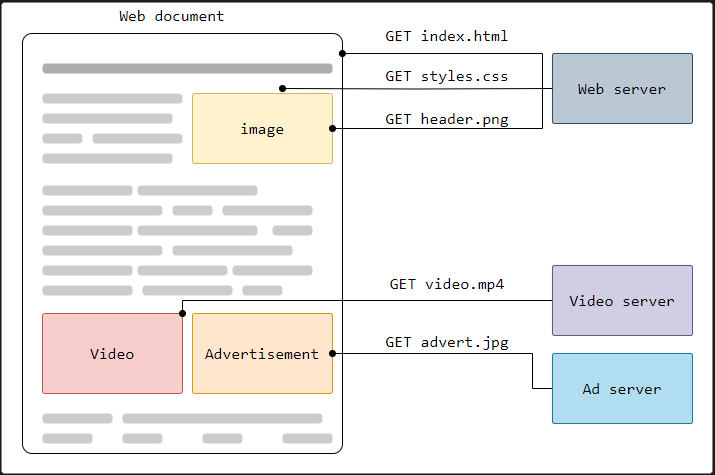
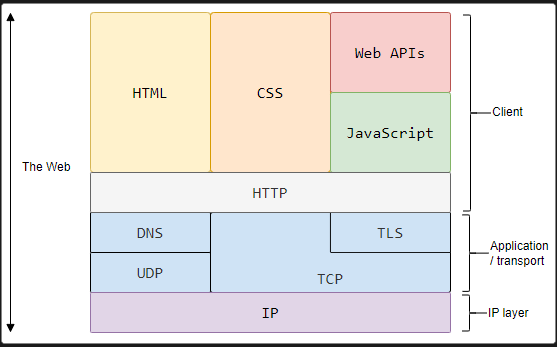
**An overview of HTTP**

HTTP é um protocolo para busca de recursos, como documentos HTML. É a base de qualquer troca de dados na web e é um protocolo cliente-servidor, o que significa que as solicitações são iniciadas pelo destinatário, geralmente o navegador da web. Um documento completo é normalmente construído a partir de recursos como conteúdo de texto, instruções de layout, imagens, vídeos, scripts e muito mais.



Clientes e servidores se comunicam por meio da troca de mensagens individuais (em oposição a um stream of data). As mensagens enviadas pelo cliente são chamadas de solicitações e as mensagens enviadas pelo servidor como resposta são chamadas de responses.

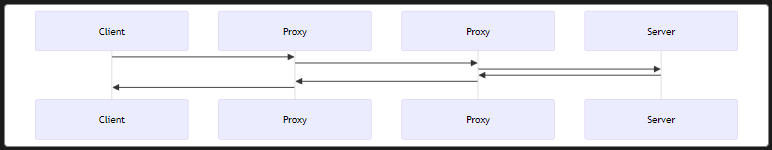


Projetado no início da década de 1990, o HTTP é um protocolo extensível que evoluiu ao longo do tempo. É um protocolo da camada de aplicação enviado via TCP ou por uma conexão TCP criptografada com TLS, embora qualquer protocolo de transporte confiável possa, teoricamente, ser usado. Devido à sua extensibilidade, é usado não apenas para buscar documentos de hipertexto, mas também imagens e vídeos, ou para postar conteúdo em servidores, como com resultados de formulários HTML. O HTTP também pode ser usado para buscar partes de documentos para atualizar páginas da web sob demanda.

**Components of HTTP-based systems**

HTTP é um protocolo cliente-servidor: as solicitações são enviadas por uma entidade, o agente do usuário (ou um proxy em seu nome). Na maioria das vezes, o agente do usuário é um navegador da web, mas pode ser qualquer coisa, por exemplo, um robô que rastreia a web para preencher e manter um índice de mecanismo de busca.

Cada solicitação individual é enviada a um servidor, que a processa e fornece uma resposta, chamada de resposta. Entre o cliente e o servidor, existem diversas entidades, coletivamente chamadas de proxies, que realizam diferentes operações e atuam como gateways ou caches, por exemplo.



