实时流协议

学号 陈照欣
北京理工大学徐特立学院 30141902 班，北京 100081
(1120191086@bit.edu.cn)

Real Time Streaming Protocol

Chen Zhaoxin
Class 30141902,School of Xu Teli, Beijing Institute of Technology, Beijing 100081

Abstract Real Time Streaming Protocol (RTSP) is an application-level network communication system that transmits real-time data from multimedia to endpoint devices by communicating directly with the server streaming data.
The protocol establishes and controls media streams between client devices and servers by acting as a network remote control of continuous media streams (such as audio and video) that are synchronized in time. It does not stream multimedia by itself, but communicates with a server that streams multimedia data. For example, when a user pauses a video that he is streaming, RTSP delivers a request for the user to pause the video to the video streaming server.
The RTSP was developed by RealNetworks, Netscape, and Columbia University, and the first draft was submitted to the Internet Engineering Task Force (IETF) in 1996. The IETF's Multi-Party Multimedia Conversation Control Work Group (MMUSIC WG) was standardized and published as RFC 2326 in 1998. RTSP 2.0 was released as RFC 7826 in 2016 as an alternative to RTSP 1.0. RTSP 2.0 is based on RTSP 1.0, but is not backward compatible except for the basic version negotiation mechanism.

Keywords RTSP; streaming; server communication

摘要 实时流协议（Real Time Streaming Protocol，RTSP）是一种应用级网络通信系统，它通过直接与流传输数据的服务器进行通信，将实时数据从多媒体传输到端点设备。

RTSP 是通过作为非异步的连续媒体流（例如音频和视频）的网络远程控制来建立和控制客户端设备和服务器之间的媒体流。它本身并不流式传输多媒体，而是与流式传输多媒体数据的服务器通信。例如，当用户暂停他正在流式传输的视频时，RTSP 会将用户暂停视频的请求传送到视频流服务器。

RTSP 由 RealNetworks 公司，Netscape 公司和哥伦比亚大学开发，第一稿于 1996 年提交给网际网络工程任务组（IETF）。由 IETF 的多方多媒体对谈控制工作群组（MMUSIC WG）进行了标准化，并于 1998 年发布为 RFC 2326。RTSP 2.0 于 2016 年发布为 RFC 7826，作为 RTSP 1.0 的替代品。RTSP 2.0 基于 RTSP 1.0，但除了基本的版本协商机制之外不向下相容。

关键词 RTSP；串流媒体；服务器通信

1、介绍

1.1 目的

实时流协议 (RTSP) 建立和控制单个或多个时间同步的连续流音频和视频等媒体。一般情况下，尽管连续的交错带有控制流是可能的，协议不会提供连续流本身。可以说，RTSP 为多媒体服务器起到了“网络遥控器”的作用。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

RTSP 中实质上并没有连接，而是用会话的概念来代替。与连接的维护方法类似，每一个会话都有一个特定的独一无二的标识符标记，并且由服务器进行分

配、识别和维护。此外，与其他协议不同，实时流协议并没有绑定诸如 TCP 连接之类的非实时流协议传输层连接。RTSP 会话过程中，一个实时流协议客户端可以建立许多传输连接到服务器，并且发出 RTSP 请求。或者可以使用无连接协议，例如 UDP。

1.2 术语

1. 集合控制 Aggregate Control

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

2. 连续媒体数据 Continuous media

实时流协议 source 和实时流协议 sink 间有数据的时序关系，即 sink 必须还原出 source 中数据间的时序关系。最普通的例子是音视频动画。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

3. 呈现描述 Presentation description

presentation 描述包含了能够展现出的所有流的信息，如编码、网络地址以及。presentation 描述有多种格式，包含但不限于 SDP 格式。

4. RIP 路由协议

RIP 路由信息协议是最古老的距离矢量路由协议中的一个。协议使用跳步数来表示路由的矩阵。RIP 通过限制最大跳步数的方式来避免网络中的环路。但这一点同时也限制了 RIP 所能支持的网路的大小。

