Pear & WebRTC

Pear Limited

演讲者：谢庭　　梨享计算前端高级工程师

邮箱：86755838@qq.com

CONTENTS

背景

WebRTC发展历程及现状

WebRTC

详解WebRTC

P2P流媒体

详解基于WebRTC的**P2P**流媒体

WebRTC与雾计算

有哪些有趣的交融

PART 1

WebRTC背景

- WebRTC的诞生背景
- Web通信的演化历史
- Pear与WebRTC
- 浏览器支持情况

WebRTC的诞生背景

- 基于FaceTime和Skype等视频通话工具，用户可以很方便地与他人进行视频对话

- 开发者们为了将用户体验优化到极致，通过大量的技术手段保障视频质量

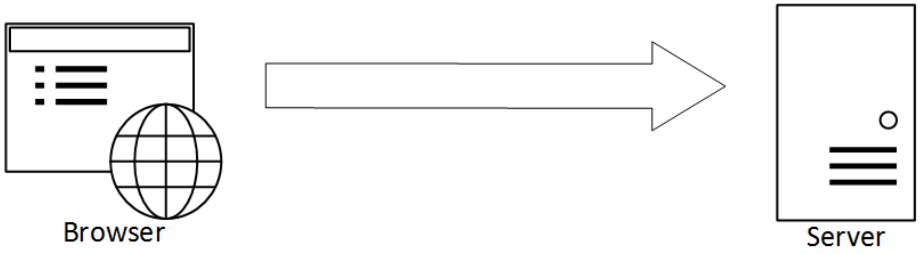
- 专利持有公司向开发者征收授权费，并构筑起巨大的技术壁垒

- 上述解决方案需要用户安装插件或应用程序，增加了使用门槛和安全风险

Web通信的演化历史

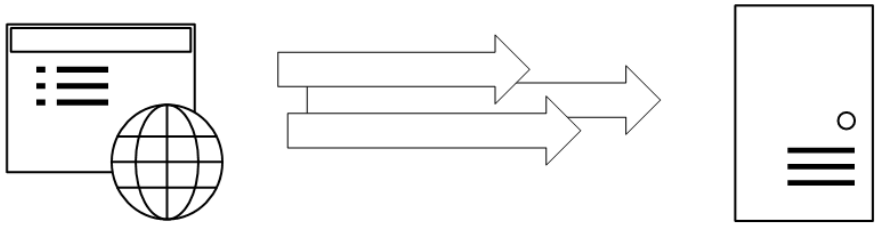
HTTP (Pre AJAX)

最初的Web，一次请求返回一个页面 (如：Yahoo!).



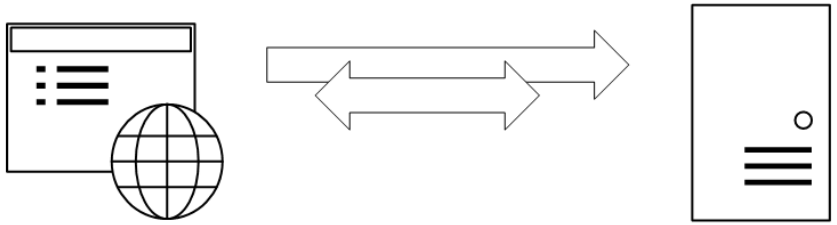
AJAX (2005→)

页面可以无刷新更新内容 (如：Google Maps, GMail).



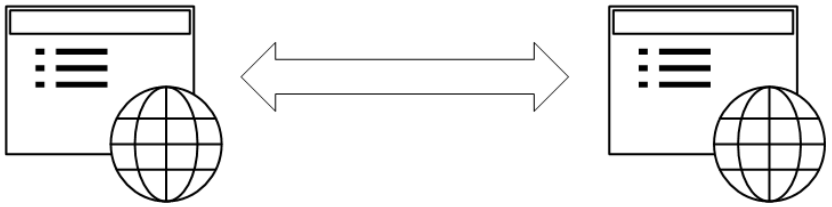
WebSocket (2011→)

页面可以和服务器建立双工通信 (如：Stack OverFlow).