Na realidade, existem mais computadores entre o navegador e o servidor que processa a solicitação: roteadores, modems e outros. Graças ao design em camadas da web, eles ficam ocultos nas camadas de rede e de transporte. O HTTP fica no topo, na camada de aplicação. Embora importantes para o diagnóstico de problemas de rede, as camadas subjacentes são, em sua maioria, irrelevantes para a descrição do HTTP.

Client: the user-agent

O user-agent é qualquer ferramenta que atua em nome do usuário. Essa função é desempenhada principalmente pelo navegador da web, mas também pode ser desempenhada por programas usados ​​por engenheiros e desenvolvedores da web para depurar seus aplicativos.

O navegador é sempre a entidade que inicia a solicitação. Nunca é o servidor (embora alguns mecanismos tenham sido adicionados ao longo dos anos para simular mensagens iniciadas pelo servidor).

Para exibir uma página da Web, o navegador envia uma solicitação original para buscar o documento HTML que representa a página. Em seguida, ele analisa esse arquivo, fazendo solicitações adicionais correspondentes a scripts de execução, informações de layout (CSS) a serem exibidas e sub-recursos contidos na página (geralmente imagens e vídeos). O navegador da Web então combina esses recursos para apresentar o documento completo, a página da Web. Os scripts executados pelo navegador podem buscar mais recursos em fases posteriores, e o navegador atualiza a página da Web de acordo.

Uma página da web é um documento de hipertexto. Isso significa que algumas partes do conteúdo exibido são links, que podem ser ativados (geralmente com um clique do mouse) para buscar uma nova página da web, permitindo ao usuário direcionar seu agente de usuário e navegar pela web. O navegador traduz essas instruções em solicitações HTTP e, posteriormente, interpreta as respostas HTTP para apresentar ao usuário uma resposta clara.

The Web server

No lado oposto do canal de comunicação está o servidor, que serve o documento conforme solicitado pelo cliente. Um servidor parece virtualmente uma única máquina; mas pode, na verdade, ser um conjunto de servidores compartilhando a carga (balanceamento de carga) ou outro software (como caches, um servidor de banco de dados ou servidores de e-commerce), gerando total ou parcialmente o documento sob demanda.

Um servidor não é necessariamente uma única máquina, mas várias instâncias de software de servidor podem ser hospedadas na mesma máquina. Com HTTP/1.1 e o cabeçalho Host, elas podem até compartilhar o mesmo endereço IP.

Proxies

Entre o navegador da Web e o servidor, vários computadores e máquinas retransmitem as mensagens HTTP. Devido à estrutura em camadas da pilha da Web, a maioria delas opera nos níveis de transporte, rede ou físico, tornando-se transparentes na camada HTTP e potencialmente tendo um impacto significativo no desempenho. Aqueles que operam nas camadas de aplicação são geralmente chamados de proxies. Eles podem ser transparentes, encaminhando as solicitações recebidas sem alterá-las de forma alguma, ou não transparentes, caso em que alteram a solicitação de alguma forma antes de repassá-la ao servidor. Os proxies podem desempenhar inúmeras funções:

* cache (o cache pode ser público ou privado, como o cache do navegador)
* filtragem (como uma verificação antivírus ou controle parental)
* load balancing (para permitir que vários servidores atendam a solicitações diferentes)
* authentication (para controlar o acesso a diferentes recursos)
* logging (registro) (permitindo o armazenamento de informações históricas)

**Basic aspects of HTTP**

HTTP is simple

O HTTP é geralmente projetado para ser legível por humanos, mesmo com a complexidade adicional introduzida no HTTP/2 pelo encapsulamento de mensagens HTTP em quadros. As mensagens HTTP podem ser lidas e compreendidas por humanos, proporcionando testes mais fáceis para desenvolvedores e complexidade reduzida para iniciantes.

HTTP is extensible

Introduzidos no HTTP/1.0, os cabeçalhos HTTP facilitam a extensão e a experimentação deste protocolo. Novas funcionalidades podem até ser introduzidas por meio de um acordo entre um cliente e um servidor sobre a semântica de um novo cabeçalho.

