合肥工業大學

英文翻译

（翻译中文）

译文题目 网络中的实时通信：问题、成就和正在进行的标准化努力

学生姓名 李磊

学　　号 2013214381

专业班级 软件 13-2 班

2017 年 6 月 1 日

网络中的实时通信：问题、成就和正在进行的标准化努力

作者：Salvatore Loreto 是芬兰爱立信研究所的 NomadicLab 多媒体技术部门的高级研究员。 他在互联网传输协议，信号协议，VoIP，IP 电话融合，IP 会议，3GPP IP 多媒体子系统，HTTP 和 Web 技术方面做出了贡献。 Loreto 在那不勒斯大学获得计算机网络博士学位。 他在 IETF 内担任 SIP 过载控制（SOC），双向或服务器启动 HTTP（HyBi）和应用程序区域（Appsawg）工作组的联合主席。 请通过 salvatore.loreto@ieee.org 与他联系。

Simon Pietro Romano 是那不勒斯大学计算机工程系的助理教授，是 Meetecho 的创始人之一，该公司是一家专注于基于互联网的会议和协作的创业公司（以及该大学的分支机构）。 他的研究兴趣主要在网络领域，特别是关于实时多媒体应用，网络安全和自主网络管理。 Romano 拥有那不勒斯大学计算机网络的博士学位。 他积极参与实时应用和基础设施领域的 IETF 标准化活动。

摘要：Web 实时通信（WebRTC）是一个即将到来的用来实现 Web 浏览器之间的对等实时通信的标准。IETF RTCWeb 和 W3C WebRTC 工作组是共同的定义 API 和用于设置并管理任何一对下一代网络浏览器之间的可靠通信通道的底层通信协议。

1. 背景

对 Web 中实现实时通信（RTC）支持目前得到了两个主要的互联网标准化机构 - IETF 和 W3C 的推动。该领域的标准化活动旨在定义一个 W3C API，通过安全访问输入外设（例如网络摄像头和麦克风），在任何设备上运行的 Web 应用程序可以在对等网络的浏览器之间中发送和接收实时媒体和数据。 API 的设计必须允许 Web 开发人员实现用于在通信会话中查找和连接参与者的功能。 W3C API 将依赖于已经被 IETF 社区确定为最适合于解决网络相关方面（控制协议，连接建立和管理，无连接传输，选择最合适的编码器和解码器等）的现有协议。然而，在两个标准化活动之间不存在清楚的区分（清楚地在单个节点中的应用级职责和远程节点之间的相互通信活动之间的接口处切断）。

这种向支持浏览器的 RTC 的迁移代表了一个重大突破，并激励了许多行业和学术研究者最近的工作。 在这里，我们讨论人们对于将互动多媒体功能集成到 Web 应用程序的日渐增长的兴趣。

2. 我们怎么到这里的？

一些文档试图促进支持浏览器的 RTC 代替传统的通信服务。在一个明显的例子中，研究人员将传统电信系统的复杂性与 Web 架构【1】杂性进行了比较，结论是前者应该向后者移动，以使这种通信能够尽可能多地用于用户。本文还为一个（现在过期的）描述了对会话发起协议（SIP）的 RESTful 接口 Internet 草案【2】做了铺垫。研究人员和实现者已经为类似的方法付出了许多其他的努力，但通常以非标准方式（例如，通过使用诸如 Adobe Flash 或 Microsoft ActiveX 的专有插件）并且没有文档。一个显着的例外是来自爱立信实验室的最新工作，研究人员试图在浏览器本身中通过利用临时的 JavaScript API 和使用了 gstreamer 的 WebKit 的正确修改版本（参见 https：//labs.ericsson.com/developercommunity/blog/beyond-html5-peerpeer-conversational-video）为实时传输协议（RTP）和媒体设备添加原生支持。