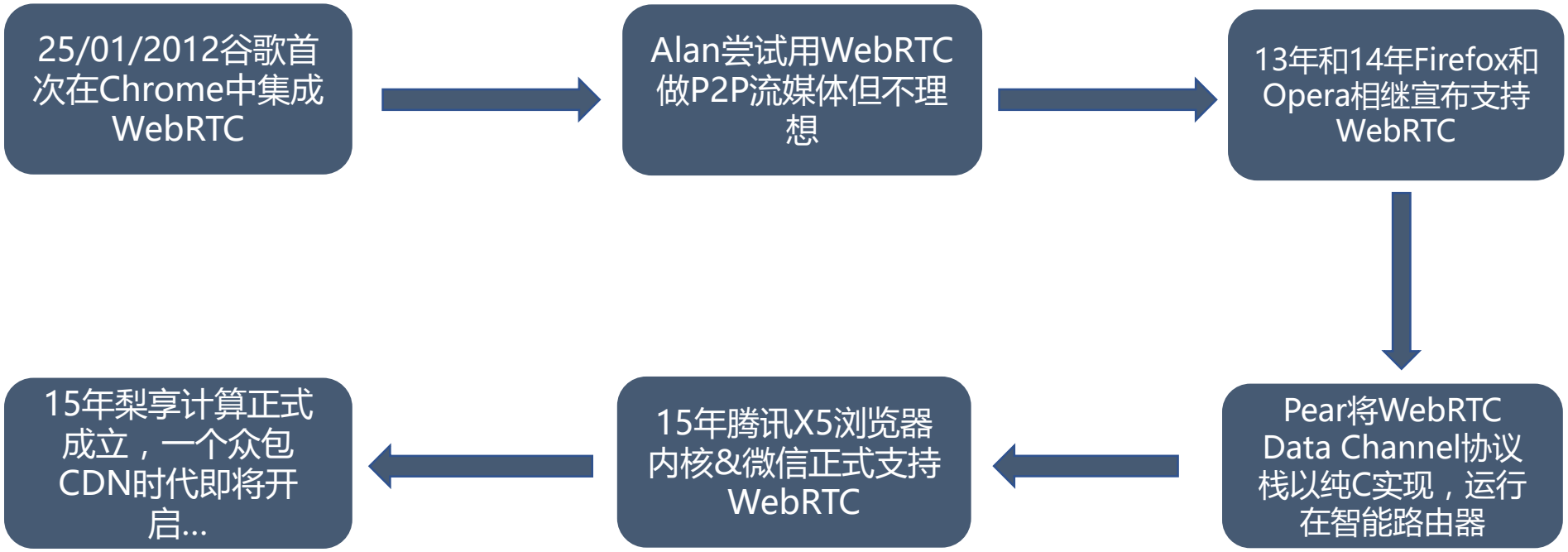


WebRTC (2014→)