HTTP is stateless, but not sessionless (sem sessão)

O HTTP é sem estado: não há vínculo entre duas solicitações realizadas sucessivamente na mesma conexão. Isso imediatamente pode ser problemático para usuários que tentam interagir com determinadas páginas de forma coerente, por exemplo, usando carrinhos de compras de e-commerce. Mas, embora o núcleo do HTTP seja sem estado, os cookies HTTP permitem o uso de sessões com estado. Usando a extensibilidade de cabeçalhos, os cookies HTTP são adicionados ao fluxo de trabalho, permitindo que a criação de sessões em cada solicitação HTTP compartilhe o mesmo contexto ou o mesmo estado.

HTTP and connections

Uma conexão é controlada na camada de transporte e, portanto, fundamentalmente fora do escopo do HTTP. O HTTP não exige que o protocolo de transporte subjacente seja baseado em conexão; exige apenas que seja confiável ou que não perca mensagens (no mínimo, apresentando um erro nesses casos). Entre os dois protocolos de transporte mais comuns na internet, o TCP é confiável e o UDP, não. Portanto, o HTTP depende do padrão TCP, que é baseado em conexão.

Antes que um cliente e um servidor possam trocar um par de solicitação/resposta HTTP, eles precisam estabelecer uma conexão TCP, um processo que requer várias viagens de ida e volta. O comportamento padrão do HTTP/1.0 é abrir uma conexão TCP separada para cada par de solicitação/resposta HTTP. Isso é menos eficiente do que compartilhar uma única conexão TCP quando várias solicitações são enviadas em sucessão próxima.

Para mitigar essa falha, o HTTP/1.1 introduziu o pipelining (que se mostrou difícil de implementar) e conexões persistentes: a conexão TCP subjacente pode ser parcialmente controlada usando o cabeçalho Connection. O HTTP/2 foi um passo além, multiplexando mensagens em uma única conexão, ajudando a manter a conexão aquecida e mais eficiente.

Experimentos estão em andamento para projetar um protocolo de transporte mais adequado ao HTTP. Por exemplo, o Google está experimentando o QUIC, que se baseia no UDP para fornecer um protocolo de transporte mais confiável e eficiente.

O que pode ser controlado pelo HTTP

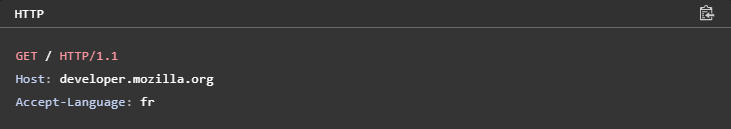
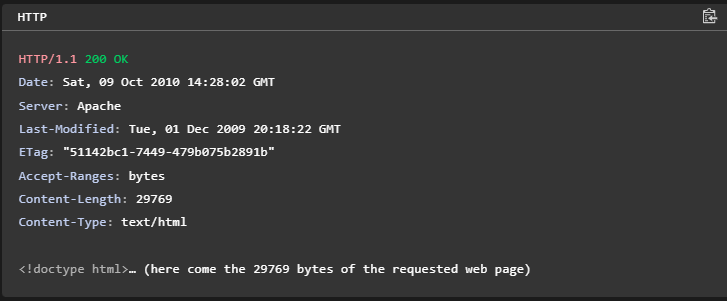
Essa natureza extensível do HTTP permitiu, ao longo do tempo, maior controle e funcionalidade da Web. Cache e métodos de autenticação eram funções gerenciadas no início da história do HTTP. A capacidade de flexibilizar a  origin constraint, por outro lado, só foi adicionada na década de 2010.