也就是说，这种迁移导致主要的标准化机构为这个问题投入新鲜的能源，最终形成了两个不同但又相互关联的工作组：IETF 内的 RTCWeb（http：//tools.ietf.org/wg/ rtcweb /章程）和 WebRTC（www.w3.org/2011/04/ webrtc-charter.html）。RTCWeb 专注于 IETF 必须解决的协议和交互，包括与传统系统（例如现有的电信系统）的互操作性。 WebRTC 正在努力定义一种允许浏览器和脚本语言与媒体设备（麦克风，网络摄像头和扬声器），处理设备（编码器/解码器）和传输功能交互的 API。这样的努力可能会扩展和增强 HTML5 规范，这个规范已经提供了一种标准的方式来从服务器到浏览器传递流式多媒体内容。

两个工作组都必须考虑将要解决的功能所产生的安全问题。我们期望（并希望看到）在这些工作组的生命周期中可以实现几个原型。

3. 开放 Web 平台

HTML5 通常被用来作为所谓的开放网络平台上取得的进步的总称，尽管 HTML 本身只是用于开发 Web 应用程序各种特征的一部分并且通常被引用为该平台的一部分。 完整的功能集还包括级联样式表（CSS），文档对象模型（DOM）约定，JavaScript 和多个脚本 API。 HTML 以结构化的方式表示应用程序及其数据，并允许开发人员使用 CSS 来设计应用程序的样式，并通过 JavaScript 控制它。所有这些技术都使用 HTTP 或 WebSocket（http：//tools.ietf.org/html/ rfc6455）通过 Web 基础架构（通过浏览器，代理和 Web 服务器）传递的。

脚本 API 允许程序员通过 JavaScript 正确地控制和利用浏览器功能。事实上，一旦向浏览器添加了新功能，W3C 还要设计新的 API，以便将这些功能公开给开发人员，让浏览器的功能更接近原生应用程序环境。

4. 架构

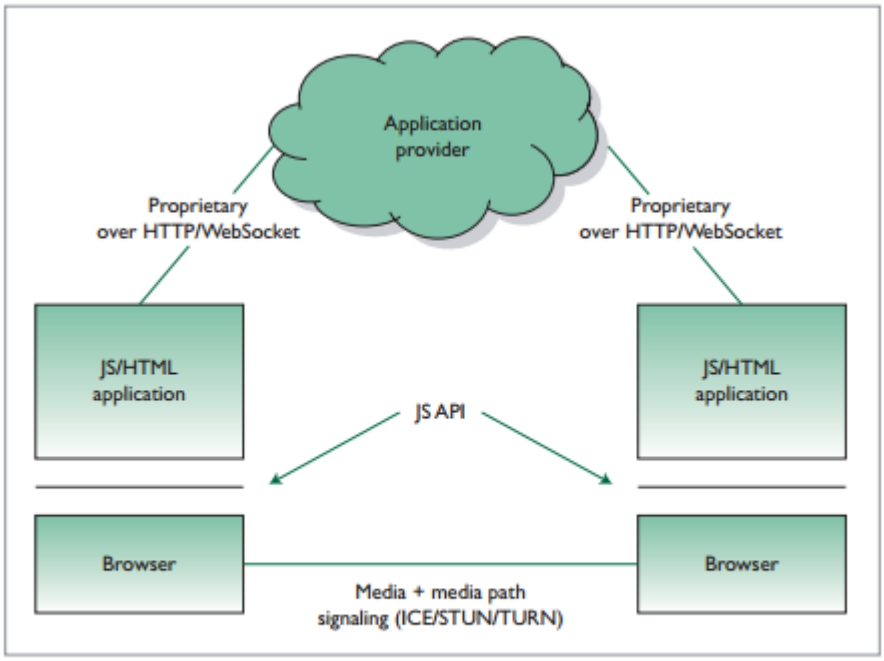

Figure 1. The RTCWeb architecture. This typical voice-over-IP communication trapezoid has a server-mediated signaling path and a direct (browser-to-browser) media path.