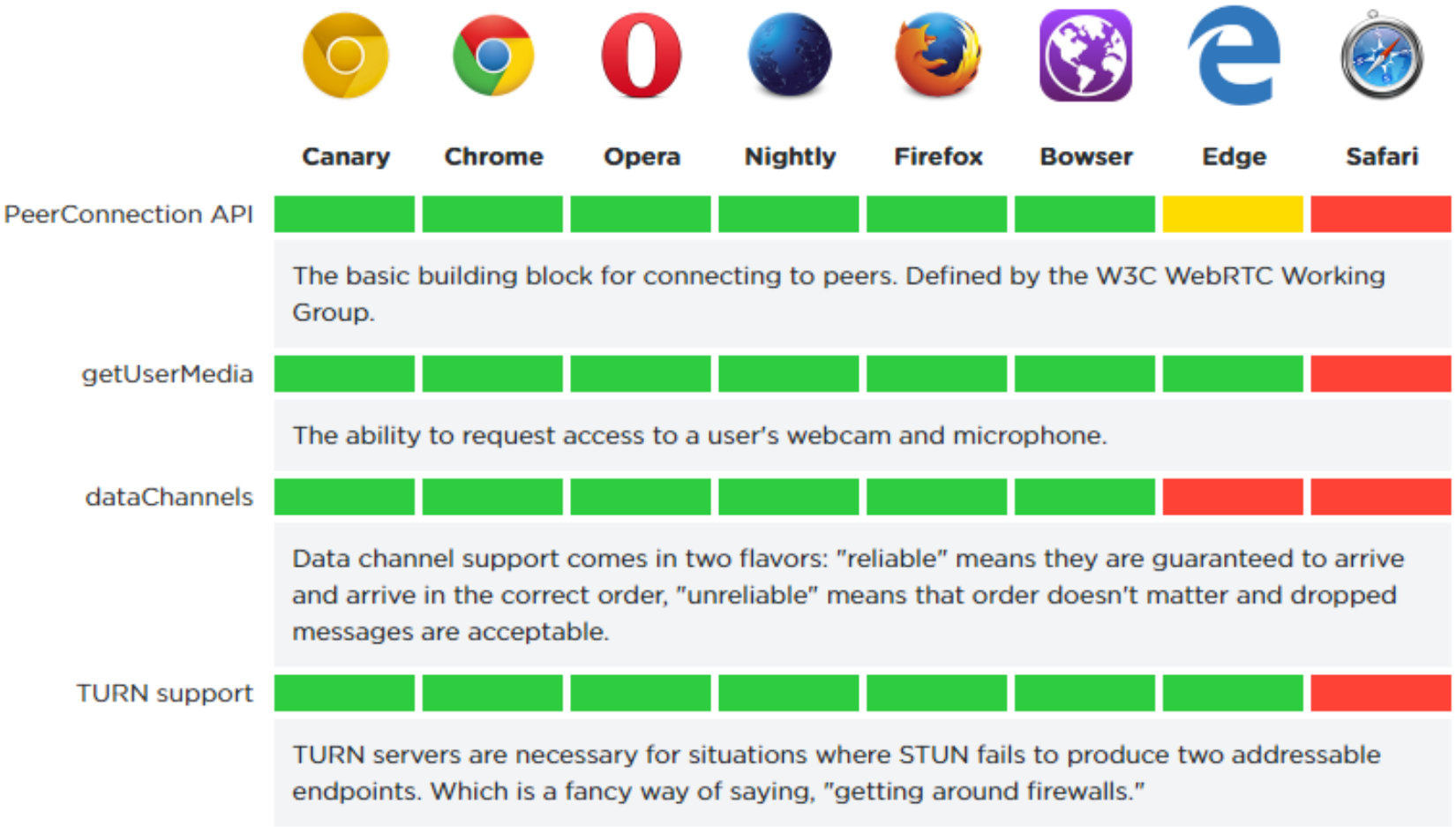
直接的页面对页面通信



Pear与WebRTC



WebRTC浏览器支持情况



Safari11即将支持WebRTC

Guides and Sample Code

Media

- **New in Safari 11.0 – Support for real-time communication using WebRTC.**
- **New in Safari 11.0 – Can**
  - Added support for t
  - Websites can access

WebRTC In Development

An API to facilitate real-time communication for browser-to-browser applications.