Aqui está uma lista de recursos comuns controláveis ​​com HTTP:

* Cache: A forma como os documentos são armazenados em cache pode ser controlada por HTTP. O servidor pode instruir proxies e clientes sobre o que armazenar em cache e por quanto tempo. O cliente pode instruir proxies de cache intermediários a ignorar o documento armazenado.
* Relaxing the origin constraint: Para evitar espionagem e outras invasões de privacidade, os navegadores da web impõem uma separação estrita entre sites. Somente páginas da mesma origem podem acessar todas as informações de uma página da web. Embora essa restrição seja um fardo para o servidor, os cabeçalhos HTTP podem relaxar essa separação estrita no lado do servidor, permitindo que um documento se torne uma colcha de retalhos de informações provenientes de diferentes domínios; pode até haver razões relacionadas à segurança para isso.
* Authentication: Algumas páginas podem ser protegidas para que apenas usuários específicos possam acessá-las. A autenticação básica pode ser fornecida por HTTP, usando o WWW-Authenticate e cabeçalhos semelhantes, ou definindo uma sessão específica usando cookies HTTP.
* Proxy e tunelamento: Servidores ou clientes geralmente estão localizados em intranets e ocultam seu verdadeiro endereço IP de outros computadores. As solicitações HTTP passam por proxies para cruzar essa barreira de rede. Nem todos os proxies são proxies HTTP. O protocolo SOCKS, por exemplo, opera em um nível inferior. Outros protocolos, como FTP, podem ser gerenciados por esses proxies.
* Sessões: O uso de cookies HTTP permite vincular solicitações ao estado do servidor. Isso cria sessões, apesar do HTTP básico ser um protocolo sem estado. Isso é útil não apenas para carrinhos de compras de e-commerce, mas também para qualquer site que permita a configuração da saída pelo usuário.

HTTP flow

Quando um cliente deseja se comunicar com um servidor, seja o servidor final ou um proxy intermediário, ele executa as seguintes etapas:

1. Abrir uma conexão TCP: A conexão TCP é usada para enviar uma ou várias solicitações e receber uma resposta. O cliente pode abrir uma nova conexão, reutilizar uma conexão existente ou abrir várias conexões TCP com os servidores.
2. Enviar uma mensagem HTTP: mensagens HTTP (antes do HTTP/2) são legíveis por humanos. Com o HTTP/2, essas mensagens são encapsuladas em quadros, tornando-as impossíveis de ler diretamente, mas o princípio permanece o mesmo. Por exemplo: 
3. Leia a resposta enviada pelo servidor, como: 
4. Feche ou reutilize a conexão para solicitações futuras

Se o pipeline HTTP estiver ativado, várias requisições podem ser enviadas sem esperar que a primeira resposta seja totalmente recebida. O pipeline HTTP tem se mostrado difícil de implementar em redes existentes, onde softwares antigos coexistem com versões modernas. O pipeline HTTP foi substituído no HTTP/2 por multiplexing requests mais robustas dentro de um quadro.

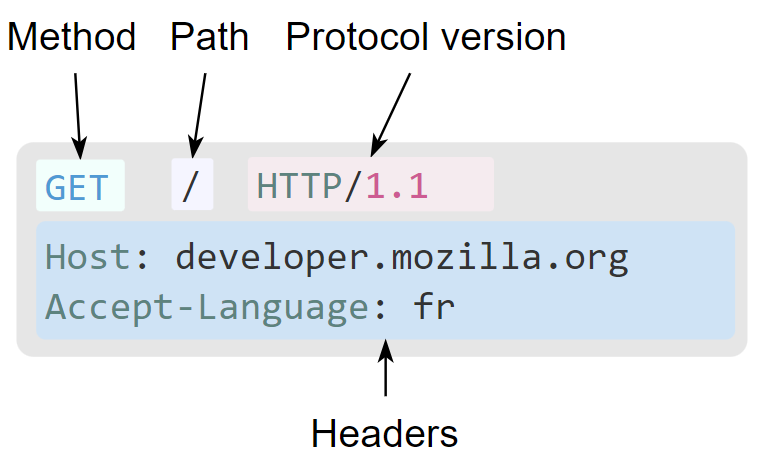
**HTTP Messages**

Mensagens HTTP, conforme definidas no HTTP/1.1 e anteriores, são legíveis por humanos. No HTTP/2, essas mensagens são incorporadas em uma estrutura binária, um quadro, permitindo otimizações como compactação de cabeçalhos e multiplexação. Mesmo que apenas parte da mensagem HTTP original seja enviada nesta versão do HTTP, a semântica de cada mensagem permanece inalterada e o cliente reconstitui (virtualmente) a solicitação HTTP/1.1 original. Portanto, é útil compreender mensagens HTTP/2 no formato HTTP/1.1.

