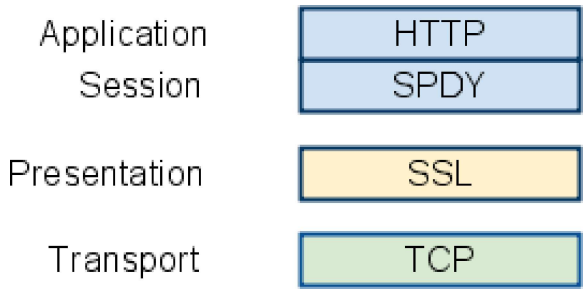
HTTP/2

SPDY（HTTP/2主要特性）

- SPDY为会话层协议，HTTP为应用层协议



HTTP	HTTP+SPDY
一个HTTP请求=一个TCP连接	多个HTTP请求=一个TCP连接，多个并发流
最多同时建立6个连接	并发流无上限
请求/响应头部未压缩	请求/相应头部压缩
部分字段（User-Agent/Host/Accept）重复发送	删除以避免重复发送
客户端主动	两端接可主动（Server Push）
无	强制使用SSL协议

- SPDY基本概念

  i. 一个SPDY的connection = 一个TCP连接

  ii. stream：双向字节流=请求+响应/推送+回复（每个stream有个单独的ID，用来区分不同的请求）

  iii. frame，以帧为基本单位发送数据

    - 控制帧

```
+----------------------------------+
|C| Version(15bits) | Type(16bits) |
+----------------------------------+
| Flags (8)  |  Length (24 bits)   |
+----------------------------------+
|               Data               |
+----------------------------------+
```

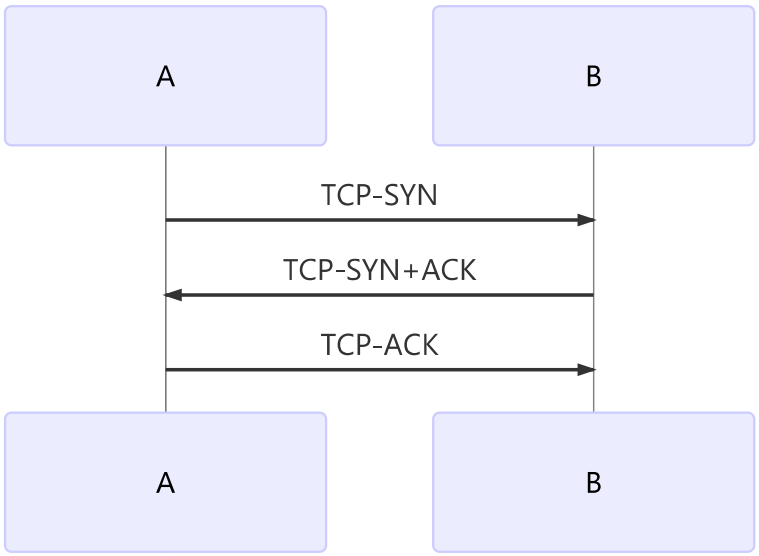
    - C = 1

        - Version = 协议版本号（当前为1）
        - Type = 控制帧的类型[SYN_STREAM-1, SYN_REPLY-2, FIN_STREAM=3, HELLO-4, ...]
        - Flags = 标识（不太清楚，只知道0x01 = FLAG_FIN，表示半关闭）
        - Length = Data的长度
        - Data = 控制帧的内容，不同Type的控制帧格式不同
    - 数据帧