Reference: http://www.w3.org/TR/webrtc/
Contact: @jonathandavis - Jon Davis

Web APIs

- **New in Safari 11.0 – WebA**
  - Added support for W

Jonathan Davis 转推了

Ricky Mondello @rmondello · 18小时
回复 @rmondello
Safari 11 is huge. WebRTC, WebAssembly, Intelligent Tracking Prevention, Auto-Play blocking, and more.
developer.apple.com/library/conten…

13 126 202

PART 3

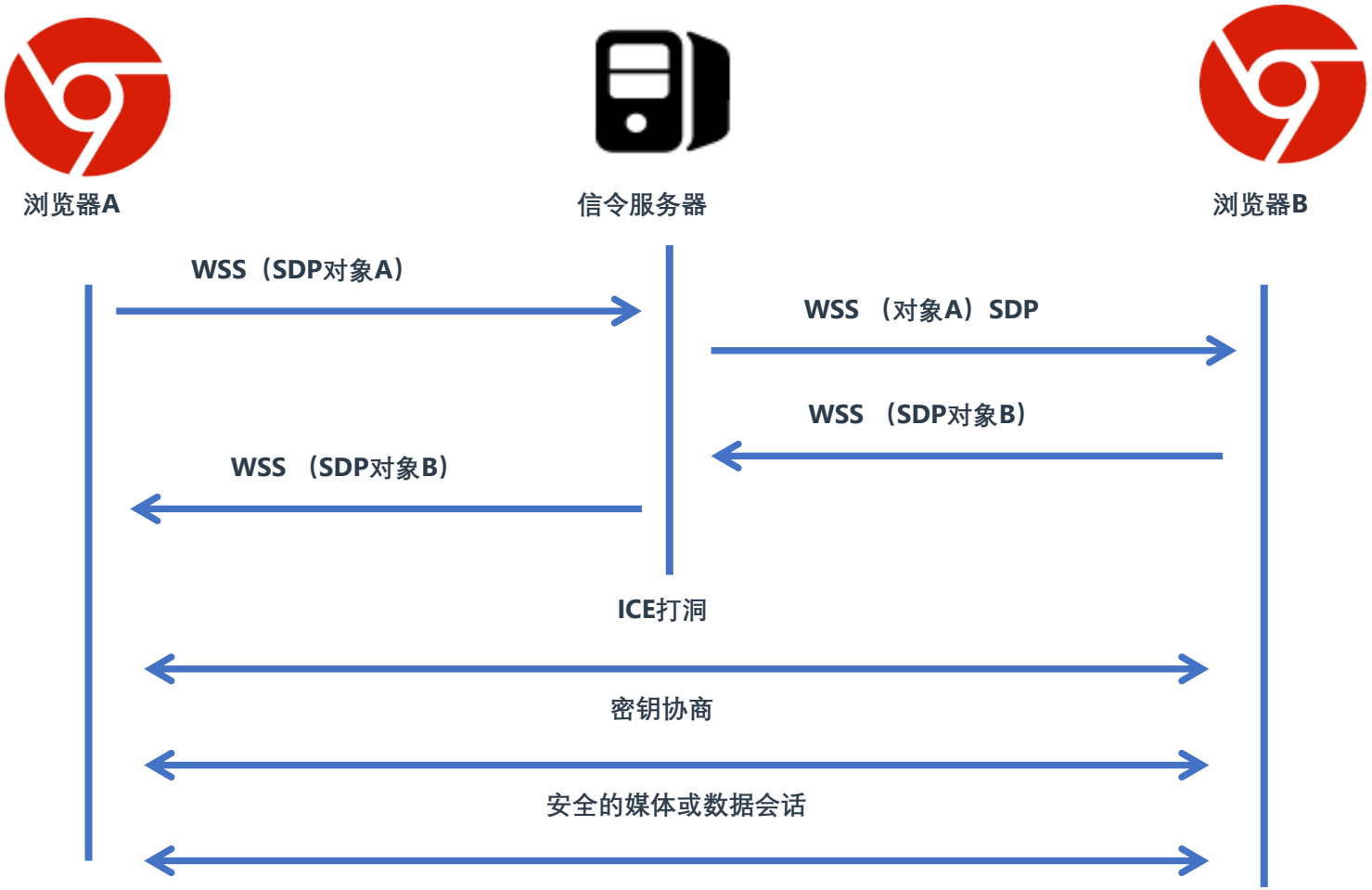
WebRTC

- 建立WebRTC媒体会话
- 典型会话流程
- WebRTC打洞原理
- WebRTC数据通道
- simple-peer
- 对WebRTC标准的持续跟进

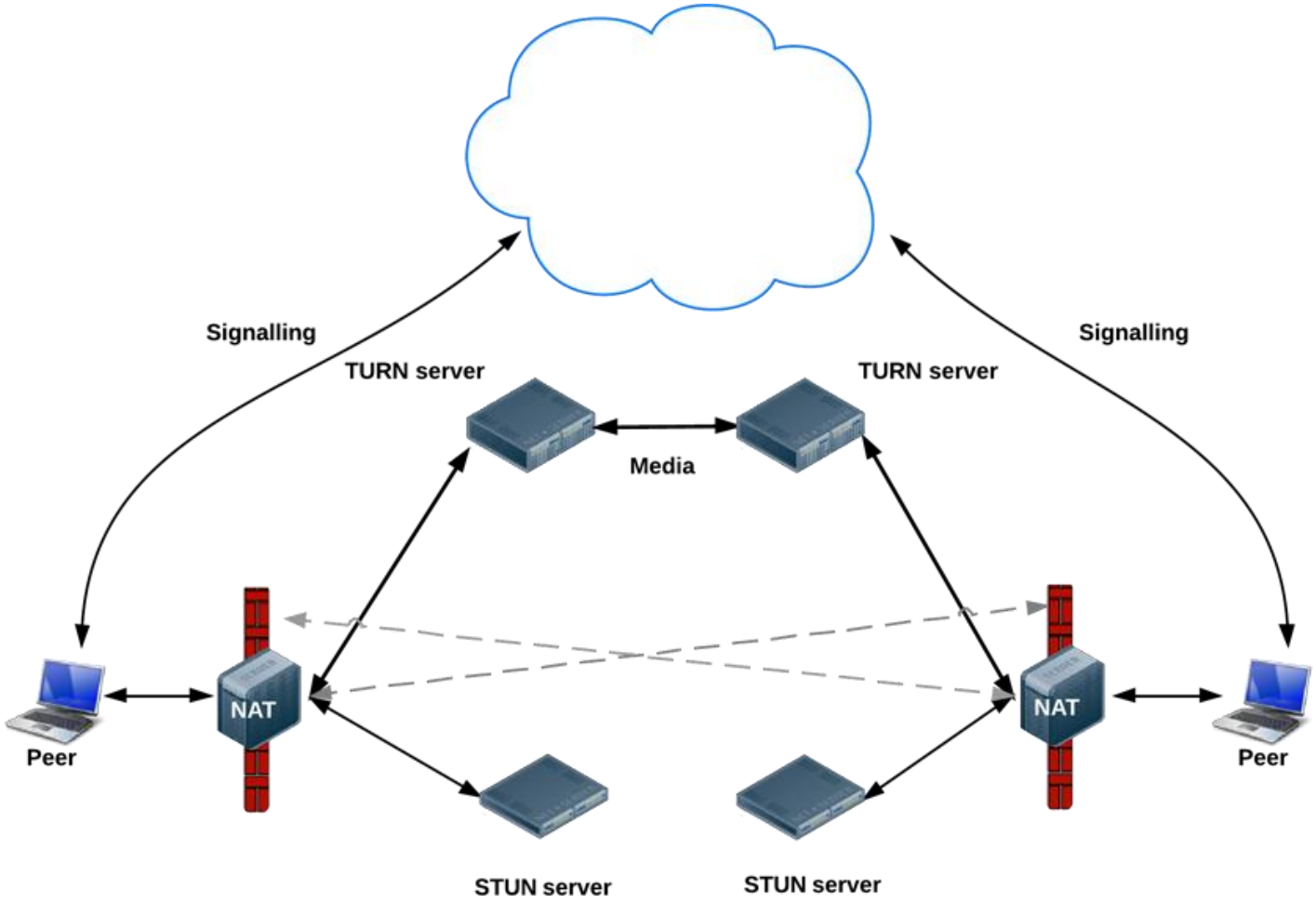
建立WebRTC媒体会话

- 获取本地媒体（通过getUserMedia）
- 建立对等连接（RTCPeerConnection对象）
- 交换媒体或数据
- 关闭连接

典型的WebRTC会话流程



WebRTC打洞原理


Signalling
TURN server
TURN server
Signalling
Media
NAT
NAT
Peer
Peer
STUN server
STUN server

WebRTC数据通道

- 数据通道：Data Channel，是WebRTC在浏览器之间建立的一种非媒体交换连接，相比WebSocket和HTTP，数据通道支持流量大、延迟低的连接，既稳定可靠，又不失灵活性
- API非常简单，类似WebSocket，通过send方法发送数据，onmessage方法接收数据

发起端：

```
var peerConnection = new RTCPeerConnection();

var dataChannel = peerConnection .createDataChannel("myLabel", dataChannelOptions);