2、RTSP 传输机制

2.1 RTSP 文法

与 HTTP 语法类似，RTSP 语法使用 BNF 范式描述。

```
OCTET    = <any 8-bit sequence of data>
CHAR     =<any US-ASCII character (octets 0-127)>
UPALPHA=<any US-ASCII uppercase letter "A".."Z">
LOALPHA=< any US-ASCII lowercase letter "a".."z">
ALPHA    = UPALPHA | LOALPHA
DIGIT    =<any US-ASCII digit "0".."9">
CTL      =<any US-ASCII control character
           (octets 0 - 31) and DEL (127)>
CR       =<US-ASCII CR, carriage return (13)>
LF       =<US-ASCII LF, linefeed (10)>
SP       =<US-ASCII SP, space (32)>
HT       = <US-ASCII HT, horizontal-tab (9)>
<">      = <US-ASCII double-quote mark (34)>
CRLF     = CR LF
LWS      =[CRLF] 1*( SP | HT )
TEXT     =<any OCTET except CTLs>
Tspecials="(" | ")" | "<" | ">" | "@"
          |  "," | ";" | ":" | "\" | <">
          |  "/" | "[" | "]" | "?" | "="
          |  "{" | "}" | SP | HT
token    =1*<any CHAR except CTLs or tspecials>
quoted-string= ( <"> *(qdtext) <"> )
qdtext    =<any TEXT except <">>
quoted-pair="\" CHAR
message-header=ield-name ":" [ field-value ] CRLF
field-name=token
field-value=*( field-content | LWS )
field-content=<the OCTETs making up the field-value
              and consisting of either *TEXT or
              combinations of token, tspecials, and
              quoted-string>
safe          = "\$" | "-" | "_" | "." | "+"
extra         ="!" | "*" | "$'$" | "(" | ")" | ","
hex           = DIGIT | "A" | "B" | "C" | "D" | "E" | "F" |
                "a" | "b" | "c" | "d" | "e" | "f"
escape        = "\%" hex hex
reserved      = ";" | "/" | "?" | ":" | "@" | "&" | "="
unreserved    = alpha | digit | safe | extra
xchar         =unreserved | reserved | escape
```

2.2 RTSP 消息格式

1. 请求消息

方法 URI RTSP 版本 CR LF
消息头 CRLF CRLF
消息体 CR LF

其中，方法包括 OPIONS、DESCRIBE、SETUP、PLAY、TEARDOWN 等,URI 为接收端地址。实时流协议版本一般都是 RTSP/1.0。每行末尾 CR LF 表示换行，需要接收端进行实时解析，最后一个 header 需要有两个换行符。消息体可有可无的，有的 Request 消息并没有消息体。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户

端可以使用一条播放/暂停指令控制所有的音视频流。

2. 回复消息

RTSP 版本 状态码 解释 CR LF

消息头 CR LF CR LF

消息体 CR LF

状态码是一个二进制数值，用于直观地表示 Request 消息的执行结果以及反馈，解释是与状态码相对应的含义解释。presentation 描述文件可以由客户端通过网页或电子邮件等其他方式获得，不一定存储在媒体服务器上。假设 presentation 描述描述了一个或多个 presentation，其中每个 presentation 保持公共时间轴。 为了说明的简单性和不失一般性，假设表示描述恰好包含一个这样的表示。 一个演示文稿可能包含多个媒体流。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

2.3 整体流程

每个媒体流都有自己独特的 RTSP URL 标识。整个 presentation 和构成 presentation 的媒体的属性由 presentation 描述文件定义，其格式超出了本规范的范围。

演示描述文件包含对构成演示的媒体流的描述，包括它们的编码方式。此外还有编程语言等信息，使客户端能够选择最合适的媒体组合。 在此演示描述中，可由 RTSP 单独控制的每个媒体流由 RTSP URL 标识，该 URL 指向处理该特定媒体流的媒体服务器并命名存储在该服务器上的流。 多个媒体流可以位于不同的服务器上； 例如，音频和视频流可以跨服务器拆分以实现负载共享。 该描述还列举了服务器能够使用的传输方法。