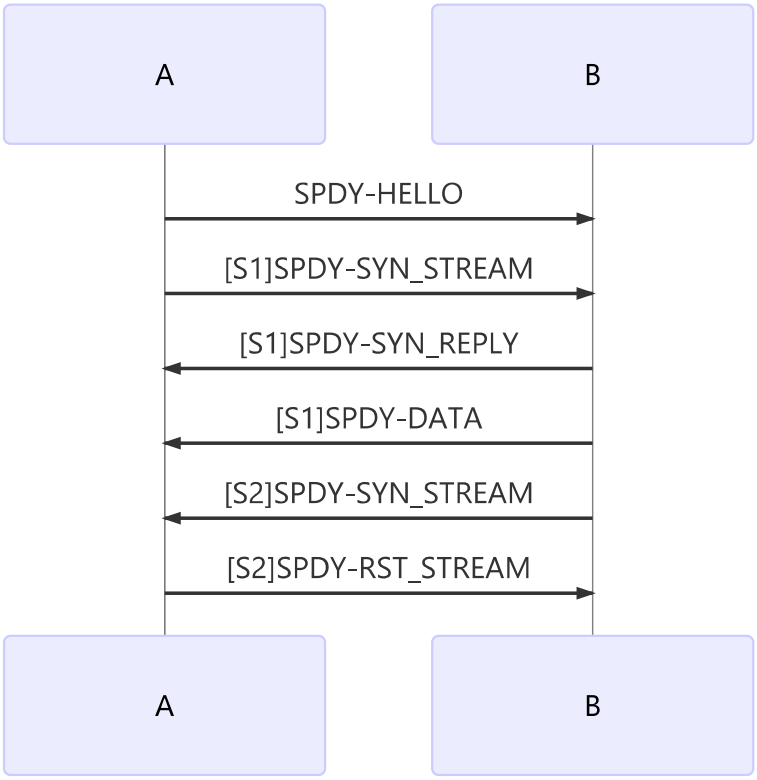
```
+----------------------------------+
|C|       Stream-ID (31bits)       |
+----------------------------------+
| Flags (8)  |  Length (24 bits)   |
+----------------------------------+
|               Data               |
+----------------------------------+
```

