Relátorio Final

QUIC (UDP, TLS 1, 2 E 3, QUIC) - HTTP/3

- Eduardo Costa Braga
- Henrique Franca Alves de Lima
- Isabela Medeiros Belo Lopes
- Júlia Vilela Cintra Galvão

HTTP:

O que é o protocolo HTTP

HTTP é um protocolo de transferência da camada de aplicação baseado em texto e é considerado a base para a comunicação de dados entre dispositivos em rede. Durante esse processo de requisição-resposta, o HTTP usa padrões e regras predefinidos para a troca de informações. Em geral, HTTP é o protocolo que clientes e servidores usam para se comunicar.

- URL:
  - É o primeiro passo no processo de troca de informações

História do HTTP

Hipertexto é um tipo de texto não linear criado por Ted Nelson em 1965, inspirado nas ideias de Vannevar Bush. Em 1989, Tim Berners-Lee propôs o projeto WorldWideWeb e, em 1991, desenvolveu com sua equipe o protocolo HTTP para recuperar documentos via links de hipertexto. Esses documentos passaram a usar HTML (HyperText Markup Language), uma linguagem textual que, além de texto, pode incorporar comandos e recursos como imagens, vídeos e áudios.

- ### Versões do HTTP

- HTTP/0.9 (1991)
  - Primeira versão, extremamente simples.
  - Apenas o método GET.
  - Sem cabeçalhos, códigos de status ou suporte a tipos de mídia além de texto
- HTTP/1.0 (1996)
  - Introdução de cabeçalhos de requisição/resposta.
  - Suporte a tipos de mídia, códigos de status e metadados.
  - Cada requisição exigia uma nova conexão.
- HTTP/1.1 (1999)
  - Amplamente adotado até hoje.
  - Suporte a conexões persistentes (keep-alive), cache, codificação em partes (chunked transfer), e pipelining.
  - Melhor controle de erros e criptografia via TLS (HTTPS).
- HTTP/2 (2015)
  - Baseado em multiplexação sobre uma única conexão TCP.
  - Redução de latência e aumento de desempenho.
  - Compactação de cabeçalhos e envio de dados binários.
  - Segurança reforçada (costuma ser usado com TLS/HTTPS).
  - Sem mudança na semântica: continua usando GET, POST, etc.
- HTTP/3 (2019)
  - Usa QUIC em vez de TCP, que roda sobre UDP.
  - Mais rápido na abertura de conexões e mais resiliente a perdas de pacotes.
  - QUIC implementa criptografia (TLS 1.3), retransmissão e controle de congestionamento diretamente.
  - Ideal para redes móveis e instáveis.
  - Compatível com HTML5, que amplia os recursos da web moderna (mas vale lembrar: HTML5 não é exclusivo do HTTP/3, ele funciona em qualquer versão).

HTTP/2 já é amplamente adotado, especialmente em sites grandes e aplicações modernas que priorizam desempenho. Porém, HTTP/1.1 ainda é o mais comum em números absolutos, especialmente em sites menores ou antigos. O HTTP/3 está em crescimento, mas ainda não é tão amplamente usado quanto o HTTP/2

O HTTP fica na sétima camada do modelo OSI, que é a camada de aplicação, onde tem a iteração entre usuários e computadores. A camada de aplicação é responsável pelos protocolos e manipulação de dados dos quais o software depende para apresentar dados ao usuário.

HTTP2

- HTTP/2 permite várias requisições e respostas simultâneas numa única conexão TCP.
- Cada requisição/resposta é dividida em "frames" e enviada em streams independentes, Isso evita a abertura de várias conexões simultâneas, como era necessário no HTTP/1.1.
- Compressão de cabeçalhos (HPACK): Os cabeçalhos HTTP (como cookies e user-agent) são muitas vezes repetidos. O HTTP/2 comprime esses dados, reduzindo o tráfego.
- Server Push: O servidor pode "adiantar" a entrega de arquivos que ele sabe que o navegador vai precisar (como CSS e JS), antes mesmo do cliente pedir.
- **Porque HTTP/2 está sendo substituído (principais defeitos)**
  - Head-of-line blocking: (HOL blocking) é um problema em redes de computadores onde um pacote no início de uma fila bloqueia outros pacotes atrás dele, mesmo que esses outros pacotes pudessem ser processados, o que limita o desempenho
  - Dependência do TCP: O TCP garante entrega confiável, mas é mais lento para iniciar conexões e exige múltiplas trocas de pacotes para negociação, Isso aumenta a latência, especialmente em redes móveis.
  - Implementação complexa: A multiplexação e compressão de cabeçalhos exigem controle mais rigoroso dos dados, o que torna a implementação e depuração mais difíceis.
    - A multiplexação é um processo que permite enviar múltiplos sinais ou fluxos de informação através de um único canal de comunicação, aumentando a eficiência da transmissão
  - Server Push pouco usado: Embora promissor, o recurso de server push é pouco usado na prática e, em muitos casos, prejudica o desempenho se mal configurado