2.4 RTSP 消息交互过程

RTSP 信息交互过程

Step1：查询服务器端的可用方法	C->S:OPTION request S->C:OPTION response
Step2：得到媒体描述信息	C->S:DESCRIBE request S->C:DESCRIBE response
Step3：建立 RTSP 会话	C->S:SETUP request S->C:SETUP response
Step4：请求开始传送	C->S:PLAY request S->C:PLAYresponse
Step5：数据传送播放中	S->C:发送流媒体数据
Step6：关闭会话，退出	C->S:TEARDOWN request S->C:TEARDOWN response

3、 RTSP 性能

3.1 易解析 Easy to parse

标准 HTTP 或者 MIME 解析器可以用来解析 RTSP

Header	type	support	methods
Accept	R	opt.	entity
Accept-Encoding	R	opt.	entity
Accept-Language	R	opt.	all
Allow	r	opt.	all
Authorization	R	opt.	all
Bandwidth	R	opt.	all
Blocksize	R	opt.	all but OPTIONS, TEARDOWN
Cache-Control	g	opt.	SETUP
Conference	R	opt.	SETUP
Connection	g	req.	all
Content-Base	e	opt.	entity
Content-Encoding	e	req.	SET_PARAMETER
Content-Encoding	e	req.	DESCRIBE, ANNOUNCE
Content-Language	e	req.	DESCRIBE, ANNOUNCE
Content-Length	e	req.	SET_PARAMETER, ANNOUNCE
Content-Length	e	req.	entity
Content-Location	e	opt.	entity
Content-Type	e	req.	SET_PARAMETER, ANNOUNCE
Content-Type	r	req.	entity
CSeq	g	req.	all
Date	g	opt.	all
Expires	e	opt.	DESCRIBE, ANNOUNCE
From	R	opt.	all
If-Modified-Since	R	opt.	DESCRIBE, SETUP
Last-Modified	e	opt.	entity
Proxy-Authenticate			
Proxy-Require	R	req.	all
Public	r	opt.	all
Range	R	opt.	PLAY, PAUSE, RECORD
Range	r	opt.	PLAY, PAUSE, RECORD
Referer	R	opt.	all
Require	R	req.	all
Retry-After	r	opt.	all
RTP-Info	r	req.	PLAY
Scale	Rr	opt.	PLAY, RECORD
Session	Rr	req.	all but SETUP, OPTIONS
Server	r	opt.	all
Speed	Rr	opt.	PLAY
Transport	Rr	req.	SETUP
Unsupported	r	req.	all
User-Agent	R	opt.	all
Via	g	opt.	all
WWW-Authenticate	r	opt.	all

3.2 安全 Secure

RTSP 使用的网络安全机制较为丰富。所有的 HTTP 安全机制，例如 basic 认证 digest 认证，都有可能被使用。此外，传输层和网络层的安全机制也可能使用。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

由于实时流协议和 HTTP 服务器在语法和使用上

具有相似性，两者的安全机制类似。

1. 认证机制

实时流协议和 HTTP 的认证机制相同，因此在身份认证方面应该遵循相同的规则。

2. 滥用服务器日志信息

容易得知，实时流协议和 HTTP 使用相似的日志机制，因此在保护日志信息时应该被平等的防护以保护服务器使用者的信息。

3. 敏感信息的传输

我们没有理由认为使用 HTTP 协议传输的数据一定比使用 RTSP 协议传输的数据更敏感。因此，RTSP 客户端、服务器和代理的实施者可以使用所有关于保护数据隐私和用户隐私的预防措施。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