Existem dois tipos de mensagens HTTP: solicitações e respostas, cada uma com seu próprio formato.

Requests

Um exemplo de solicitação HTTP:

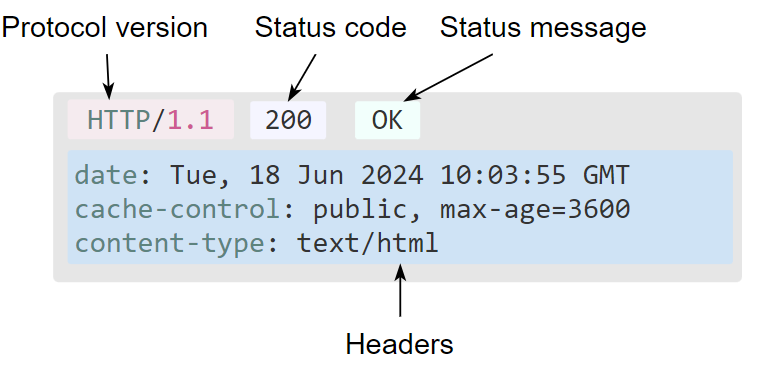


Requests consistem nos seguintes elementos:

* Um método HTTP, geralmente um verbo como GET, POST ou um substantivo como OPTIONS ou HEAD, que define a operação que o cliente deseja realizar. Normalmente, um cliente deseja buscar um recurso (usando GET) ou publicar o valor de um formulário HTML (usando POST), embora mais operações possam ser necessárias em outros casos.
* Path, o caminho do recurso a ser buscado; a URL do recurso sem elementos óbvios do contexto, por exemplo, sem o protocolo (http://), o domínio (aqui, developer.mozilla.org) ou a porta TCP (aqui, 80).
* A versão do protocolo HTTP.
* Cabeçalhos opcionais que transmitem informações adicionais para os servidores.
* Body, um corpo, para alguns métodos como POST, semelhante aos das respostas, que contém o recurso enviado.

Responses

Um exemplo de resposta:



Responses consistem nos seguintes elementos:

* A versão do protocolo HTTP que eles seguem.
* Um código de status, indicando se a solicitação foi bem-sucedida ou não e o motivo.
* Uma mensagem de status, uma breve descrição não autoritativa do código de status.
* Cabeçalhos HTTP, como os de solicitações.
* Opcionalmente, um corpo contendo o recurso recuperado.

APIs based on HTTP

A API baseada em HTTP mais comumente usada é a Fetch API, que pode ser usada para fazer requisições HTTP a partir de JavaScript. A Fetch API substitui a XMLHttpRequest API.

Outra API, eventos enviados pelo servidor, é um serviço unidirecional que permite que um servidor envie eventos ao cliente, usando HTTP como mecanismo de transporte. Usando a interface EventSource, o cliente abre uma conexão e estabelece manipuladores de eventos. O navegador do cliente converte automaticamente as mensagens que chegam no fluxo HTTP em objetos Event apropriados. Em seguida, ele as entrega aos manipuladores de eventos registrados para o tipo do evento, se conhecido, ou ao manipulador de eventos onmessage, se nenhum manipulador de eventos específico do tipo tiver sido estabelecido.

Conclusão

O HTTP é um protocolo extensível e fácil de usar. A estrutura cliente-servidor, combinada com a capacidade de adicionar cabeçalhos, permite que o HTTP avance junto com os recursos estendidos da Web. Embora o HTTP/2 adicione alguma complexidade ao incorporar mensagens HTTP em quadros para melhorar o desempenho, a estrutura básica das mensagens permanece a mesma desde o HTTP/1.0. O fluxo da sessão permanece básico, permitindo que seja investigado e depurado com um monitor de rede HTTP.