QUICスタート

2024/06/25

Tam

練習

- 手を上げてみましょう！

質問とかあったら使ってください。

アンケート

- HTTP って単語を聞いたことがある人挙手！

アンケート

- HTTP/3 って単語を聞いたことがある人挙手！

アンケート

- QUIC って単語を聞いたことがある人挙手！

今なぜ QUIC なのか？

- ブラウザと Web サーバーとの通信プロトコルである HTTP
- この HTTP の次期バージョンである HTTP/3 で QUIC が採用されることが決定している。

QUIC ≠ HTTP/3

- QUIC が使えれば、即 HTTP/3 が使えるわけではない。
- HTTP/3 のトランスポート層に QUIC が採用されただけ。
- HTTP/1, 2 でのトランスポート層は TCP および TLS などが相当する。

QUIC = TCP & TLS？

- 乱暴な言い方をすれば、その通りです。
- 新たに QUIC を採用するのは、 QUIC の特徴が重要だから。
- つまり QUIC の長所を理解するためには、 TCP & TLS の短所も理解しないといけない。

TCP とは？

- 接続型の双方向通信路。（乱暴な言い方をしています。）
- 簡単に言うと、

1. 片方からもう片方へ “Hello!” と文字列を送ると、
2. もう片方では “Hello!” とそのままの文字列が受け取れる。

- 接続はクライアント（サービス利用）側からサーバー（サービス提供）側への一方向。
- 接続が確立されれば、データは双方向でやり取りが可能。
- 複数の文字列を送っても、送った順番に受け取れる。

TCP で HTTP 通信を使う方法

```
% telnet www.google.com 80  < ------- telnet コマンドで Google サイトに接続
Trying 172.217.161.228...
Connected to www.google.com.
Escape character is '^]'.
GET /  < ----------------------------入力して ENTER
HTTP/1.0 200 OK
Date: Sun, 28 Feb 2021 07:04:15 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2021-02-28-07; expires=Tue, 30-Mar-2021 07:04:15 GMT; path=/; domain=.google.com; Secure
Set-Cookie: NID=210=WVVRF7ZfcovqExYo_5d0YW9cbESA8v9MhcxQ1CoO_enDhboMmApg0lD4qWWwvhePM_gDK
OOQxMlfZlqIpG4JR1egcP5-OG9zZ-B1eXc9N3Xv3PubRfo2EKWSTKkaTv9ZhUdxP2sLSnjxD8fFAB
IbsDfObJibCL-TDQZ1FxO2xsU; expires=Mon, 30-Aug-2021 07:04:15 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding

<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="ja">
<head><meta content="&#19990;&#30028;&#20013;&#12398;&#12354;
（以下 HTML が続きます。）
```

TLS とは？

- TCP は暗号化されていない。
- 無線LAN など、途中経路上でパケットを覗き見られると、通信内容が分かる。
- TLS は TCP 通信路上に暗号通信路を構築できる。

SSH ≠ TLS

- （分かっている人も多いとは思いますが）同じ暗号通信路が使える SSH と TLS は別物です。
- SSH は（主に） PC へのログインに使用するために開発されました。
  なので、使用するためには、サーバー側のアカウント情報が（一般的には）必要になります。
- TLS は汎用的な暗号通信路を提供します。

高校生にもわかる鍵交換の仕組み

- ディフィー・ヘルマン（DH と略します）鍵交換

1. g, p は適切に設定され、公開されているとする。
2. a, b はお互いに計算し、秘密にしておく。（相手にも渡さない。）
3. クライアント側： A = g^a mod p を計算し、サーバーに送る。
4. サーバー側： B = g^b mod p を計算し、クライアントに送る。
5. クライアント側： B^a = g^(b * a) mod p を計算して、秘密鍵とする。
6. サーバー側： A^b = g^(a * b) mod p を計算して、秘密鍵とする。

TLS で暗号通信が出来る仕組み

1. クライアント側からサーバー側へ ClientHELO が送られる。
2. サーバー側からクライアント側へ ServerHELO が送られる。
3. サーバー側からクライアント側へサーバーの鍵が送られる。
4. クライアント側からサーバー側へクライアントの鍵が送られる。
5. 双方でお互いの鍵情報で正しく復号できるか確認する。
6. 確認完了メッセージをお互いに送りあう。
7. 暗号通信開始。

ネットワークの性質

（唐突ですが）以下のような話を聞いたことはありませんか？

- ネットワークは相手に情報が本当に届いているかどうか分からない。
- もちろん反応も期待できない。
- 情報の順番も保証されない。
- 情報が化けることもある。

TCP の性質と違うことない？

TCP はスゴイ

- 送った情報すべてに番号を割り振る。
- 受け取った情報をチェックして、送った順番通りに並べ、順番に受け取る。
- 情報が正しいか、毎回 CRC チェックを行っている。
- 途中抜けている番号やエラーがあれば、再送を要求する。

3 handshake（スリーハンドシェイク）

- TCP は相手と通信を開始するまでに、手続きが必要。
- 手続きの概要：

1. クライアント側からサーバー側へ通信開始要求を行う。
2. サーバー側からクライアント側へ通信開始許可を行う。
3. クライアント側からサーバー側へ通信開始受諾を行う。

- これらの手続きを踏むことによって、接続の開始を確実にします。
- これを 3 handshake と言います。

TCP & TLS のデメリット

- 通信開始までに 3 handshake & 鍵交換と、何往復ものパケット交換が必要。
- パケット消失やデータ化けがあると、パケットの再送要求しか復元方法がない。
- （触れてないけど） OS のカーネルレベルで実装されているため、一般アプリでカスタマイズすることはほぼ不可能。

QUIC 登場！

- 基本は TCP と同じ 3handshake だが、暗号通信路も一緒に作ってしまう。

同じサーバーで条件があえば、 3 handshake すらも必要なし。

- エラー訂正符号を使用していて、少しのパケット消失なら、残りのパケットから復元可能。

エラー訂正符号を使用しているのは確実なのですが、どの程度の復元能力があるのかはまだ把握していません。

- UDP プロトコルとして通信するので、一般アプリからも（頑張れば）構築可能。

たくさんの数学的知識を使うので、ライブラリを使用するのでなければ、現実的ではありません。

QUIC のデメリット

- プロトコルは大変難しく、普通のプログラマが組めるものではなさそう。
- 通信品質が安定している環境下では TCP よりもはるかに遅いし、 CPU パワーも必要。
- 規格にまだ若干の揺れがあり、使うライブラリ同士によって通信できないことがある。

QUIC の注意点

- gQUIC（Google 製）と iQUIC（IETF勧告）には互換性がない。
- （最近の QUIC は）ALPN に対応している。
- 大きめの UDP が疎通出来ないといけない。

まとめ

QUIC を使おう！