RTSP 协议考虑到网络中可能存在着一些自私结点，为了延长 RTSP 协议的自组织网络的寿命，通常采用以下两种方法。首先是减少传播拓扑控制消息的中继结点的数量以及考虑这些中继结点的剩余能量水平。这些目标都可以通过集群概念来成功部署到 RTSP 中。RTSP 就是这样一种基于节点剩余能量和连通性指标的新型聚类算法和中继结点选择算法。协议的内在机制会鼓励结点在路由的过程中更多的正常工作。仿真结果表明，基于能量和连通性的新型 RTSP 模型可以有效地延长网络寿命。

3.3 传输层独立性 Transport-independent

实时流协议可能会使用不可靠（unreliable）数据报（datagram）协议 UDP、可靠数据报协议 RDP（RFC 1151）、或保证了应用层可靠性的可靠流协议如 TCP。

3.4 多服务器兼容（Multi-server capable）

演示文稿中的每个实时流协议媒体流都可以驻留在不同的服务器上。 客户端自动与不同的媒体服务器建立多个并发控制会话。 媒体同步在传输层执行。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

3.5 媒体控制和会议初始化相独立（Separation of stream control and conference initiation）

当邀请一个媒体服务器进入会议时，流控制被分割。唯一的条件是，会议初始化协议要提供或者可以被用于生成一个唯一的标识符。实际情况下，通常会使用 SIP 和 H.323 协议邀请某服务器进入会议。

3.6 中立的呈现描述（Presentation description neutral）

RTSP 并没有将 presentation 描述具体化，并且可携带当前使用的描述格式。但至少，presentation 描述中要包含一个 RTSP URI.

3.7 代理和防火墙亲和性（Proxy and firewall friendly）

应用层和传输层防火墙可以轻易地处理实时流协议协议。防火墙需要能够处理 SETUP 方法为 UDP 媒体流打开一个通道。

3.8 HTTP 亲和性（HTTP-friendly）

由于 RTSP 使用了一部分 HTTP 的概念，所以现有的框架都可以被重用，如用于关联内容和标签的 PICS(Platform for Internet Content Selection)。然而，RTSP 也不仅仅只是在 HTTP 基础上添加方法而已，因为大多是情况下，RTSP 还需要维护服务器状态来实现连续媒体的控制。

3.9 恰当的服务器控制（Appropriate server control）

如果一个客户端可以开启一个流，它一定要能够关闭这个流。这种情况下，服务器不被允许向客户端传输客户端本身不能截断的流。

3.10 能力协商（Capability negotiation）

若协议中的基础属性不被允许使用，一定有明确的机制让客户端知悉哪些方法是无效的，由此客户端可以提供正确的用户接口。例如，最常使用的移动进度条功能，是 seeking 特性的实现。当 seeking 特性不受支持时，用户不能移动进度条。

4、RTSP 状态及转换

4.1 RTSP 状态概述

RTSP 控制一个可能是通过单独协议所传输的媒体。这个传输协议可能与控制通道无关。因此，倘若媒体服务器并没有接收到 RTSP 请求，数据的传输依然可能不被中断。整个生命周期内，单个媒体流仍然可能会被不同 TCP 连接传输而来的 RTSP 请求所控制。因此，服务器需要维护会话状态以便能够正确关联同一个流的 RTSP 请求。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

RTSP 中许多方法并不会改变状态，会修改状态的方法有如下几个：

1. SETUP
   使服务器为流分配资源并开启 RTSP 会话
2. PLAY 和 RECORD
   在 SETUP 中分配的流上开始数据传送
3. PAUSE
   在不释放服务器资源的前提下暂时中断数据流。
4. TEARDOWN
   释放数据流相关的资源，终止服务器端实时流协议对话。

4.2 状态转换

1. 客户端状态机

状态名	描述
Init	SETUP 信息已发送，正在等待回应
Ready	已收到 SETUP 回应或者在 playing 状态中受到 PAUSE 回复
Playing	收到 Play 信息的回复
Recording	收到 RECORD 信息的回复

