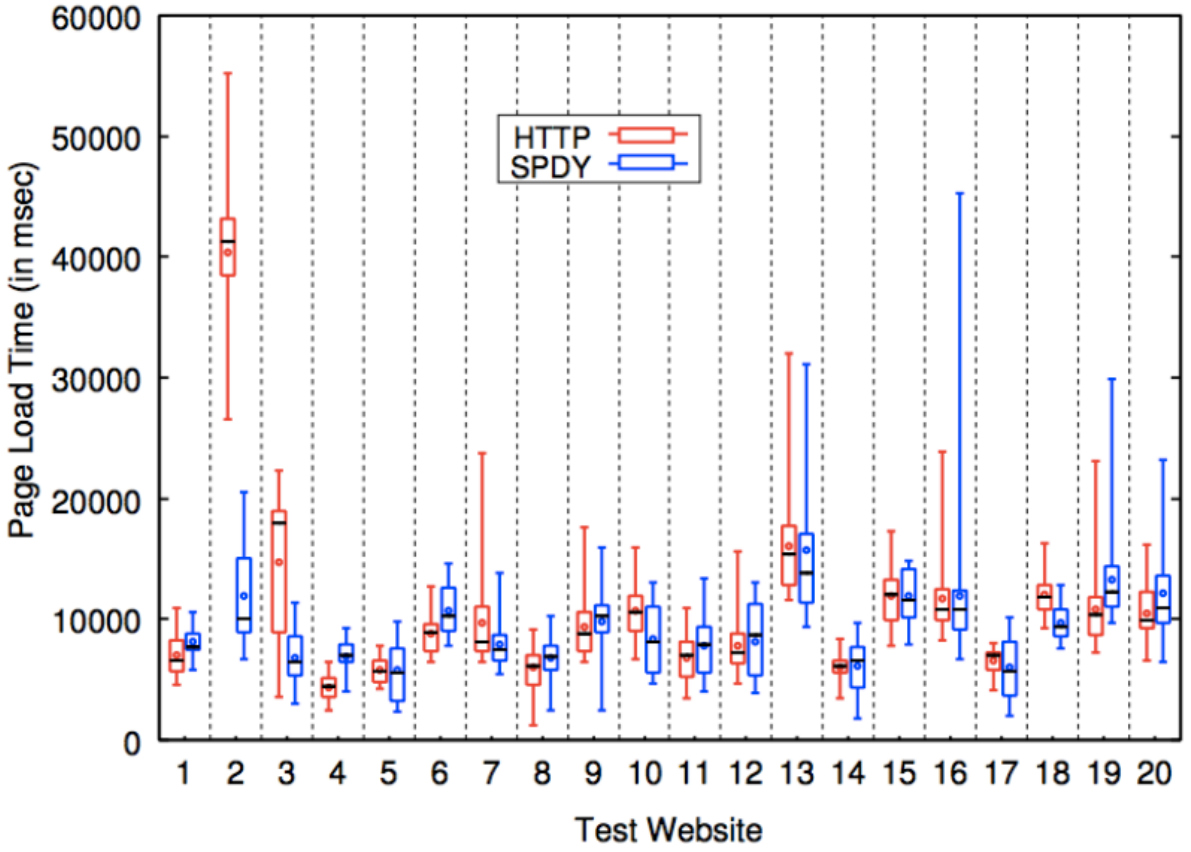
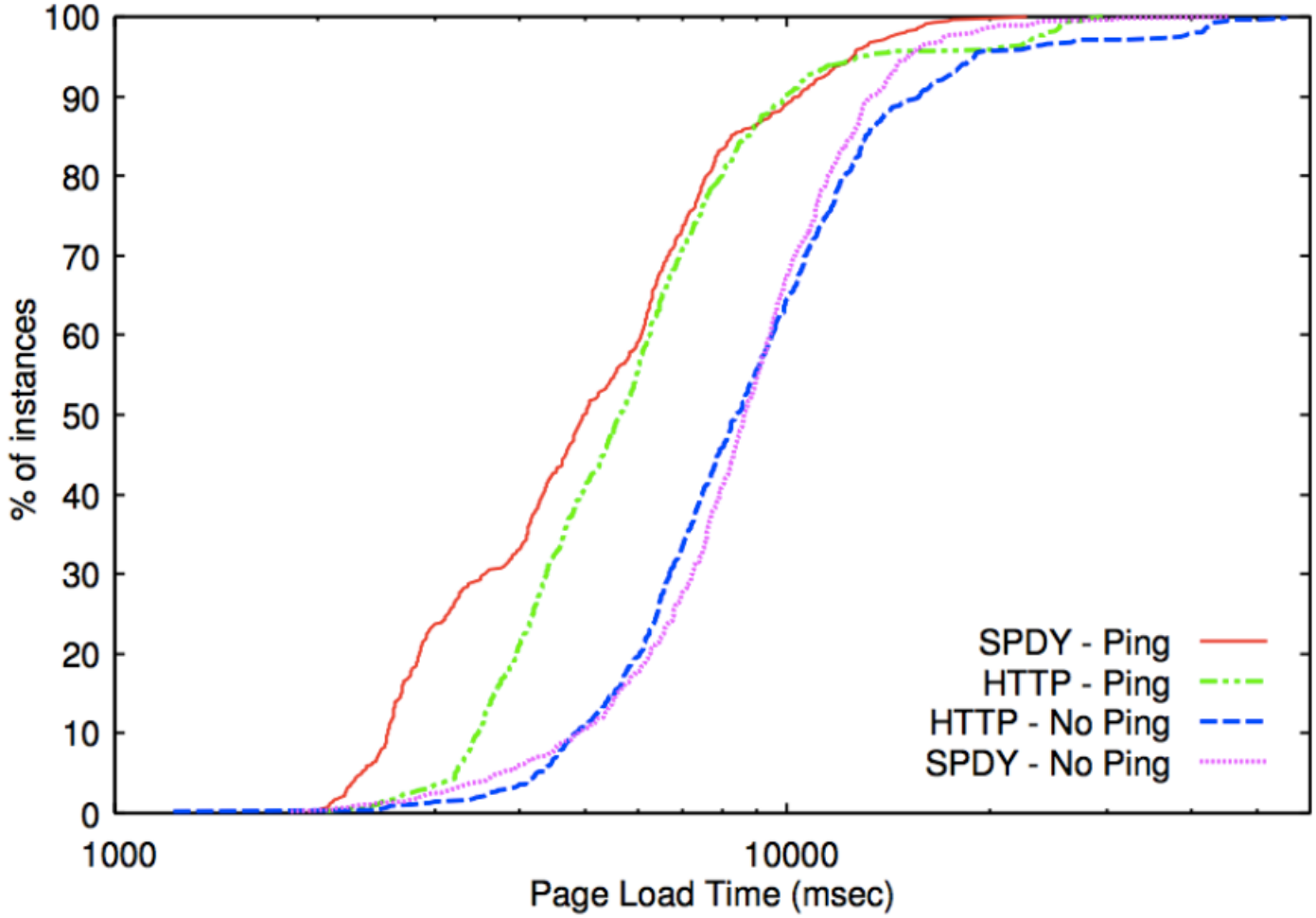


Figure 3: Page Load Time for different web sites with HTTP and SPDY.

Erman, Jeffrey, et al. "Towards a SPDY'ier mobile web?." Networking, IEEE/ACM Transactions on 23.6 (2015): 2010-2023.