```

对等端：

```
var peerConnection = new RTCPeerConnection();

var peerConnection.ondatachannel = function(e) {
    dc = e.channel
};
```

数据通道配置

- dataChannelOptions是一个普通的Javascript对象，开发人员可以通过对其进行配置来让应用在UDP或者TCP的优势之间进行切换，比如让数据传输得更加稳定可靠，或者传输得更快（鱼与熊掌不可兼得）：

- ordered：设置数据的接受是否需要按照发送时的顺序
- maxRetransmitTime：设置数据发送失败时，多久重新发送
- maxRetransmits：设置数据发送失败时，最多重发次数

- 主要是配置ordered，当设置为true时数据通道表现更像TCP，false时表现更像UDP。

simple-peer

- 原生WebRTC API对开发者还是不够友好，而且开发者还要解决浏览器兼容问题
- simple-peer是对WebRTC API的封装，支持video/voice和data channel
- 已经被多个项目使用，稳定可靠

simple-peer 导入与兼容性检测

- 通过browserify导入或者直接script标签引入simplepeer.min.js
- 首先检测当前浏览器是否支持：

```
var Peer = require('simple-peer')

if (Peer.WEBRTC_SUPPORT) {
  // webrtc support!
} else {
  // fallback
}
```

simple-peer video/voice

```
var SimplePeer = require('simple-peer')

// get video/voice stream
navigator.getUserMedia({ video: true, audio: true }, gotMedia, function () {})

function gotMedia (stream) {
  var peer1 = new SimplePeer({ initiator: true, stream: stream })
  var peer2 = new SimplePeer()

  peer1.on('signal', function (data) {
    peer2.signal(data)
  })

  peer2.on('signal', function (data) {
    peer1.signal(data)
  })

  peer2.on('stream', function (stream) {
    // got remote video stream, now let's show it in a video tag
    var video = document.querySelector('video')
    video.src = window.URL.createObjectURL(stream)
    video.play()
  })
}
```

simple-peer data channel

```
var SimplePeer = require('simple-peer')

var peer1 = new SimplePeer({ initiator: true })
var peer2 = new SimplePeer()

peer1.on('signal', function (data) {
  // when peer1 has signaling data, give it to peer2 somehow
  peer2.signal(data)
})

peer2.on('signal', function (data) {
  // when peer2 has signaling data, give it to peer1 somehow
  peer1.signal(data)
})

peer1.on('connect', function () {
  // wait for 'connect' event before using the data channel
  peer1.send('hey peer2, how is it going?')
})

peer2.on('data', function (data) {
  // got a data channel message
  console.log('got a message from peer1: ' + data)
})
```

