第一页 无

第二页 下面我会简单选一些方面用到头部字段进行讲解

第三页 还需要在响应体中附上永久性URI的超链接文本，这里我也不是很明白该如何附上永久性url的超链接文本？？？？？？

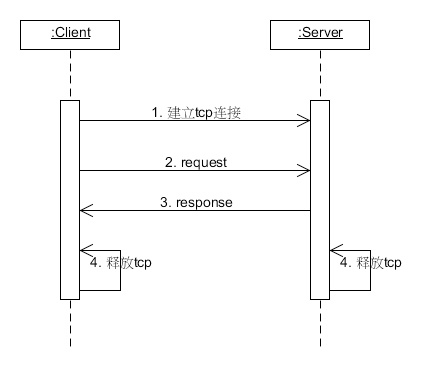
第四页 ssl安全套接层安全协议，现在升级成tls传输层安全协议

1、认证用户和服务器，确保数据发送到正确的客户机和服务器；

2、加密数据以防止数据中途被窃取；

3、维护数据的完整性，确保数据在传输过程中不被改变。

重复请求head：当请求



4.Keep-Alive工作原理

与HTTP1.0需要主动声明不同的是，HTTP1.1默认支持这一特性，两者的交互流程如下：

HTTP1.0 Keep-Alive的数据交互流程:

建立tcp连接

Client 发出request，并声明HTTP版本为1.0，且包含header:"Connection： keep-alive"。

Server收到request，通过HTTP版本1.0和"Connection： keep-alive"，判断连接为长连接；故Server在response的header中也增加"Connection： keep-alive"。

同时，Server不释放tcp连接，在Client收到response后，认定为长连接，同样也不释放tcp连接。这样就实现了会话的保持。

直到会话保持的时间超过keepaliveTime时，client和server端将主动释放tcp连接。

HTTP1.1 Keep-Alive的数据交互流程:

建立tcp连接

Client 发出request，并声明HTTP版本为1.1。

Server收到request后，通过HTTP版本1.1就认定连接为长连接；此时Server在response的header中增加"Connection： keep-alive"。

Server不释放tcp连接，在Client收到response后，通过"Connection： keep-alive"判断连接为长连接，同样也不释放tcp连接。

这个过程与http1.0类似，仅是http1.1时，客户端的request不用声明"Connection： keep-alive"。

HTTP/1.1并不支持 HTTP 首部压缩，为此 SPDY 和 HTTP/2 应运而生， SPDY 使用的是通用的DEFLATE 算法，而 HTTP/2 则使用了专门为首部压缩而设计的 HPACK 算法。

spdy

SPDY的主要目的是减少50%以上的页面加载时间，但是呢不增加部署的复杂性，不影响客户端和服务端的Web应用，只需要浏览器和Web服务器支持SPDY。主要有以下几点：

多路复用，一个TCP连接上同时跑多个HTTP请求。请求可设定优先级。

去除不需要的HTTP头，压缩HTTP头，以减少需要的网络带宽。

使用了SSL作为传输协议提供数据安全。

对传输的数据使用gzip进行压缩

提供服务方发起通信，并向客户端推送数据的机制。

实质上，SPDY就是想不影响HTTP语义的情况下，替换HTTP底层传输的协议来加快页面加载时间。

三、WebSocket

WebSocket则提供使用一个TCP连接进行双向通讯的机制，包括网络协议和API，以取代网页和服务器采用HTTP轮询进行双向通讯的机制。

本质上来说，WebSocket是不限于HTTP协议的，但是由于现存大量的HTTP基础设施，代理，过滤，身份认证等等，WebSocket借用HTTP和HTTPS的端口。

由于使用HTTP的端口，因此TCP连接建立后的握手消息是基于HTTP的，由服务器判断这是一个HTTP协议，还是WebSocket协议。 WebSocket连接除了建立和关闭时的握手，数据传输和HTTP没丁点关系了。

WebSocket也有自己一套帧协议

四、SPDY和WebSocket的关系

SPDY和WebSocket的关系比较复杂。

补充关系，二者侧重点不同。SPDY更侧重于给Web页面的加载提速，而WebSocket更强调为Web应用提供一种双向的通讯机制以及API。

竞争关系，二者解决的问题有交集，比如在服务器推送上SPDY和WebSocket都提供了方案。

承载关系，试想，如果SPDY的标准化早于WebSocket，WebSocket完全可以侧重于API，利用SPDY的帧机制和多路复用机制实现该API。 Google提出草案，说WebSocket可以跑在SPDY之上。WebSocket的连接建立在SPDY的流之上，将WebSocket的帧映射到SPDY的帧上。

融合关系，如微软在HTTP Speed+Mobility中所做的。

第五页：无

第六页：无

第七页：

1. HTTP Speed+Mobility

还有一个有趣的技术叫做HTTP Speed+Mobility，和SPDY一样都是HTTP 2.0标准的竞争者，HTTP Speed+Mobility来自微软。HTTP SM借鉴了SPDY和WebSocket的协议，将二者揉为一体，又有所取舍。

HTTP SM的设计原则包括：

保留HTTP的语义，这一点和SPDY一致，但也正应如此，抛弃了SPDY里的ServerPush。

遵守分层的网络架构，TCP能做的，HTTP SM不做，因此去除了SPDY的流控。

使用现有标准，因此使用HTTP/1.1 Upgrade header机制，借用了WebSocket的握手机制和帧格式（RFC6455）。

客户端掌握内容的控制，因此不强制使用压缩和SSL/TLS。

考虑到网络的费用和电力，这点考虑到了移动设备以及物联网，提供了Credit Control机制。

HTTP SM分以下几层：

会话层和帧协议，这部分取自WebSocket协议。包括握手机制，以及帧格式。

流层（包括多路复用），这部分主要借鉴SPDY，包括多路复用，流优先级，但增加了Credit Control。这部分作为 WebSocket协议的扩展。

HTTP层，在流层上实现HTTP语义，这部分也借鉴自SPDY。

2. Network-Friendly HTTP

NF是HTTP 2.0候选方案之一，主要提出以下改进：

对HTTP头的名称进行二进制编码

对通用HTTP头进行分组

请求/应答的多路复用

分层模型

NF同样定义了帧和流，