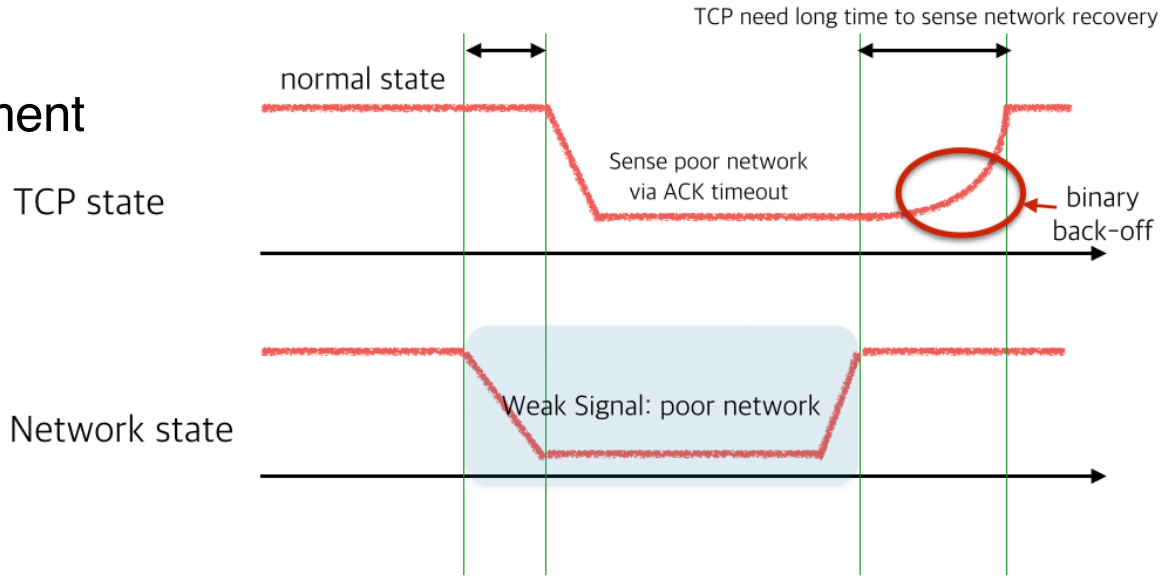
Erman, Jeffrey, et al. "Towards a SPDY'ier mobile web?." Networking, IEEE/ACM Transactions on 23.6 (2015): 2010-2023.

It’s The TCP, **STUPID!**

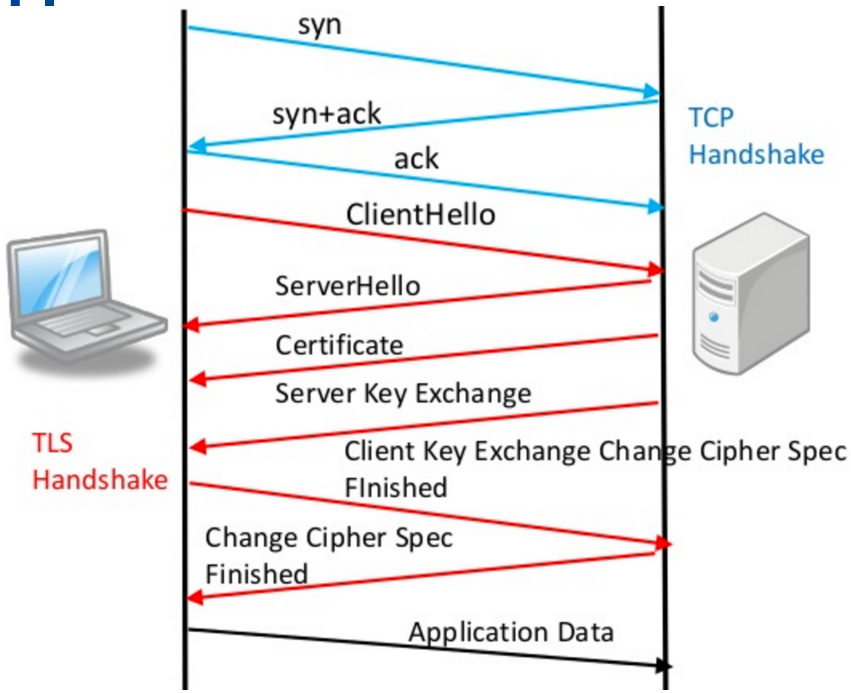
문제는 TCP

Issues in TCP+TLS

- Head-of-line blocking (HoL)
  - TCP retransmissions
- TCP 3-way handshake cost (~= 1.5 rtt)
- TLS negotiation handshake cost (~= 3 rtt)
- Large backoff on packet loss
- Poor performance in mobile environment
  - Wifi <-> Cellular transition
  - NAT, Firewall, …



TCP+TLS 4.5 RT Connection



1.5 RT

3 RT

http://www.slideshare.net/shigeki_ohtsu/quic-overview

TCP Slow Start

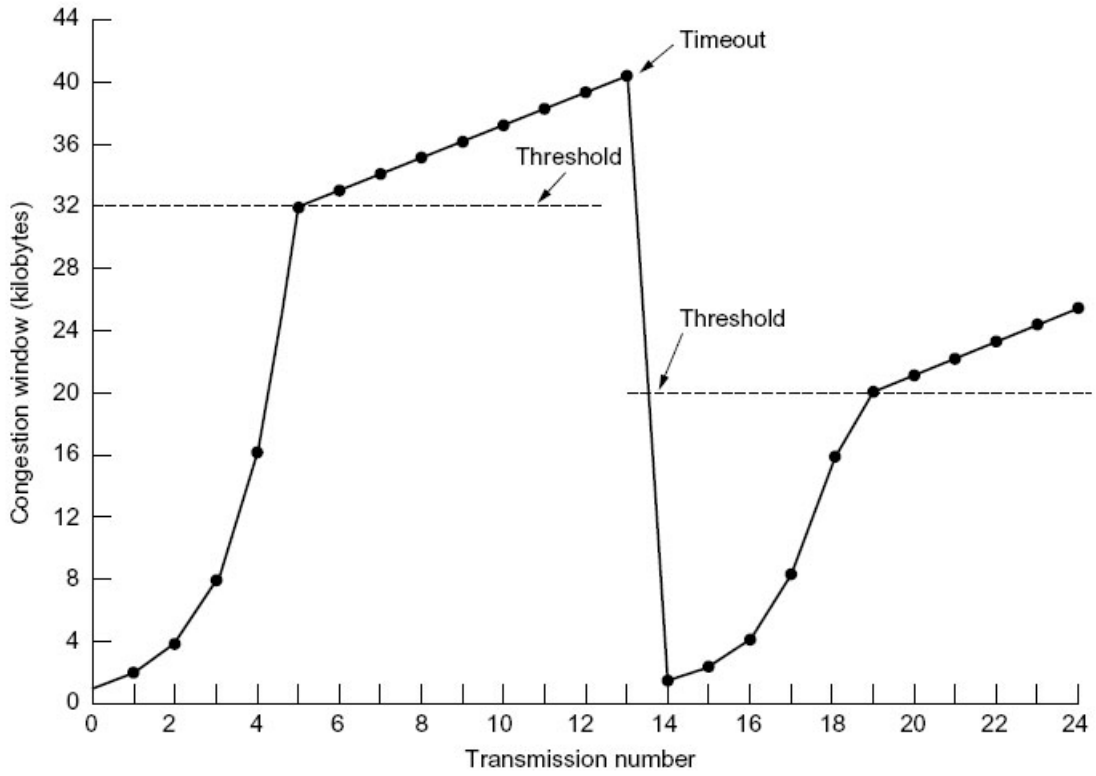


Fig. 6-37. An example of the Internet congestion algorithm.

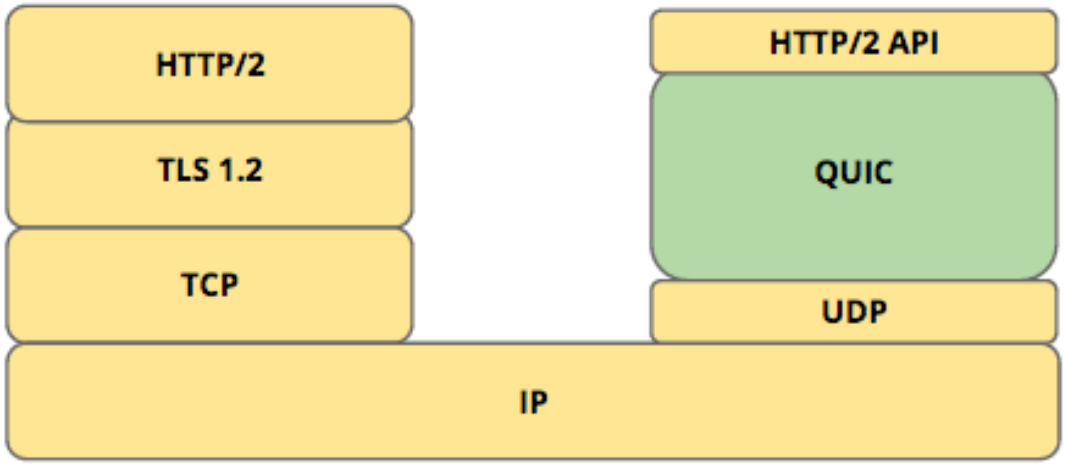
http://www.site.uottawa.ca/~bochmann/CourseModules/NetworkQoS/

“We should solve this without modifying middle-boxes and OS kernels”

What is QUIC?