        - C = 0
        - Flags = 标识（不太清楚）
        - Length = Data的长度

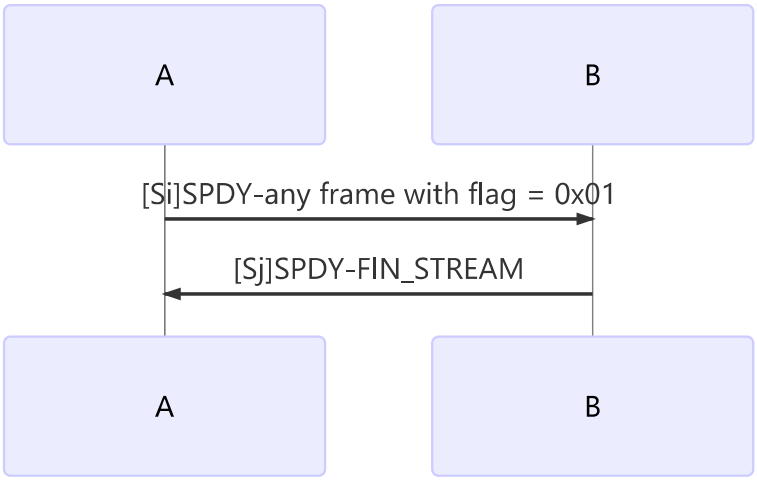
- SPDY会话过程
    - 建立TCP连接



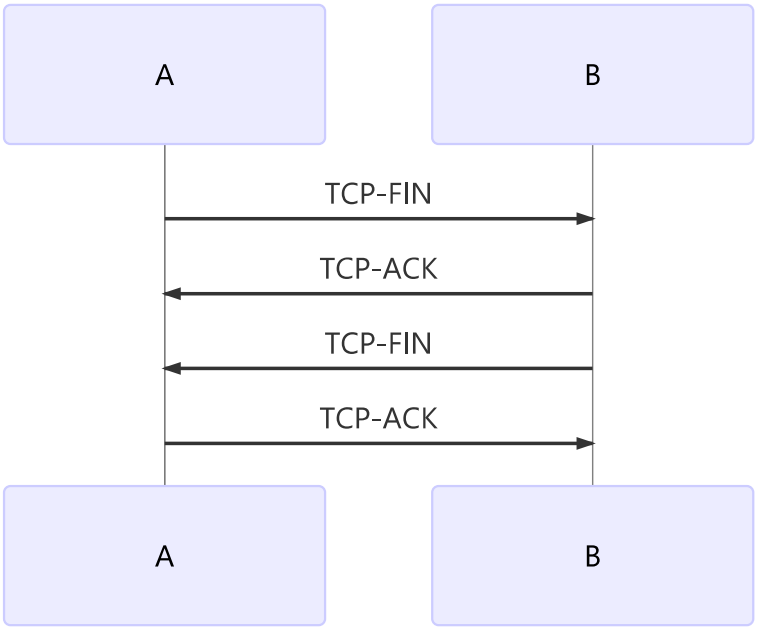
    - 流



- 结束流（以下两种形式任选其一，双端接可发起）



- 结束会话

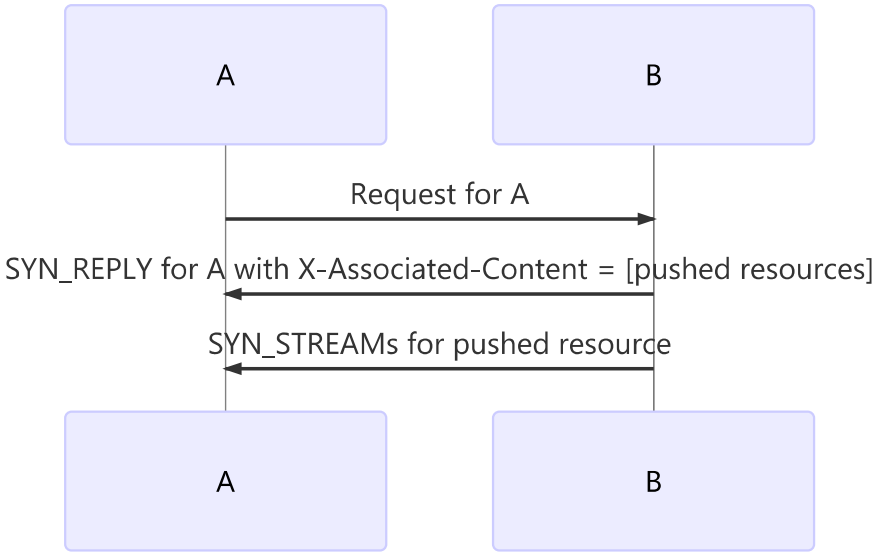


- SPDY主要特性
  - 多路复用
  - 请求优先级区分设定
  - HTTP头部压缩
  - Server Push
  - Server Hint
- Server Push

> 页面有一些很重要的资源需要尽快加载，例如核心的css文件。浏览器加载它们的延迟来自多方面，这几个是HTTP/2能解决的：1）同域名并发连接数限制造成的阻塞时间；3）浏览器从HTML 中找出外链资源这段时间；3）浏览器发起请求到服务端收到请求这段时间。

> Server Push 则会在客户端请求页面 HTML 时，新建流将最重要的资源一并返回。同时，如果服务端要推送的资源浏览器已经缓存过，客户端会发送 RST_STREAM 帧来终止流，服务端收到这个信号之前所传输的数据就造成了带宽浪费。这个问题可以通过在服务端记录给每个客户端发送过何种资源，何时过期来优化。

  - Server Push的特性：
    - 由服务器端在已知客户端会需要某些资源的情况下，主动提前发起stream将资源发送至客户端
    - 只支持HTTP GET请求
    - 客户端可返回RST_STREAM进行阻止
    - nginx不支持，Apache和node.js支持