---

HTTP3

A principal diferença em relação às versões anteriores é que ele não usa o TCP, mas sim o QUIC, um novo protocolo de transporte criado pelo Google. Ele apresenta multiplexação em tempo real, Assim como o HTTP/2, permite enviar várias requisições/respostas simultaneamente na mesma conexão. A grande vantagem é que, com QUIC, a perda de pacotes em uma stream não bloqueia as outras, evitando o famoso head-of-line blocking que ainda ocorria no HTTP/2 por causa do TCP. O HTTP3 usa criptografia embutida, obrigatoriamente criptografia TLS 1.3. Isso melhora a segurança e reduz o tempo de conexão (handshake mais rápido).

- Os protocolos usados no HTTP3 são:
  - Aplicação: HTTP3

    - Transporte: QUIC
    - Rede: UDP
    - Segurança: TLS 1.3 (integrado ao QUIC)
- Impedimentos para o crescimento do HTTP3: Ao substituir o TCP por QUIC, ele resolve problemas antigos de latência e congestionamento. No entanto, sua adoção ainda é gradual por causa de obstáculos técnicos na infraestrutura da internet atual. Apesar de suas vantagens em desempenho, o HTTP/3 ainda enfrenta obstáculos para ampla adoção. Sua implementação exige que servidores suportem o protocolo QUIC baseado em UDP, o que demanda mudanças técnicas e uso de bibliotecas específicas. Além disso, muitos equipamentos de rede, como firewalls e proxies, são otimizados para TCP e podem bloquear ou limitar conexões via UDP. Redes corporativas mais antigas frequentemente restringem o tráfego UDP por motivos de segurança, impedindo o uso do novo protocolo. Por fim, como o HTTP/2 já traz boas melhorias, muitos sites optam por adiar a migração devido ao custo e à complexidade técnica

---

Relembrando TCP e UDP

- TCP: O TCP (Transmission Control Protocol) é um protocolo de rede que garante a entrega confiável e ordenada de dados entre dois dispositivos. Ele utiliza um handshake de três vias para estabelecer a conexão e um handshake de quatro vias para fechá-la
- UDP: Ele é conforme seu nome sugere, um protocolo sem conexões baseado em datagrama. O que significa que não há handshakes e garantias de pedidos ou entrega. Isso quer dizer que quaisquer passos possíveis para garantir entrega, integridade de dados e outros fatores ficam a cargo da camada da aplicação. Assim, uma aplicação que seja formada sobre UDP pode escolher estratégias para implementar, dependendo do caso concreto, ou pode possivelmente impulsionar elementos como camadas de links como somas de verificação até evitar sobrecargas. A especificação do formato de pacote UDP é mínima, seu cabeçalho consiste da porta de origem, porta de destino, comprimento em bytes do cabeçalho do pacote e dados do pacote, além da soma de verificação. Ela pode ser usada para verificar a integridade de dados tanto para o cabeçalho quando para a parte de dados do pacote. A soma de verificação é opcional quando a camada de protocolo subjacente é o IPv4 e mandatória para o IPv6. Até agora, o UDP tem sido usado para coisas como a sincronização de relógio de sistemas de computador, aplicações VoIP, transmissão de vídeos, sistema DNS e protocolo DHCP

Diferença de SSL para TLS

O SSL (Secure Sockets Layer) foi o antecessor do TLS e teve como função original proteger a comunicação entre cliente e servidor, garantindo confidencialidade, integridade e autenticidade dos dados. Desenvolvido pela Netscape nos anos 90, ele teve três versões principais: SSL 1.0 (nunca lançada), SSL 2.0 (lançada em 1995, mas insegura) e SSL 3.0 (de 1996, mais estável, porém ainda vulnerável com o tempo). O TLS 1.0, lançado em 1999, é basicamente uma evolução direta do SSL 3.0, com mudanças internas e correções de segurança. Apesar de manter a ideia e até parte da estrutura do SSL 3.0, o nome foi alterado para marcar essa nova fase, supervisionada pela IETF (Internet Engineering Task Force).