Making the internet faster with...

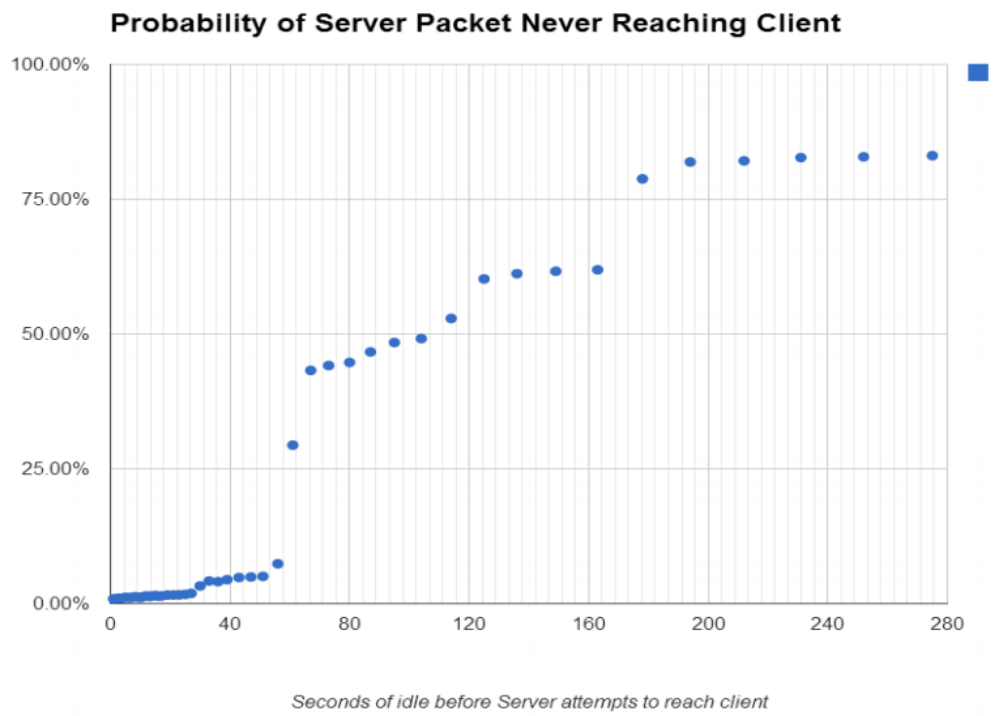
Quick
UDP
Internet
Connections



구글의 목표

- 오늘날의 인터넷 세상에 바로 적용 가능할 것
- 통신 과정의 Latency를 개선할 것
- Packet Loss로 인한 HOL 문제를 해결할 것
- TCP의 혼잡 제어 알고리즘을 개선할 것
- TLS 이상으로 안전할 것
- Mobile 환경에서 Cellular <=> Wifi 사이의 마이그레이션이 일어나도 잘 동작할 것

How much idle until unbinding?



91% ~ 94% of users can make outbound UDP

QUIC

- Head-of-line blocking (HoL) (TCP / HTTP)
- TCP 3-way handshake cost (~= 1.5 rtt)
- TLS negotiation handshake cost (~= 3 rtt)
- Large backoff on packet loss
- Poor performance in mobile environment