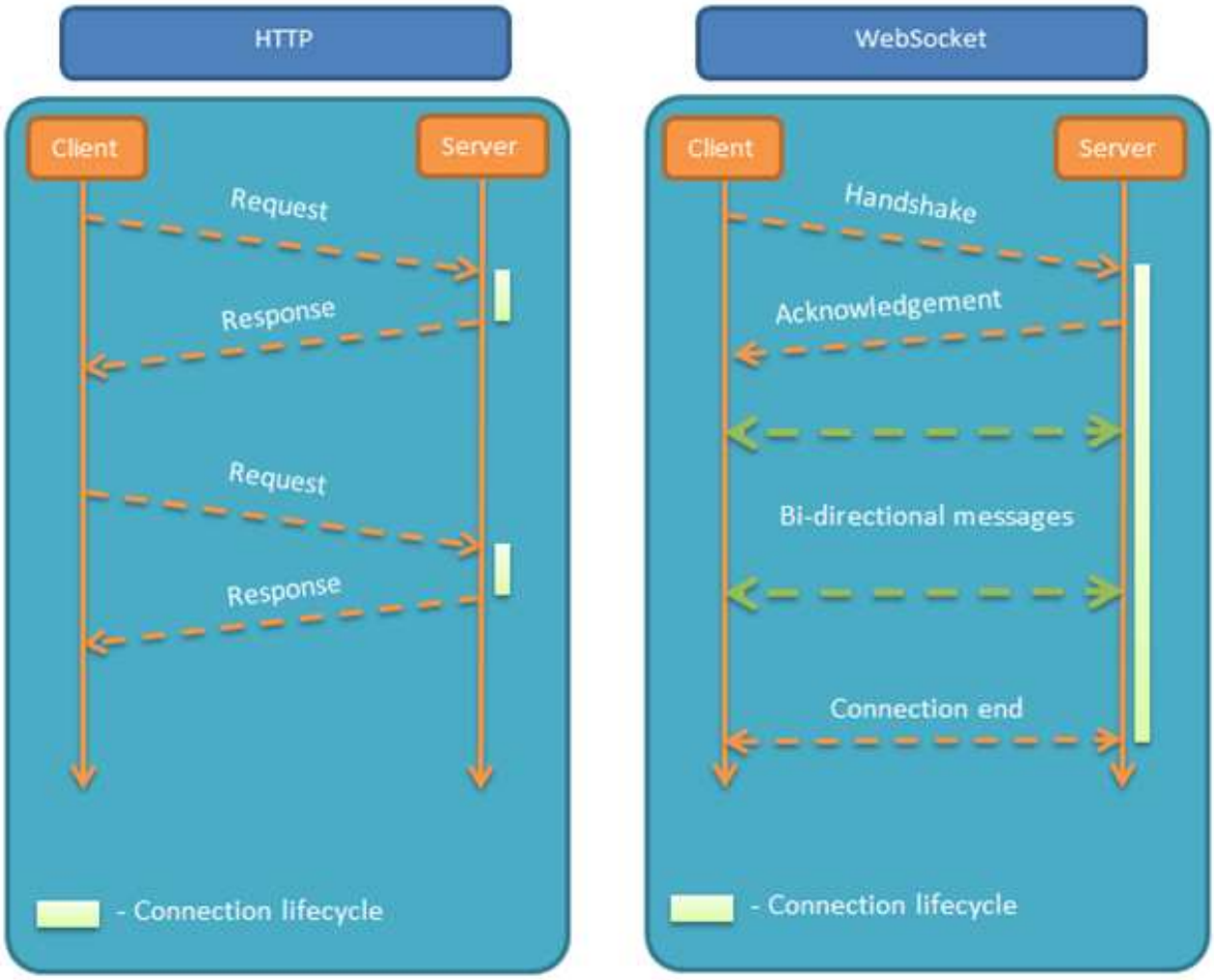


SPDY vs WebSocket

> 两者侧重点不同，SPDY侧重于给Web页面加载提速，而WebSocket强调于提供一种双向的通讯机制以及API；两者为竞争关系。

- WebSocket介绍

- 出现背景：很多网站为了实现推送技术，所用的技术都是轮询。轮询是在特定的的时间间隔（如每1秒），由浏览器对服务器发出HTTP请求，然后由服务器返回最新的数据给客户端的浏览器。这样会浪费很多的带宽等资源。
- WebSocket：在单个TCP连接上进行全双工通讯的协议
- 使用ws/wss为统一资源标志符，类似于https，wss表示在TLS上的ws；WebSocket使用和HTTP相同的TCP端口（80/443）



- SPDY与WebSocket对比

SPDY	WebSocket
基于HTTP的会话层协议	不限于HTTP的应用层协议（？）
基于SSL/TLS	SSL/TLS可选
无	有定义API

- WebSocket JS API

```
interface WebSocket : EventTarget
{
    readonly attribute DOMString url;
    attribute EventHandler onopen;    // 用于指定连接成功后的回掉函数
```

```
    attribute EventHandler onerror;
    attribute EventHandler onclose
    attribute EventHandler onmessage; // 用于指定收到数据后的回掉函数
    void send(DOMString data);
    void send(Blob data);
    void close(...);
    var readyState;    // CONNECTING/OPEN/CLOSING/CLOSED
    ...
};
```

- WebSocket握手请求示例:基于HTTP1.1

```
GET / HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Host: example.com
Origin: http://example.com
Sec-WebSocket-Key: sN9cRrP/n9NdMgdcy2VJFQ==
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: fFBooB7FAkLlXgRSz0BT3v4hq5s=
Sec-WebSocket-Location: ws://example.com/
```