RTC 的架构模型是浏览器 RTC 梯形（见图 1），它允许媒体路径直接在浏览器之间流动，而无需任何中间服务器。信号路径可以跨越服务器，根据需要修改，解释或管理信号。

想法是让客户端 Web 应用程序（通常是混合 HTML 和 JavaScript）与 Web 浏览器通过 WebRTC API 交互，让他们正确使用和控制浏览器功能，并主动（例如，查询浏览器功能）和被动（接收浏览器生成的通知）地与浏览器进行交互。上述应用浏览器 API 因此必须提供一组广泛的功能，例如连接管理（以 P2P 方式），编码/解码能力，协商，选择和控制，媒体控制，防火墙和网络地址转换（NAT）遍历。从技术角度来看，API 是以 JavaScript 实现的，JavaScript 已经证明了它在 Web 应用程序的客户端上作为强大的脚本语言的有效性。

应用程序-浏览器之间的 API 的设计是一个具有挑战性的问题，但并不能解决手头的所有问题。 实际上，完整的图像设想了跨网络的连续的实时的数据流，沿着该路径没有另外的中间体，一次性完成两个（或甚至更多个）浏览器之间的直接通信。 因此，我们讨论的是浏览器到浏览器的通信，这是基于 Web 通信的革命性方法，因为它允许 P2P（每个对等体是浏览器）数据通信首次进入 Web 应用程序领域。

想象一下，在两个浏览器之间的实时音频和视频通话。 在这种场景中的通信可能涉及两个浏览器之间的直接媒体流，其中媒体路径通过涉及以下实体的复杂的交互序列来协商和实例化：

1) 调用者的浏览器和调用者的 JavaScript 应用程序（例如，通过 JavaScript API）；
2) 调用者的 JavaScript 应用程序和应用程序提供程序（通常为 Web 服务器）；
3) 应用程序提供程序和被调用 JavaScript 应用程序；
4) 被调用的 JavaScript 应用程序和被调用浏览器（再次通过应用程序-浏览器 JavaScript API）。

考虑到总体情况，我们接下来将详细介绍 RTCWeb 最相关的功能。

5. 信号

自成立以来，WebRTC 设计背后的一般思想是完全指定如何控制媒体平面，同时尽可能多地将信号平面留给应用层。 原理是不同的应用可能倾向于使用不同的标准化信号协议（例如 SIP 或 XMPP）或一些自定义的协议。在这种方法中，浏览器必须交换的重要信息是多媒体会话描述，其指定传输（和交互式连接建立[ICE]）信息以及媒体类型，格式和建立媒体通路所必需的所有相关联的媒体配置参数。

以会话描述协议（SDP）“blob”的形式交换会话描述信息的原始想法具有若干缺点，其中一些将难以处理。 因此，IETF 正在标准化 Javascript 会话建立协议【3】。 JSEP 提供了一个应用程序需要的用来处理协商过的的本地和远程会话描述（通过任何期望的信号机制进行协商）的接口，以及用于应用程序与 ICE 状态机交互的标准方法。 JSEP 方法将驱动信号状态机的责任完全丢给了应用程序。不是简单地转发浏览器发送到远程端的消息，应用必须在正确的时间调用正确的 API，并将会话描述和相关的 ICE 信息转换为其选择的信号协议定义的消息。

6. API

W3C WebRTC 1.0 API【4】使得 JavaScript 应用程序可以利用新颖的浏览器的实时功能。 它需要浏览器核心提供建立必要的音频，视频和数据通道所需的功能。 然而，正在进行的标准化工作还没做出关于浏览器核心将使用哪些音频（G.711，opus，vorbis 等）和视频（H.264，VP8 等）编解码器的决定。 当前的假设是所有媒体和数据流将始终被加密。

该 API 的设计围绕着三个主要概念：PeerConnection，MediaStreams 和 DataChannel。

6.1. 对等连接

PeerConnection 允许两个用户通过浏览器到浏览器直接通信。PeerConnection 表示与远程对等体的关联，远程对等体通常是在远程端运行的相同 JavaScript 应用程序的另一个实例。 通过由 Web 服务器的页面中的脚本代码提供的信号通道来协调通信 。例如，使用 XMLHttpRequest 或 WebSocket。 一旦调用浏览器建立对等连接，它可以直接发送 MediaStream 对象到远程浏览器。