Multiple stream
Not ordered packet

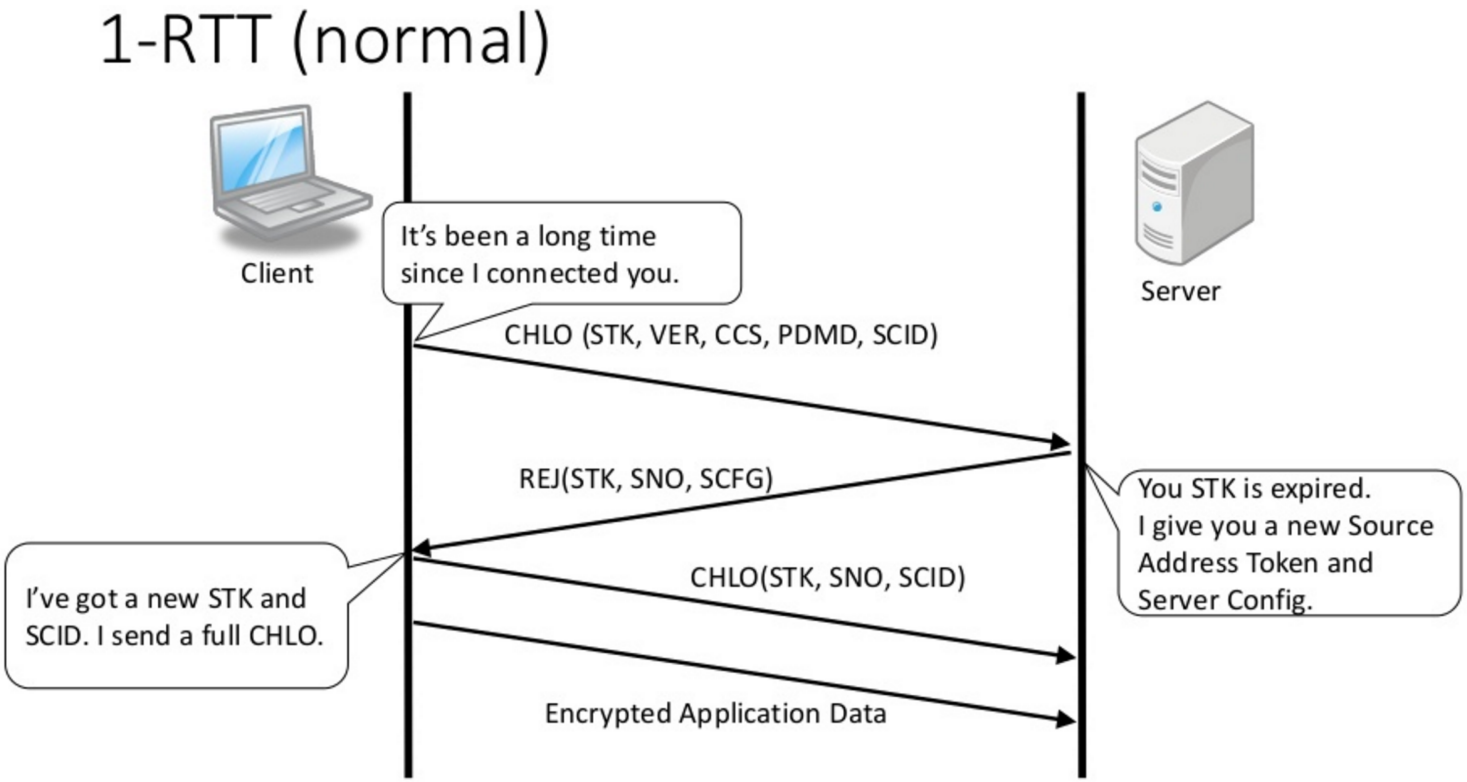
0RTT ~ 1RTT
connection establish

Migration support

Improved congestion control

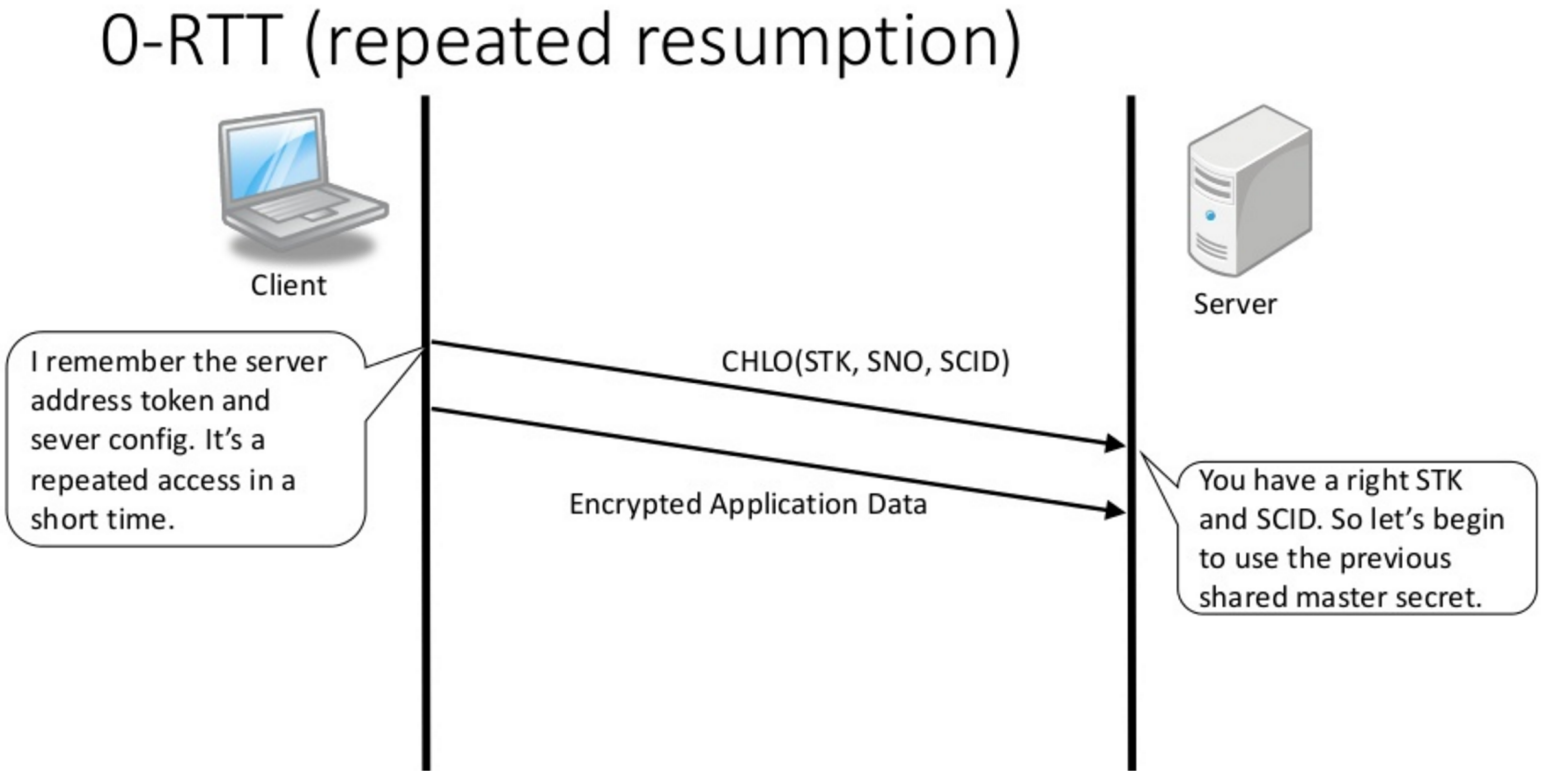
Static connection config

QUIC 1-RTT Connection



http://www.slideshare.net/shigeki_ohtsu/quic-overview

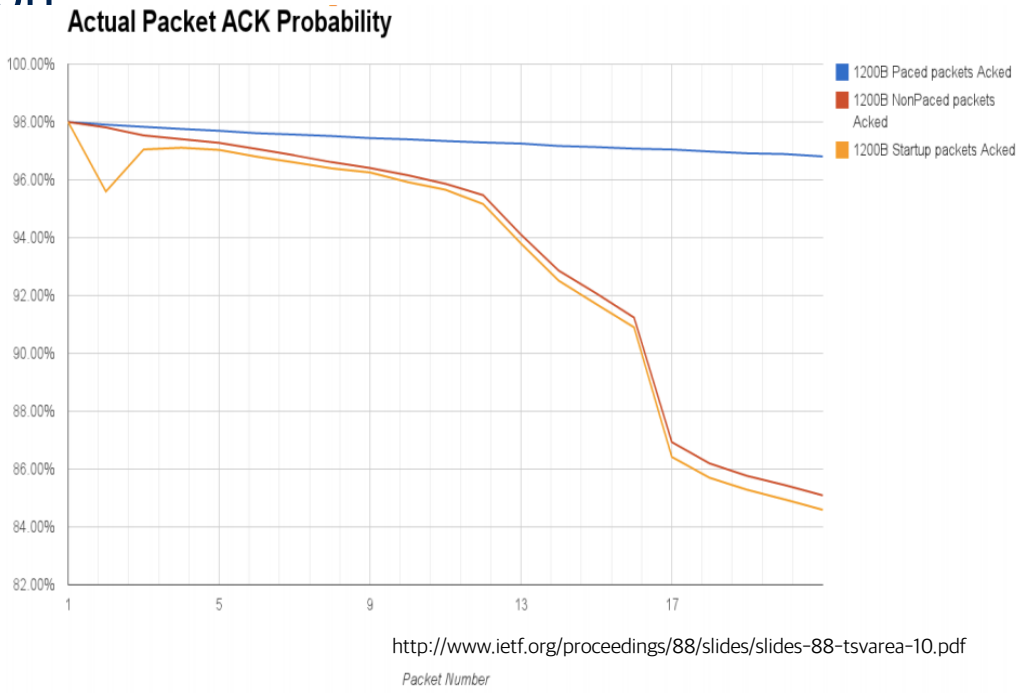
QUIC 0-RTT Connection



http://www.slideshare.net/shigeki_ohtsu/quic-overview

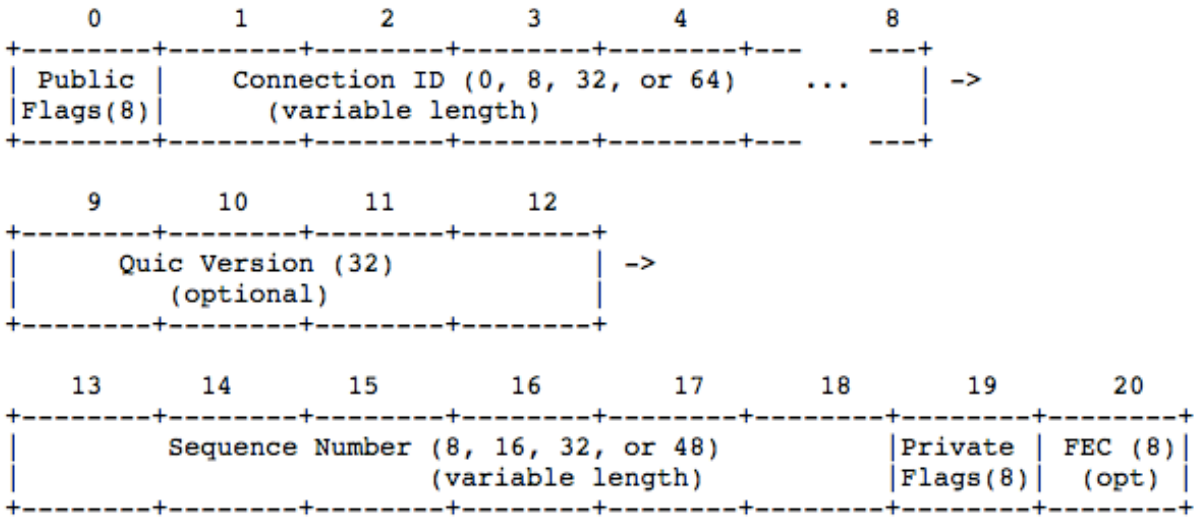
QUIC Congestion Control

- Pluggable Congestion Control Algorithm
  - Based on TCP cubic
  - Pacing based on bandwidth estimation
  - Equivalent to 2 TCP connection



http://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf

Connection Migration



실제 서비스에 적용하기

초기 목표

- 빠르게 프로토타이핑한 후 정말로 효과가 있는지 검증
- 한 가지 이상의 다른 언어와 잘 결합할 수 있는 형태로 구현 (Go? Python? Java? Rust?)
- 멀티코어를 완전히 활용해서 실제 Production에 사용 가능한 수준의 RPS로 최적화
- 현재 제공중인 서비스에 Transparent 하게 적용할 수 있는 형태여야 함
  - QUIC 적용을 위해 모든 인프라와 클라이언트를 뜯어 고쳐야 한다면 사용할 수 없다!
- 큰 공수 없이 유지/보수 가능해야 함
  - QUIC은 아직도 매우 활발히 변화가 일어나는 프로토콜 (1년에 메이저 버전 3~4개 릴리즈)
  - 메이저 릴리즈 될 때마다 새로 구현해야 하면 매우 곤란