通常来说，客户端会在接收到回复的时候改变状态。应当注意到，当某些需求发生在未来某一时间时（例如 PAUSE），状态依然会发生相应的改变。如果对象不需要明确的 SETUP（例如，它可以通过多播组获得），则状态从准备态开始。此时准备和播放两种状态。当到达请求范围的末尾时，客户端还会将状态从 Playing/Recording 更改为 Ready。

State	Message sent	Next state after response
Init	SETUP TEARDOWN	Ready Init
Ready	PLAY RECORD TEARDOWN SETUP	Playing Recording Init Ready
Playing	PAUSE TEARDOWN PLAY SETUP	Ready Init Playing Playing
Recording	PAUSE TEARDOWN RECORD SETUP	Ready Init Recording Recording

2. 服务器端状态机

状态名	描述
Init	初始状态，尚未收到合法的 SETUP 信息
Ready	已收到 SETUP 信息，并且已回复。
Playing	成功接收上一个 PLAY 请求，对其回复发出。数据正在发送
Recording	服务器正在记录媒体数据

通常来说，服务器端在收到请求时即改变状态。当服务器处于播放或记录状态并且处于单播模式，如果它在定义的时间间隔内[1]没有从客户端接收到“健康”信息，例如 RTCP 报告或实时流命令，它可以恢复到 Init 状态并停止实时流对话。服务器可以在

[1] 通常为一分钟

Session 响应头中声明另一个超时值。当服务器处于 Ready 状态，如果超过一分钟的时间间隔没有收到 RTSP 请求，它可能会恢复为 Init。请注意，某些请求（例如 PAUSE）可能会在未来的某个时间或位置生效，并且服务器状态会在适当的时间发生变化。服务器在客户端请求的范围结束时从播放或者记录状态恢复到准备状态。服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

REDIRECT 消息在发送后立即生效，除非它有范围标头指定重定向何时生效。在这种情况下，服务器状态也会在适当的时候发生变化。、

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

如果对象不需要明确的 SETUP 指令，则状态从 Ready 开始，只有两种状态，Ready 和 Playing。

“Next State”列指示在发送成功响应后假定的状态。如果请求导致状态码为 3xx，则状态变为 Init。状态码 4xx 不会导致任何变化。

State	Message Received	Next State
Init	SETUP TEARDOWN	Ready Init
Ready	PLAY SETUP TEARDOWN RECORD	Playing Ready Init Recording
Playing	PLAY PAUSE TEARDOWN SETUP	Playing Ready Init Playing
Recording	RECORD PAUSE TEARDOWN SETUP	Recording Ready Init Recording

5、RTSP 与其他协议的关系及扩展

5.1 RTSP 与 HTTP 关系

实时流协议和 HTTP 存在功能交集，它还可以与 HTTP 交互，因为与流内容的初始联系通常是通过网页进行的。当前的协议规范旨在允许 Web 服务器和实现实时流协议的媒体服务器之间的不同切换点。例如，可以使用 HTTP 或 实时流协议检索表示描述，这减少了基于 Web 浏览器的场景中的往返，但也允许完全不依赖 HTTP 的独立实时流服务器和客户端。由 实时流协议控制的流可以使用实时传输协议(RTP)，但操作实时流协议不依赖于用于承载的传输机制连续媒体。

然而，RTSP 与 HTTP 之间仍然存在着本质上的不同。数据传输是在不同的协议中带外进行的。HTTP 是一个不是对称的，只允许客户端发出请求，同时服务器负责回复。而在实时流协议中，实时流协议客户端和 RTSP 媒体服务器均可以发出请求，并且允许以全双工和半双工方式运行。此外，实时流协议请求是有状态的，他们可能需要在请求得到确认应答后才会进一步设置参数和后续控制，因此几乎所有的 RTSP 服务器都需要默认维护状态情况。

5.2 RTSP 扩展

由于并非所有媒体服务器都具有相同的功能，媒体服务器必然会支持不同的请求集。例如：

一个服务器可能只支持点播，也就不用支持 RECORD 请求。

如果一个服务器只支持直播，那么它可能不支持 seeking。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