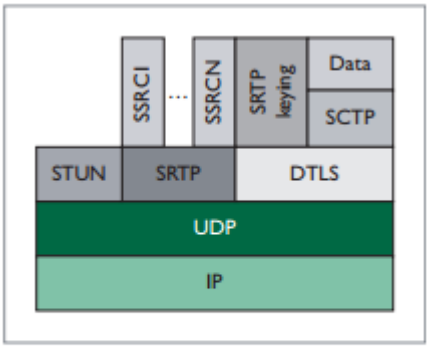

Figure 2. The RTCWeb protocol stack. RTCWeb allows for secure transmission of multiplexed real-time flows over the Internet.

如图 2 所示，对等连接机制使用 ICE 协议、STUN 服务器和 TURN 服务器，以使基于 UDP 的媒体流穿过 NAT 和防火墙。 ICE 允许浏览器发现有关其部署的网络的拓扑的足够的信息，以找到最佳的可利用的通信路径。 使用 ICE 还提供了一种安全措施，因为它可以防止不受信任的网页和应用程序向不期望接收它的主机发送数据。

远程主机在到达时将每个信号消息馈送到接收对等连接。 API 发送信号消息，大多数应用程序将被视为不透明团块，但是 Web 应用程序必须通过 Web 服务器安全而有效地转移到其他对等体。

6.2. 媒体流

MediaStream 是音频或视频的实际数据流的抽象表示。它用作管理媒体流上的操作的句柄，例如显示流的内容，记录它或将其发送到远程对等体。MediaStream 可以扩展为表示来自（远程流）或发送到远程节点（本地流）的流。LocalMediaStream 表示来自本地媒体捕获设备（例如网络摄像头或麦克风）的媒体流。

要创建和使用 LocalMediaStream，Web 应用程序必须通过 getUserMedia（）函数请求用户访问。 应用程序指定需要访问的媒体类型 - 音频或视频。 浏览器界面中的设备选择器授予或拒绝访问。 一旦应用程序完成，它可以通过调用 LocalMediaStream 上的 stop（）函数撤销自己的访问。

媒体面信号在对等体之间在带外执行。图 2 显示了媒体传输所需的协议栈。安全实时传输协议（SRTP）携带用于监视与数据流相关联的传输统计的媒体数据和 RTP 控制协议（RTCP）信息。 数据报传输层安全（DTLS）用作 SRTP 密钥和用于关联管理。 IETF 仍在讨论将 SDP 安全描述用作媒体流（SDES）作为替代关键字和关联管理的选项。

在多媒体会话中，每个介质通常通过各自的 RTCP 分组被携带在的单独的 RTP 会话中。 然而，为了解决给每个流打开新的端口的问题，IETF 正在努力尽可能地通过组合（即复用）单个 RTP 会话【5】中的多媒体传输来减少基于 RTP 的实时应用消耗的传输层端口的数量 。

6.3. 数据通道

DataChannel 提供一种通用传输服务，允许 Web 浏览器以双向，P2P 方式交换通用数据。在 IETF 内，标准化工作已经达成了关于使用封装在 DTLS 中的流控制传输协议（SCTP）来处理非媒体数据类型【6】的一般共识。在基于 UDP 的 ICE 上的 DTLS 上封装 SCTP，提供了 NAT 穿透、保密、源认证和完整性保护传输解决方案。此外，该解决方案使得数据传输与并行媒体传输平滑地互连，并且两者可

以共享单个传输层端口号。 IETF 选择 SCTP，因为它本身支持具有可靠，不可靠和部分可靠传递模式的多个流。它允许应用程序在 SCTP 关联中向对等 SCTP 端点打开几个独立流（每个方向上最多 65，536 个）。每个流实际上表示单向逻辑信道，提供有序传递。

应用程序可以发送有序或无序的消息序列。 仅为在同一流上发送的所有有序消息保留消息传递顺序。 但是，DataChannel API 是双向的，这意味着每个 DataChannel 绑定传入和传出的 SCTP 流。