Chromium Project

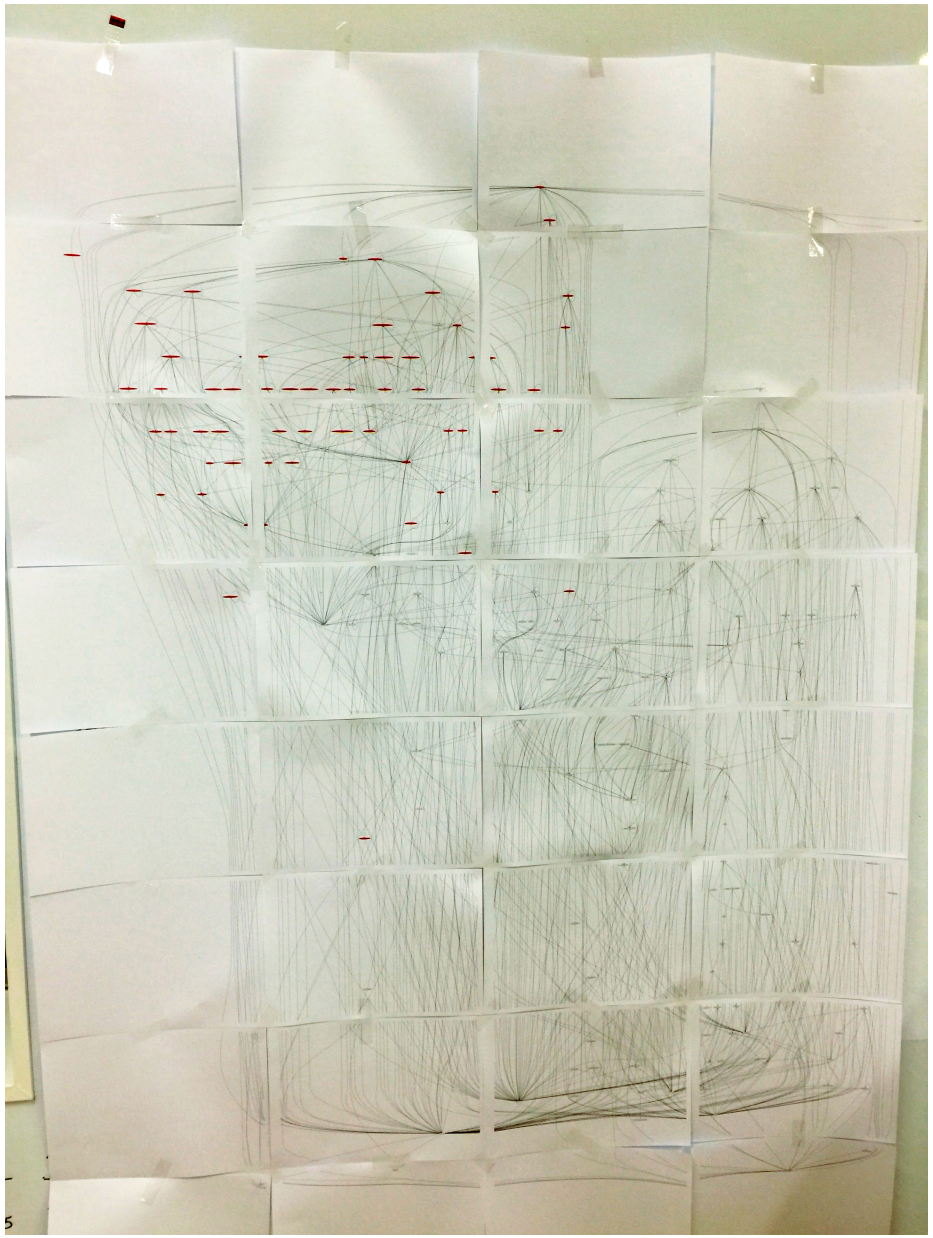
모든 것의 시작

https://www.chromium.org/quic

Chromium Project

- Toy Server / Toy Client / Chrome 브라우저를 이용해 동작 확인 가능
  - Not performant at scale
- 구글에서 사용하는 서버 구현체는 아직도 미공개
- Event Loop, Multi-threading, I/O 등등 구현 필요

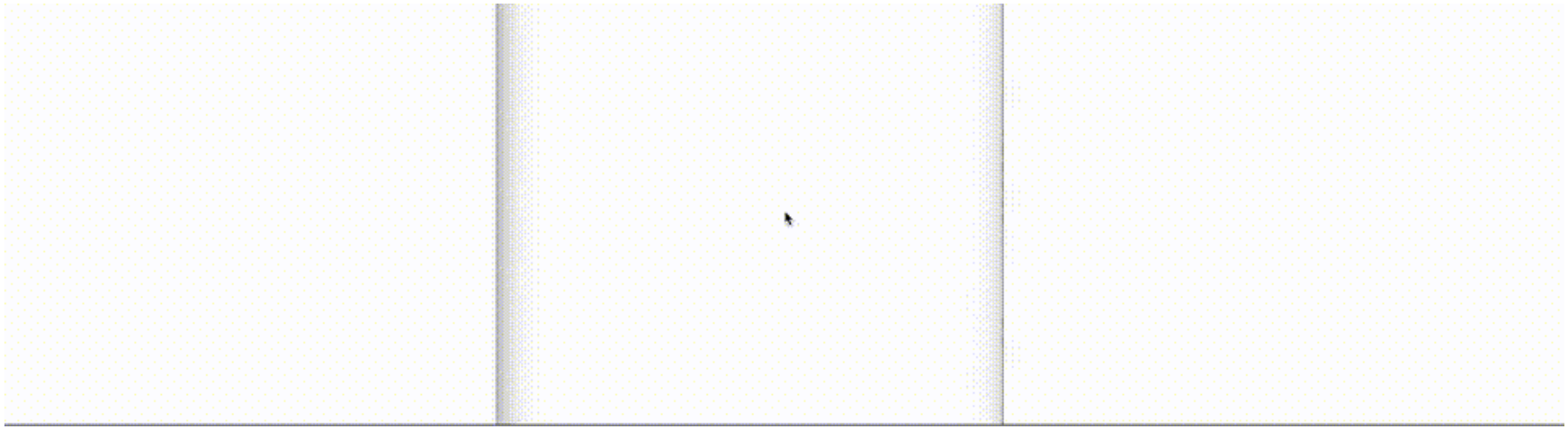
“Chromium 위에 직접 구현해서 붙여볼까...?”



Dependency Hell

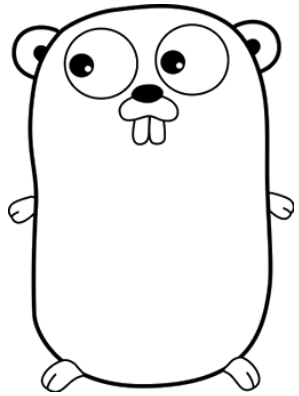
libquic / goquic

- **libquic**
  - Chromium codebase에서 주기적으로 sync
  - Python 스크립트와 자체 패치 시스템을 이용해 Dependency 최대한 제거
- **goquic**
  - Go 언어로 만든 libquic binding
  - 메인 로직의 경우 libquic의 C++ class를 cgo로 사용 => 유지보수 최소화
  - 멀티쓰레딩, GC, Non-blocking I/O 등 구현시 Golang의 장점을 살릴 수 있음