- Diferenças entre SSL e TLS 1.0: • Segurança:
  - TLS 1.0 corrige várias vulnerabilidades existentes no SSL 3.0.
  - Criptografia: TLS usa algoritmos de criptografia mais modernos e seguros.
  - Handshake: O processo de negociação inicial no TLS é mais robusto e resistente a ataques.
  - Padronização: O TLS passou a ser um protocolo formalmente padronizado pela IETF, enquanto o SSL era mantido pela Netscape

---

TLS (Transport Layer Security)

É um protocolo criptográfico que protege a comunicação entre dois dispositivos em uma rede, como o navegador e um servidor web. Seu principal objetivo é garantir segurança, privacidade e integridade dos dados trocados pela internet. Ele funciona sobre protocolos como HTTP, SMTP ou FTP, criando um canal seguro por onde os dados trafegam criptografados, impedindo que sejam interceptados ou modificados por terceiros. De forma geral, o funcionamento do TLS se baseia em um processo chamado handshake, que ocorre logo no início da comunicação. Nele, o cliente (por exemplo, o navegador) e o servidor negociam quais algoritmos criptográficos usarão, trocam chaves públicas e estabelecem uma chave simétrica para criptografar a sessão. Esse processo garante que, mesmo que os dados sejam interceptados, eles não possam ser lidos ou alterados.

Ao longo dos anos, o TLS passou por diversas versões, cada uma aprimorando a segurança e a eficiência do protocolo:

- TLS 1.0 (1999) foi a primeira versão oficial do protocolo, criada como sucessora do SSL 3.0. Apesar de oferecer avanços importantes para a época, ela ainda tinha fragilidades e, com o tempo, passou a ser considerada insegura.
- TLS 1.1 (2006) veio para corrigir algumas dessas falhas, como a vulnerabilidade a ataques de injeção de IV, mas não representou uma mudança drástica no protocolo, e também acabou caindo em desuso.
- TLS 1.2 (2008) marcou uma evolução significativa, com suporte a algoritmos criptográficos mais robustos, como o SHA-256, e melhorias na flexibilidade e resistência

contra ataques. É a versão mais usada até hoje, mesmo com o crescimento da adoção do TLS 1.3.

- TLS 1.3 (2018) representa a versão mais moderna e segura. Ela removeu algoritmos antigos e inseguros, simplificou drasticamente o processo de handshake — reduzindo o número de mensagens trocadas entre cliente e servidor — e aumentou a privacidade. Essas mudanças resultaram em conexões mais rápidas, com menor latência e maior proteção contra espionagem e ataques de downgrade.

Handshake TLS

O handshake TLS (ou aperto de mãos TLS) é o processo inicial que acontece entre um cliente (como um navegador) e um servidor antes que qualquer dado real seja transmitido com segurança. O objetivo do handshake é estabelecer uma conexão segura, definindo algoritmos criptográficos e gerando chaves de sessão compartilhadas que serão usadas para criptografar os dados trocados

- Como funciona o handshake TLS (versão geral, como o TLS 1.2):
  1. Cliente --> Servidor (ClientHello)
     - O cliente envia uma mensagem com:
       - A versão do TLS suportada
       - Uma lista de suites criptográficas que pode usar
       - Um número aleatório (nonce)
       - Outras informações, como extensões TLS
  2. Servidor --> Cliente (ServerHello)
     - O servidor responde com:
       - A versão TLS escolhida
       - A suite criptográfica que será usada
       - Outro número aleatório
       - Seu certificado digital (com chave pública) para identificação
  3. Autenticação e troca de chaves:
     - O cliente:
       - Verifica o certificado do servidor (garantindo que é válido e confiável)
       - Gera um segredo pré-mestre (pre-master secret), criptografa com a chave pública do servidor e envia.
  4. Geração da chave de sessão:
     - Ambos os lados (cliente e servidor), com base nos números aleatórios trocados e no segredo pré-mestre, usam a mesma fórmula para gerar uma chave de sessão simétrica, que será usada para criptografar os dados da comunicação.
  5. Mensagem de finalização (Finished):
     - Cada lado envia uma mensagem "Finished" criptografada com a nova chave para indicar que o handshake foi concluído com sucesso.
     - A partir daí, a comunicação segura começa.