当在实例化的 PeerConnection 对象上第一次调用 CreateDataChannel（）函数时，应用程序设置数据通道（即创建 SCTP 关联）。 每次对 CreateDataChannel（）函数的后续调用只是在现有 SCTP 关联中创建一个新的 DataChannel。

7. 一个简单的例子

Alice 和 Bob 使用常见的呼叫服务。 要进行通信，它们必须同时连接到实现该服务的 Web 服务器。 实际上，当 Bob（或 Alice）将他的浏览器指向呼叫服务网页时，他将下载 HTML 页面和保持浏览器通过安全的 HTTP 或 WebSocket 连接连接的 JavaScript。

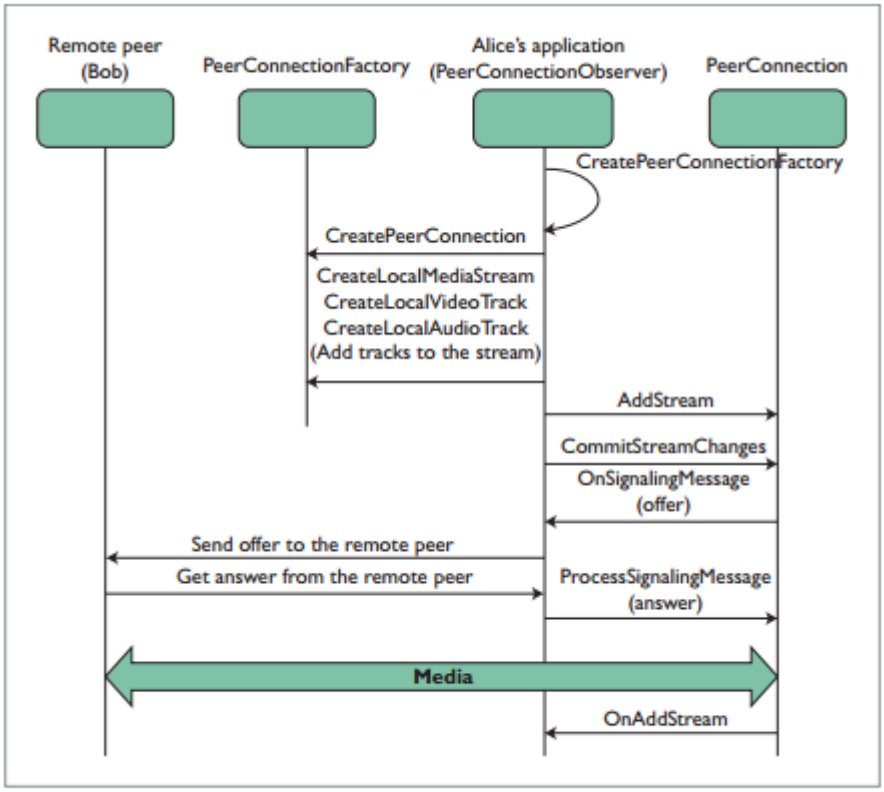

Figure 3. Call setup from Alice's perspective. We can use the WebRTC API to create a direct media connection between Alice's and Bob's browsers.

图 3 表示了与建立实时的，浏览器支持的通信信道相关联的典型呼叫流程。当 Alice 点击网页按钮开始与 Bob 的通话时，JavaScript 实例化一个 PeerConnection 对象。 一旦创建了 PeerConnection，主叫服务端的 JavaScript 代码必须通过 MediaStream 函数设置媒体。 Alice 还必须授予允许呼叫服务访问她的相机和她的麦克风的权限。

在当前的 W3C API 中，一旦应用程序添加了一些流，Alice 的浏览器就会生成一个信号消息。 IETF 尚未定义此消息的确切格式。它必须包含媒体信道信息和 ICE 候选，以及将通信绑定到 Alice 的公钥的指纹属性。然后 Alice 的浏览器将此消息发送到信号服务器（例如，通过 XMLHttpRequest 或 WebSocket）。信号服务器处理来自 Alice 的浏览器的消息，确定这是到 Bob 的呼叫，并且向 Bob 的浏览