현재 알려진 서버 구현체

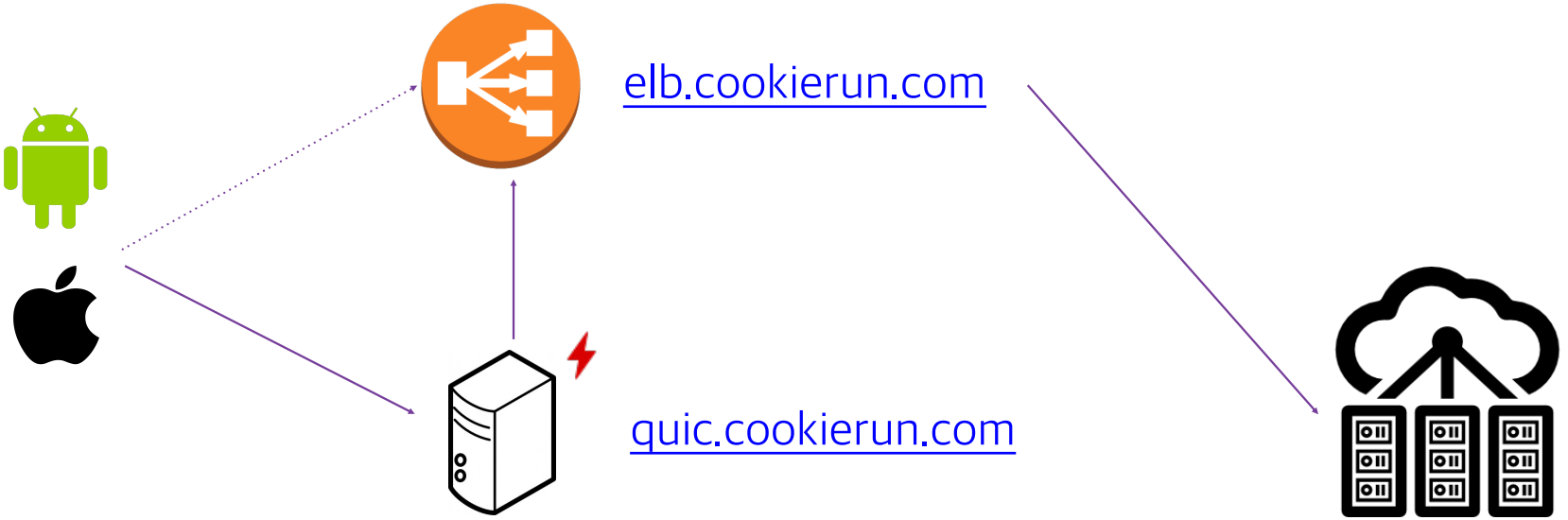
- **GoQuic**: Linux나 Mac, FreeBSD 환경에서 구동할 수 있는 Reverse Proxy
  - https://github.com/devsisters/goquic/
  - Docker Image로도 제공: http://devsisters.github.io/goquic/
- **quic-go**: Pure Go Implementation
  - https://github.com/lucas-clemente/quic-go
- **Caddy**: Web Server like Apache, nginx, … implemented in Golang
  - https://github.com/mholt/caddy
  - QUIC support based on quic-go



현재 알려진 서버 구현체

- 치트키: **Google App Engine**을 사용하면 자동으로 QUIC 통신이 가능함

서비스에 적용하기



서비스에 적용하기 - 서버

- Load Balancer에 UDP support가 되지 않는 경우 DNS RR 기반으로 부하 분산 (예: AWS)
  - 외부에서의 장애 감지 로직과 Failover Plan이 꼭 필요
- QUIC 서버의 경우 대부분 CPU가 병목일 만큼 연산량이 많은 편

서비스에 적용하기 - 클라이언트

- Android: Cronet (https://chromium.googlesource.com/chromium/src/+/master/components/cronet)
  - JNI를 통해 C++쪽 네트워크 스택을 이용할 수 있는 형태
  - ICS 이하 기기 지원 X, ARMv6 기기 지원 X
  - 파편화로 인해 특정 플랫폼에서 크래시 나는 경우가 종종 있음
    - 실제로 겪은 예: NEON 미지원 기기(Tegra 칩셋) 에서 100% 크래시
    - 디바이스 커버리지 위주로 꼼꼼한 테스트 필요
  - Crash Analytics
    - 기기/OS/칩셋 등 공통점 파악
    - Chrome 쪽 문제인 경우: crbug.com
    - 우리 쪽 문제인 경우: 대부분 race condition

서비스에 적용하기 - 클라이언트

- iOS: CrNet (https://chromium.googlesource.com/chromium/src.git/+/master/ios/crnet/)
  - iOS 기본 네트워크 스택을 교체하는 형태로 작동
  - Android에 비해 큰 문제는 없는 편
  - Obj-C ARC(automatic reference counting) 사용하지 않으므로 MRR로 변경 필수

- Android/iOS 공통으로, static library 빌드와 실제 app 빌드시 파라미터를 맞추는 작업이 매우 중
  - 특히 **-DNDEBUG** 플래그 => 수많은 원인 모를 크래시의 주범
  - no-RTTI가 Default => C++ 프로젝트에서 RTTI 사용시 수동으로 켜줘야 함

- https://omahaproxy.appspot.com/ 를 기준으로 stable chrome version에 맞추기

Alternate Protocol

▼ General

Request URL: https://app.snapchat.com/
Request Method: GET
Status Code: 404
Remote Address: 172.217.25.209:443

▼ Response Headers

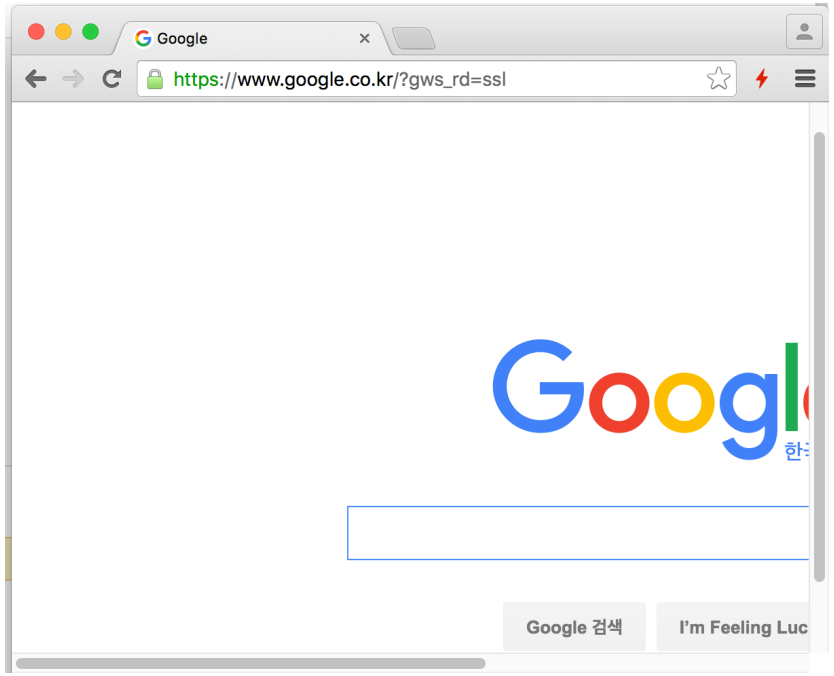
alt-svc: quic=":443"; ma=2592000; v="36,35,34,33,32,31,30,29,28,27,26,25"
alternate-protocol: 443:quic
cache-control: private
content-encoding: gzip
content-length: 164
content-type: text/html; charset=utf-8
date: Wed, 03 Aug 2016 06:14:50 GMT
server: Google Frontend
status: 404
vary: Accept-Encoding

“ 이 서버는 QUIC을 지원합니다!
만약 이 클라이언트가 QUIC 통신을 할 줄 안다면,
udp:443 포트로 오세요! ”

- 첫 통신의 경우 평범한 HTTP를 이용해 Negotiation하고, 그 다음 요청부터 QUIC으로 접속
- Cronet에선 CronetEngine.addQuicHint 을 통해 Hint를 추가해두면 QUIC부터 시도

Chrome HTTP/2 & SPDY Indicator

DEVIEW 2016



chrome://net-internals/#quic

nts (2781)

- QUIC Enabled: true
- Origins To Force QUIC On:
- Connection options:
- Consistent Port Selection Enabled: false
- Load Server Info Timeout Multiplier: 0.25
- Enable Connection Racing: false
- Disable Disk Cache: false
- Prefer AES: false
- Maximum Number Of Lossy Connections: 0
- Packet Loss Threshold: 1
- Delay TCP Race: true
- Store Server Configs In Properties File: null
- Idle Connection Timeout In Seconds: 30
- Disable PreConnect If 0RTT: false
- Disable QUIC On Timeout With Open Streams: false