No TLS 1.3, o processo é mais simples e rápido:

- O número de etapas foi reduzido.
- O cliente já envia o segredo logo no primeiro pacote (junto com o ClientHello).
- Algoritmos antigos foram removidos.
- Isso reduz a latência, melhorando a performance e segurança

TLS 1.3

Publicado em 2018, foi uma reformulação profunda do protocolo. Vários algoritmos considerados inseguros, como RC4, SHA-1 e MD5, foram removidos. O processo de handshake foi simplificado, reduzindo a quantidade de mensagens trocadas entre cliente e servidor, o que diminuiu a latência. O TLS 1.3 também antecipou a criptografia das comunicações, protegendo inclusive os dados de negociação.
Uma das novidades mais marcantes é o suporte ao 0-RTT (zero roundtrip time), que permite reusar dados de conexões anteriores para acelerar novas conexões, embora com cautela quanto à segurança. resumo de cada um desses termos relacionados à segurança e criptografia:

- BEAST (Browser Exploit Against SSL/TLS): Foi um ataque revelado em 2011 que explorava uma vulnerabilidade no TLS 1.0 e SSL 3.0 ao usar blocos de cifra (CBC) para decifrar cookies de sessão e dados sensíveis interceptando e manipulando pacotes.
- SHA-256 (Secure Hash Algorithm 256 bits): Algoritmo de hash seguro da família SHA-2, amplamente utilizado para garantir integridade de dados. Gera um resumo fixo de 256 bits a partir de qualquer entrada, sendo muito mais resistente a colisões do que versões antigas como o SHA-1.
- AES (Advanced Encryption Standard): Um dos algoritmos de criptografia simétrica mais seguros e rápidos em uso hoje. Trabalha com blocos de 128 bits e chaves de 128, 192 ou 256 bits, sendo amplamente adotado por governos e sistemas de segurança.
- Lucky13: Ataque de criptoanálise contra implementações do TLS usando cifras em modo CBC. Ele explora pequenas diferenças de tempo no processamento de mensagens para recuperar informações criptografadas, como cookies e senhas.
- RC4 (Rivest Cipher 4): Algoritmo de cifra de fluxo muito usado no passado, mas hoje considerado inseguro devido a várias vulnerabilidades que permitem recuperar texto original a partir de dados criptografados.
- SHA-1 (Secure Hash Algorithm 1): Algoritmo de hash amplamente usado até meados dos anos 2000, mas que hoje é considerado obsoleto por estar vulnerável a colisões — situações em que duas entradas diferentes geram o mesmo resumo.
- MD5 (Message Digest 5): Algoritmo de hash de 128 bits muito rápido, porém altamente vulnerável a colisões e ataques de falsificação. Foi amplamente usado, mas atualmente não é recomendado para segurança.

0-RTT

recurso introduzido no TLS 1.3 que permite ao cliente enviar dados já na primeira mensagem (ClientHello), antes mesmo de completar totalmente o handshake com o servidor.

Normalmente, em uma conexão TLS tradicional, o cliente e o servidor precisam trocar algumas mensagens (idas e voltas, ou "round-trips") antes que qualquer dado seja transmitido com segurança. No TLS 1.2, isso pode exigir até duas rodadas completas. No TLS 1.3, esse processo foi reduzido, e com o 0-RTT ele pode ser praticamente imediato — sem nenhuma rodada adicional de ida e volta.

Para isso, o cliente precisa já ter se conectado ao servidor anteriormente. Nesse contato anterior, ele recebeu do servidor uma informação chamada "ticket de sessão", que contém os parâmetros criptográficos da sessão. Ao se reconectar, ele usa esse ticket para tentar reusar a mesma chave e enviar dados imediatamente.

- **Vantagens**
    - Reduz drasticamente a latência em conexões repetidas.
    - Ideal para aplicativos de tempo real, como mensagens instantâneas, streaming e requisições web rápidas.
- **Desvantagens e risco**
    - Por ser baseada em dados de uma sessão anterior, o 0-RTT não oferece garantia de confidencialidade direta contra replay attacks (repetição de mensagens por invasores). Isso significa que, se um atacante interceptar uma mensagem 0-RTT, ele pode tentar reenviá-la ao servidor, e este pode aceitá-la como válida.

TLS e HTTP

O HTTP, por padrão, é um protocolo não seguro — os dados trafegam em texto puro. Para garantir criptografia, autenticação e integridade, utiliza-se o TLS como uma camada adicional de segurança.

Quando o HTTP é usado com TLS, ele é chamado de HTTPS (HTTP Secure). Ou seja, HTTPS = HTTP + TLS.