器发送信号消息。 Bob 的浏览器上的 JavaScript 处理传入的消息并提醒 Bob。如果 Bob 决定应答呼叫，则在他的浏览器中运行的 JavaScript 将实例化与来自 Alice 的消息相关的 PeerConnection。然后，将进行类似于 Alice 的浏览器上的过程。Bob 的浏览器验证呼叫服务被批准并且媒体流被创建；之后，它创建包含媒体信息和 ICE 候选的信号消息，并且经由信号服务向 Alice 返回指纹。

8. 拥塞控制

针对交互式媒体和数据传输专门构想的拥塞控制机制的讨论尚处于早期阶段--IETF 甚至尚未决定是否开始工作。 该想法是紧密地结合流的拥塞控制，理想地仅对于所有 WebRTC 传送（即音频，视频或数据）仅使用单个拥塞控制实例。同时，对 WebRTC 传输包的不同部分进行优先级排序应该是可能的。

9. 安全考虑

正如我们所描述的，RTCWeb 方法和相关的架构绝对是通讯行业的一个新挑战，因为它们允许 Web 浏览器与服务器端进行少量直接交流或不进行交流。 这是非常具有挑战性的，因为它要求我们考虑几个问题，其中信任和安全问题是根本。 关于这最后一点，当我们允许直接的浏览器到浏览器通信时，一组新的潜在安全威胁变得突出。当前的基本 Web 安全策略基于隔离（也称为沙盒）原则，该原则允许用户保护其计算机免受恶意脚本和跨站点内容引用。

使用此模型，安全策略主要涉及在浏览器中运行的 JavaScript 代码，其实际上充当受访网站的可信计算基础。 将范围扩大到浏览器到浏览器的通信揭开了一般安全问题的新方面。

首先，我们必须考虑通信安全，其方式与我们允许任何两个端点之间的直接通信的其他网络协议（例如 SIP）一样 - 除非我们设想存在作为透明中介的中继。

第二，如果我们让浏览器直接相互通信，那么在实际数据交换阶段开始之前，必须存在允许用户验证同意的机制。 同意验证不应依赖于可信服务器的存在，因此应当由旨在发起与潜在对等体通信的浏览器直接实施。 最后但并非最不重要的是，通过 Web 的 RTC 场景清楚地要求浏览器与它所驻留的节点进行深层交互。例如，在与目标对等体进行多媒体呼叫之前访问本地音频和视频设备。我们必须定义访问策略，这可能涉及某种形式的用户同意，这也让我们面对了几个新的挑战。

10. 总结与展望

RTCWeb 的愿景是标准化一个开放式框架，使得浏览器到浏览器的多媒体应用程序沿着最直接的路径，而无需安装插件。 两个主要的浏览器厂商已经在开发者版本中提供了框架的早期实现。 然而，两者都不完全符合标准，即使他们被期望很快做到。

在不久的未来，标准化活动会集中注意在拥塞控制、音视频编码解码器的选择、远程呈现、数据通道增强。

[参考文献]

[1] H. Sinnreich and W. Wimmreuter, “Communications on the Web,” Elektrotechnik und Informationstechnik, vol. 127, 2010, pp. 187-194; http://dx.doi.org/10.1007/ s00502-010-0742-1.

[2] H. Sinnreich and A. Johnston, “Sip APIs for Communications on the Web,” IETF Internet draft, June 2010.

[3] J. Uberti and C. Jennings, “Javascript Session Establishment Protocol” IETF Internet draft, June 2012.

[4] A. Bergkvist et al., “WebRTC 1.0: RealTime Communication between Browsers,” W3C working draft 09, Feb. 2012; www.w3.org/TR/webrtc/.

[5] C. Holmberg and H. Alvestrand, “Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers,” IETF Internet draft, Feb. 2012.

[6] R. Jesup, S. Loreto, and M. Tuexen, “RTCWeb Datagram Connection,” IETF Internet draft, Mar. 2012.