QUIC sessions

View live QUIC sessions

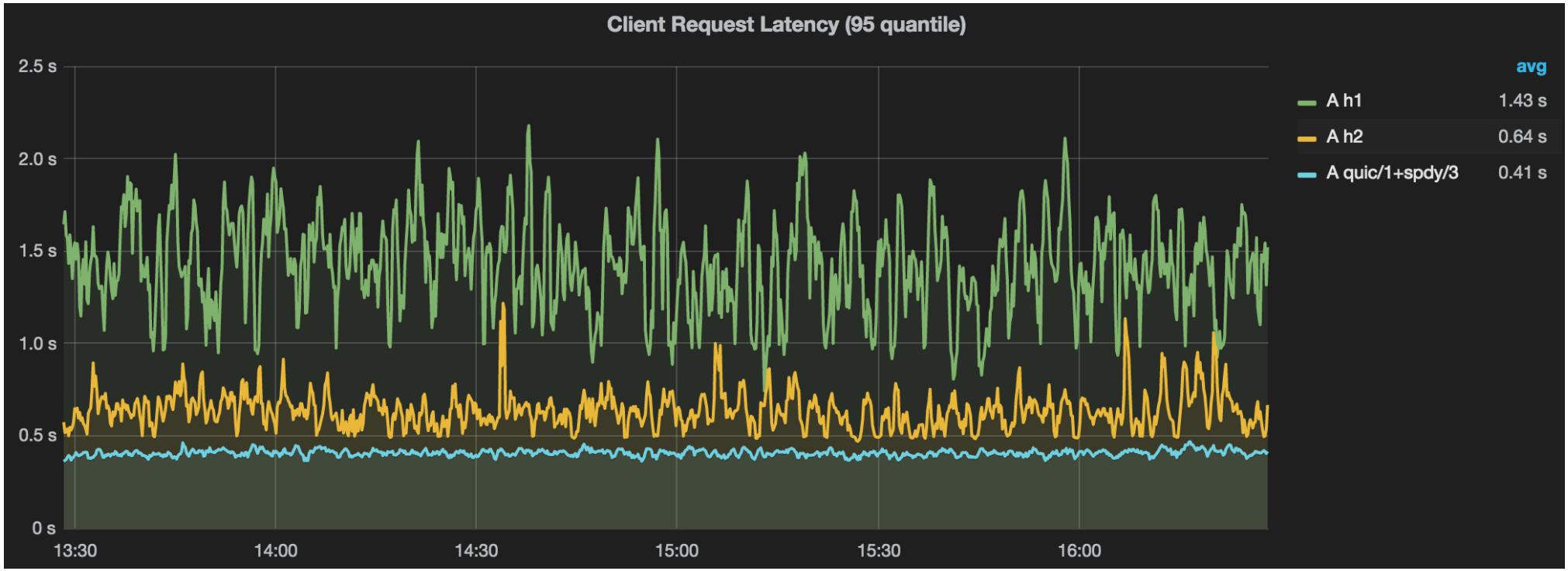
Host	Secure	Version	Peer address	Connection UID	Active stream count	Active streams	Total stream count	Packets Sent	Packets Lost	Packets Received	Connected
0.client-channel.google.com:443 1.client-channel.google.com:443 2.client-channel.google.com:443 26.client-channel.google.com:443 3.client-channel.google.com:443 4.client-channel.google.com:443	true	QUIC_VERSION_34	74.125.204.189:443	8448379936305602083	1	317	157	2762	0	3062	true
ad.doubleclick.net:443	true	QUIC_VERSION_34	216.58.197.198:443	2671149245733368514	0	None	2	6	0	5	true
apis.google.com:443 clients5.google.com:443 gg.google.com:443 google.com:443 maps.google.com:443 mt0.google.com:443 mt1.google.com:443 mt3.google.com:443 ogs.google.com:443 plus.google.com:443	true	QUIC_VERSION_34	216.58.200.206:443	3366539945084709897	0	None	12	52	0	80	true
clients4.google.com:443	true	QUIC_VERSION_34	216.58.200.206:443	16532108381080919402	0	None	1	11	0	9	true
fonts.googleapis.com:443	true	QUIC_VERSION_34	74.125.204.95:443	9337525089060514018	0	None	1	6	0	6	true
fonts.gstatic.com:443	true	QUIC_VERSION_34	216.58.200.195:443	16885793477731135313	0	None	0	3	0	3	true
lh3.ggpht.com:443	true	QUIC_VERSION_34	216.58.197.193:443	5811017809632471013	0	None	6	100	0	191	true
lh4.googleusercontent.com:443	true	QUIC_VERSION_34	216.58.200.193:443	18277597058782833292	0	None	23	417	0	803	true
mail.google.com:443	true	QUIC_VERSION_34	216.58.200.197:443	6030766412135104364	0	None	1	12	0	16	true
maps.googleapis.com:443	true	QUIC_VERSION_34	172.217.27.74:443	15295841871019338558	0	None	3	8	0	8	true
maps.gstatic.com:443 ssl.gstatic.com:443	true	QUIC_VERSION_34	216.58.200.195:443	12270305061463096716	0	None	4	13	0	17	true
stats.g.doubleclick.net:443	true	QUIC_VERSION_34	74.125.203.154:443	14232518293074091124	0	None	1	5	0	4	true
www.google-analytics.com:443	true	QUIC_VERSION_34	216.58.200.206:443	12633132038676602217	0	None	2	8	0	7	true
www.google.co.kr:443	true	QUIC_VERSION_34	216.58.197.195:443	7626362355839807651	0	None	28	74	0	80	true
www.google.com:443	true	QUIC_VERSION_34	216.58.200.196:443	10823391921666407226	0	None	16	628	0	1226	true
www.googleadservices.com:443	true	QUIC_VERSION_34	216.58.197.194:443	8283537004624215398	0	None	1	4	0	3	true
www.googletagmanager.com:443	true	QUIC_VERSION_34	216.58.197.200:443	11689457257936289207	0	None	1	10	0	15	true
www.gstatic.com:443	true	QUIC_VERSION_34	216.58.196.227:443	15718087903297706993	0	None	1	7	0	7	true