有些服务器并不支持流参数的设置，也自然不需要支持 GET_PARAMETER 和 SET_PARAMETER。

RTSP 可通过三种方式进行扩展，根据变化程度排列如下：

1. 对于已有的方法，可以像 C++重构一样，通过添加新的不同的参数参数进行方法的扩展，同时接收端也能安全地选择忽略这些参数。如果客户端需要在某个方法扩展不受支持时获得一个负确认，需要在 Require 头文字中添加相应的信息。
2. 可以添加新的方法。如果信息接收端无法解析请求，则像发送端发送一个代码 501，之后发送端不应该再使用该方法。此时客户端应使用 Public 回复头文字将支持的方法全部列出
3. 可定义新版本协议，几乎允许变动所有元素

6、总结

随着互联网的逐渐发展，无线通信的逐渐普及，移动无线自组织网络技术在国内外都越来越得到研究人员的重视。其中最为关键的一点就是路由技术。由于移动无线自组织网络节点具有很高的移动性，导致网络的拓扑结构会经常的发生变化。这样的特殊性为路由协议的设计引入了更多复杂的问题。

在实时流协议出现之前，HTTP 可能是最受欢迎的协议。我们可以设想，有这样一个场景是在网页上浏览一个视频，根据 HTTP 协议，正常的流程是访问它的 URL 并且开始下载。缓存完毕后播放。对于较为简陋的视频设备、显示器，或者是糟糕的网络带宽，这样的过程可能会很漫长。HTTP 可能会多次断开并尝试重连，最后还需要校验文件的完整性，极大影响了观看体验。毕竟当时的分辨率、帧率和带宽限制了通过互联网传输的媒体文件的大小，信息共享只能通过各种硬盘、U 盘、CD 等以存储文件的形式进行传输。

随着硬件设备技术的发展，采集设备的分辨率不断提升。显示器支持更高的帧率，网络带宽也成倍增长，这为更好的观看体验奠定了基础。随着网络资源的日益丰富，用户时间的稀缺性日益突出。为了快速观看和判断视频本身是否符合您的口味，在线实时观看成为一大诉求。而传统的 HTTP 下载显然无法满足这一需求，因此 RTSP 在流媒体的追捧中脱颖而出。

服务器或者客户端可以同时对多条数据媒体流进行控制。以最常见的音视频流来说，这意味着客户端可以使用一条播放/暂停指令控制所有的音视频流。

RTSP 的实施效果非常好，但它的缺点是需要转换协议才能播放。相信未来在这个领域中会涌现更多更有效的路由协议。

参 考 文 献

[1] 彭国刊. 一个基于 RTSP 协议的自适应流媒体传输系统[D].华南理工大学,2014.

[2] Performance Comparison of Wireless Mobile Ad-Hoc Network Routing Protocols Arun Kumar B. R. Lokanatha C. Reddy Prakash S. Hiremath

[3]https://en.wikipedia.org/wiki/ Real Time Streaming Protocol

[4] M. Akhlaq and T. R. Sheltami, "RTSP: An Accurate and Energy-Efficient Protocol for Clock Synchronization in WSNs," in IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 3, pp. 578-589, March 2013, doi: 10.1109/TIM.2012.2232472.

[5] F. Suárez-Ruiz, T. S. Lembono and Q. Pham, "RoboTSP – A Fast Solution to the Robotic Task Sequencing Problem," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 1611-1616, doi: 10.1109/ICRA.2018.8460581.

[6] H. Schulzrinne, "A comprehensive multimedia control architecture for the Internet," Proceedings of 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '97), 1997, pp. 65-76, doi: 10.1109/NOSDAV.1997.629366.

[7] Schulzrinne, H., "RTP profile for audio and video conferences with minimal control", RFC 1890, January 1996.

[8] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext transfer protocol - HTTP/1.1", RFC 2068, January 1997.