QUIC

QUIC (Quick UDP Internet Connections) é um protocolo de transporte criado pelo Google e posteriormente padronizado pela IETF. Ele foi desenvolvido para substituir o TCP (usado em HTTP/1.x e HTTP/2) e melhorar o desempenho da web moderna.

Diferente do TCP, que é baseado em conexões confiáveis e ordenadas, o QUIC é construído sobre o UDP, um protocolo simples e rápido, e implementa na camada de transporte tudo o que o TCP faz — e mais.

O QUIC foi projetado para ser confiável (como o TCP) seguro (como o TLS) e rápido, com menor latência. Para isso, ele integra transporte + segurança + controle de congestionamento em um único protocolo. Com um foco em correntes UDP, QUIC alcança multiplexação sem ter que ficar sobre uma conexão TCP. QUIC desenvolve sua conexão em um nível mais alto que TCP. Novas correntes dentro de conexões QUIC não são forçadas a aguardar até que as outras sejam finalizadas. Conexões QUIC também se beneficiam ao eliminar a sobrecarga do handshake do TCP, o que reduz a latência. Embora o QUIC acabe com os recursos de confiabilidade do TCP, ele compensa com a camada UDP, oferecendo retransmissão de pacotes, pedidos e assim por diante.

Conexões QUIC, que mencionamos anteriormente, combinam TLS e handshakes de transporte. Uma vez estabelecidas, elas são identificadas por CIDs únicas (IDs de conexão). Esses IDs persistem ao longo das mudanças de IP e podem ajudar a garantir downloads ininterruptos em, por exemplo, uma mudança de 4G para WiFi. Isso é relevante, principalmente porque cada vez mais o tráfego da Internet é conduzido em dispositivos móveis

Principais Componentes

1. Baseado em UDP
    - Usa o protocolo UDP (porta 443), que é leve e rápido.
    - O QUIC cria seu próprio controle de conexão e confiabilidade em cima do UDP
2. Criptografia embutida com TLS 1.3
    - A segurança é integrada: toda conexão QUIC é automaticamente criptografada.
    - Diferente de TCP, onde o TLS é uma camada separada, aqui faz parte do protocolo
3. Multiplexação de streams
    - Permite que vários fluxos (streams) independentes sejam enviados simultaneamente pela mesma conexão.
    - Um erro em um stream não bloqueia os outros (diferente do TCP e HTTP/2, onde o head-of-line blocking pode travar todos os streams)
4. Handshake rápido com 0-RTT
    - Em conexões repetidas, nenhum round-trip (ida e volta) é necessário para iniciar a comunicação.
    - Isso acelera significativamente o tempo de carregamento de páginas.
5. Controle de congestionamento e perda
    - QUIC implementa controle de congestionamento inteligente, similar ao TCP Cubic ou BBR, mas com mais flexibilidade para evolução futura.
    - Detecta e retransmite apenas os pacotes perdidos
6. Migração de conexões

- Como QUIC identifica conexões por IDs e não pelo endereço IP/porta, ele permite trocar de rede (por exemplo, Wi-Fi para 4G) sem desconectar.

O HTTP/3 usa exclusivamente o QUIC como protocolo de transporte

- Em vez de TCP + TLS, agora temos UDP + QUIC (com TLS 1.3 embutido).
- Isso traz grandes melhorias na latência, resiliência e desempenho de carregamento de páginas web.

Por que o QUIC é importante?

1. Redução de latência: conexões estabelecidas em 0-RTT são muito mais rápidas.
2. Eliminação do head-of-line blocking: múltiplos streams independentes sem bloqueios.
3. Segurança nativa: só aceita conexões criptografadas com TLS 1.3.
4. Migração de conexões sem queda: ideal para dispositivos móveis que mudam de rede.
5. Mais controle por parte das aplicações: porque roda no espaço do usuário, não no kernel, facilitando atualizações.

Defeitos e desvantagens do QUIC

1. UDP bloqueado em algumas redes:
   - Muitas redes corporativas ou firewalls bloqueiam tráfego UDP por padrão, dificultando a adoção.
2. Maior complexidade para firewalls e middleboxes:
   - Como tudo é criptografado, dispositivos de inspeção não conseguem analisar ou gerenciar o tráfego com facilidade (ao contrário do TCP, que é mais "transparente").
3. Implementações mais pesadas:
   - Como o QUIC roda na camada de aplicação e exige criptografia embutida, ele pode consumir mais CPU em servidores.
4. Adaptação da infraestrutura:
   - Servidores, proxies, roteadores e ferramentas de monitoramento precisam ser atualizados para lidar com QUIC corretamente.