QUIC Status

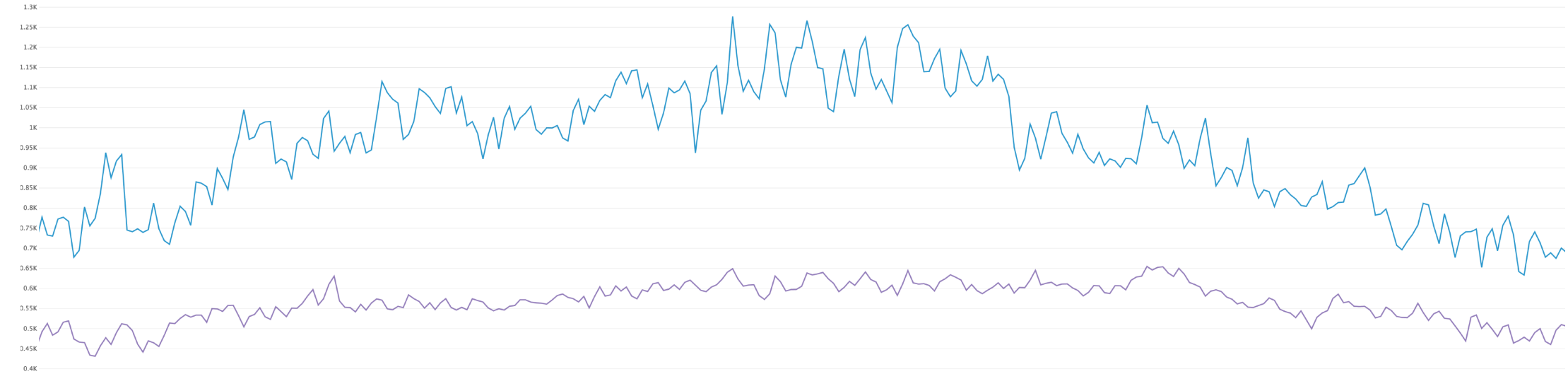


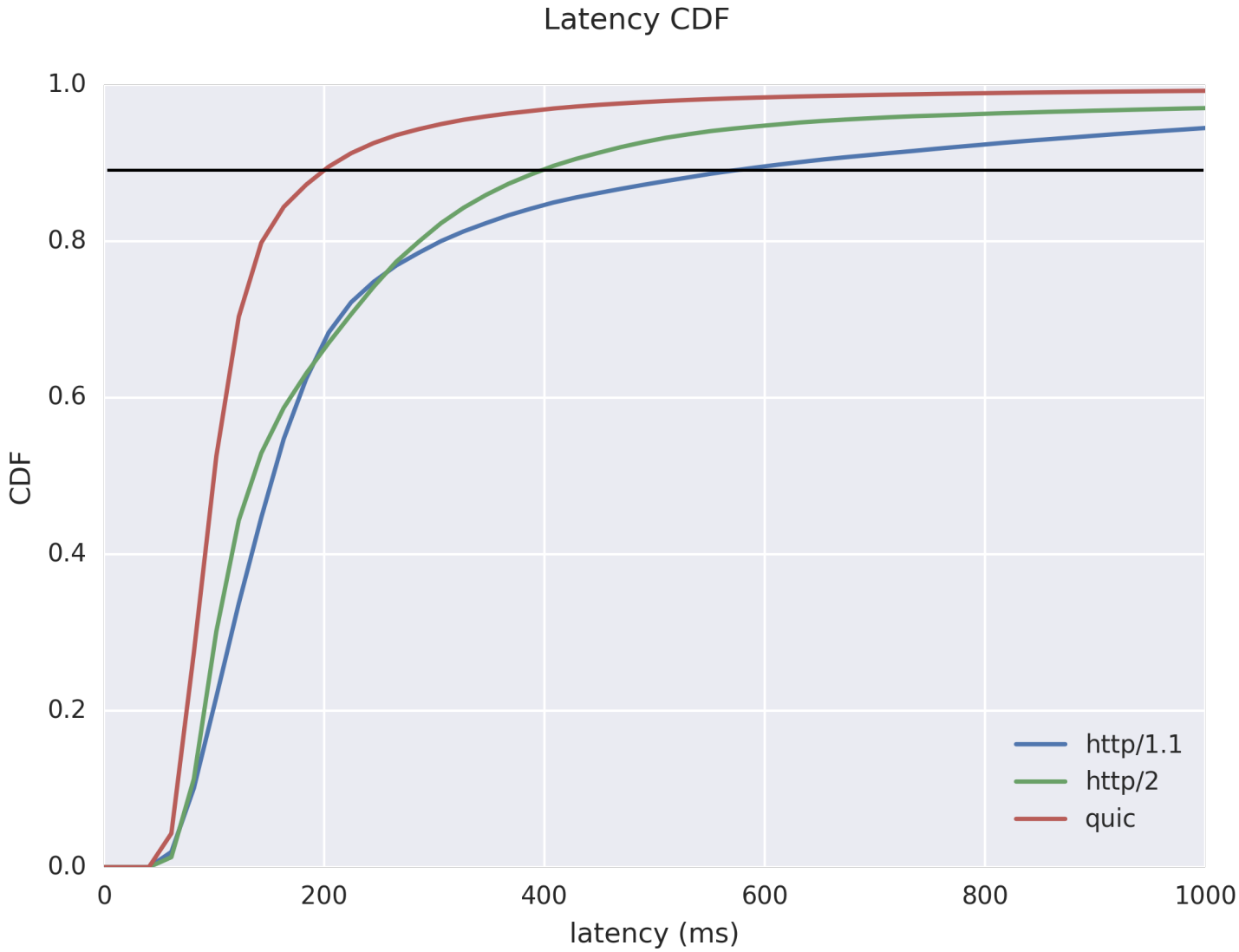
그리고 Akamai, Microsoft, LINE 등에서 연구중인 것으로 알려져 있음

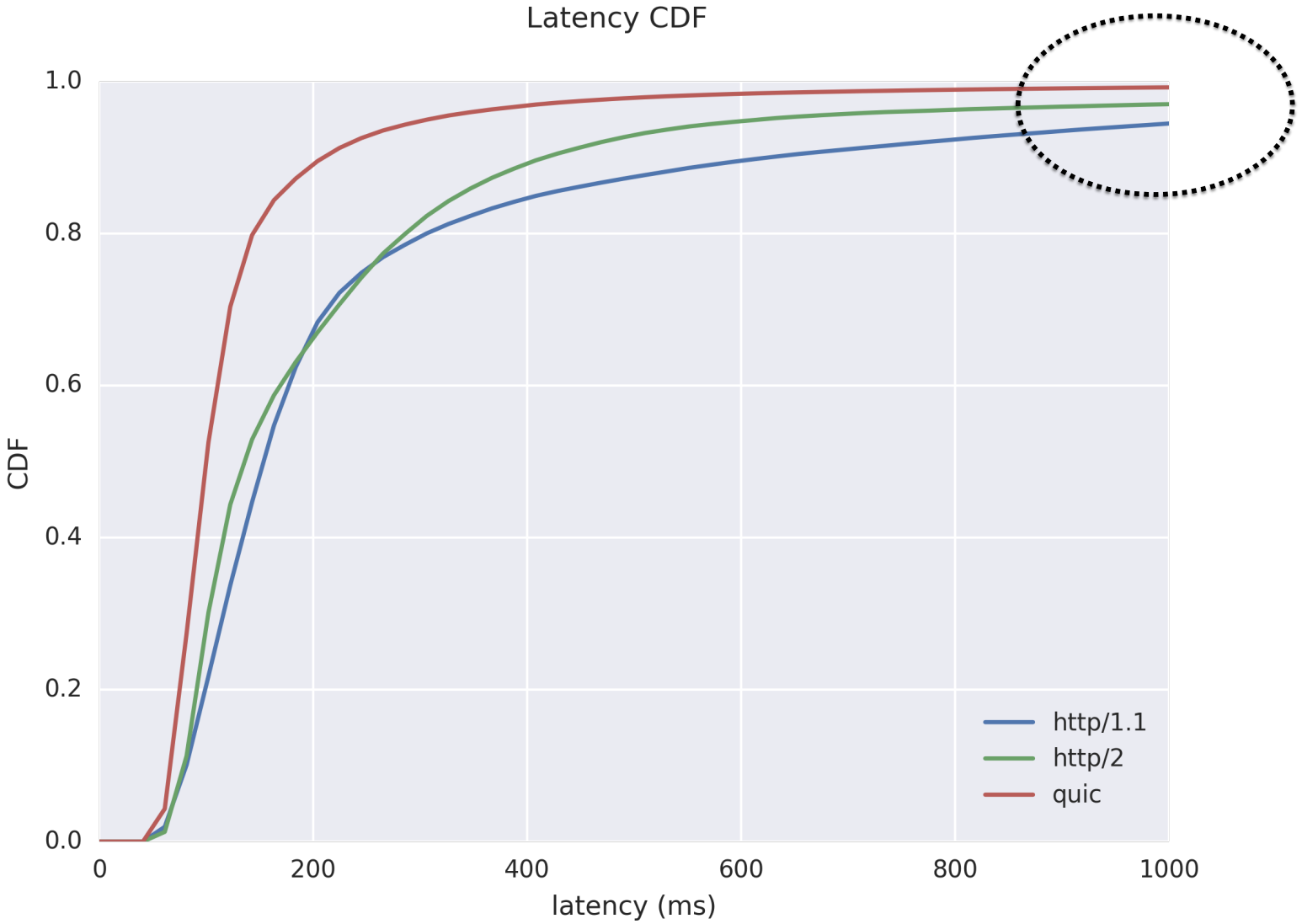
Results


Client Request Latency (95 quantile)
2.5 s
2.0 s
1.5 s
1.0 s
0.5 s
0 s
13:30
14:00
14:30
15:00
15:30
16:00
avg
A h1 1.43 s
A h2 0.64 s
A quic/1+spdy/3 0.41 s

DEVIEW
2016


1.3K
1.25K
1.2K
1.15K
1.1K
1.05K
1K
0.95K
0.9K
0.85K
0.8K
0.75K
0.7K
0.65K
0.6K
0.55K
0.5K
0.45K
0.4K


Latency CDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
latency (ms)
http/1.1
http/2
quic


Latency CDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
latency (ms)
http/1.1
http/2
quic

DEVIEW
2016

Q&A

Thank You

Acknowledgements

http://www.flaticon.com/free-icon
http://www.slideshare.net/shigeki_ohtsu/quic-overview
https://docs.google.com/presentation/d/1T9GtMz1CvPpZtmF8g-W7j9XHZBOCp9cu1fW0sMsmpoo/edit#slide=id.gb7cc33ba7_25_11
https://www.chromium.org/spdy/spdy-data
http://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf
https://golang.org/doc/gopher/

Erman, Jeffrey, et al. "Towards a SPDY'ier mobile web?." Networking, IEEE/ACM Transactions on 23.6 (2015): 2010-2023.