对WebRTC标准的持续跟进

cheedoong (Reporter)
Description • a year ago

```
What did you do?
================
1. We are building our own WebRTC protocol stack.
2. Found there are different interoperation behaviors/problems our stack <--> Chrome vs. our stack <--> Firefox.
3. Look into the details, we found Firefox's implementation is not in compliance with an RFC document.
 ...

What should have happened?
==========================
SDP offer generated for onIceCandidate(e):
s=SIP Call
, where it should be
s=-

 ...
```

Eric Rescorla (:ekr)
Comment 2 • a year ago

Why do you believe that s= MUST be "-"? RFC 4566 simply requires it to be textual and non-empty:
5.3. Session Name ("s=")

...
Here's what RFC 3264 says:

...
The document you cite is informative, not normative, so as far as I can tell if you are refusing to accept this s= line you have a defect in your code.

对WebRTC标准的持续跟进

cheedoong (Reporter)
Comment 4 • a year ago

Agree with Erk.

It is informative, not normative. Besides I think it's suggestive.
After all, semantically it is not suitable to be counted as an SIP call, if the SDP is generated for a PeerConnection serving a DataChannel.

I suggest change "s=SIP Call" to "s=-" in this scenario (serving DataChannel), as Chrome dose.

Eric Rescorla (:ekr)
Comment 5 • a year ago

I agree that it would be slightly better to have a "-" and we'd be happy to take a patch on that, I suspect.

Randell Jesup [:jesup]
Comment 6 • a year ago

Ok, I just tried this and have s=- for a/v calls and DataChannel calls....

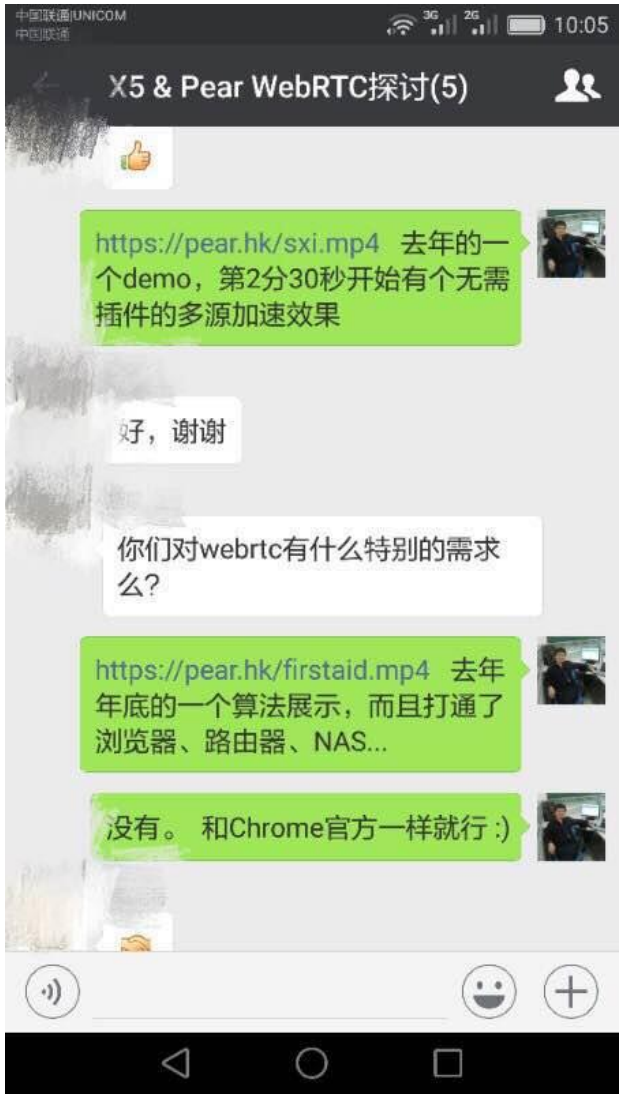
Looking at the image you uploaded, it shows o=MOZILLA-SIPUA-28.0. This means it's Firefox 28.0 ... current version is Firefox 45(!).

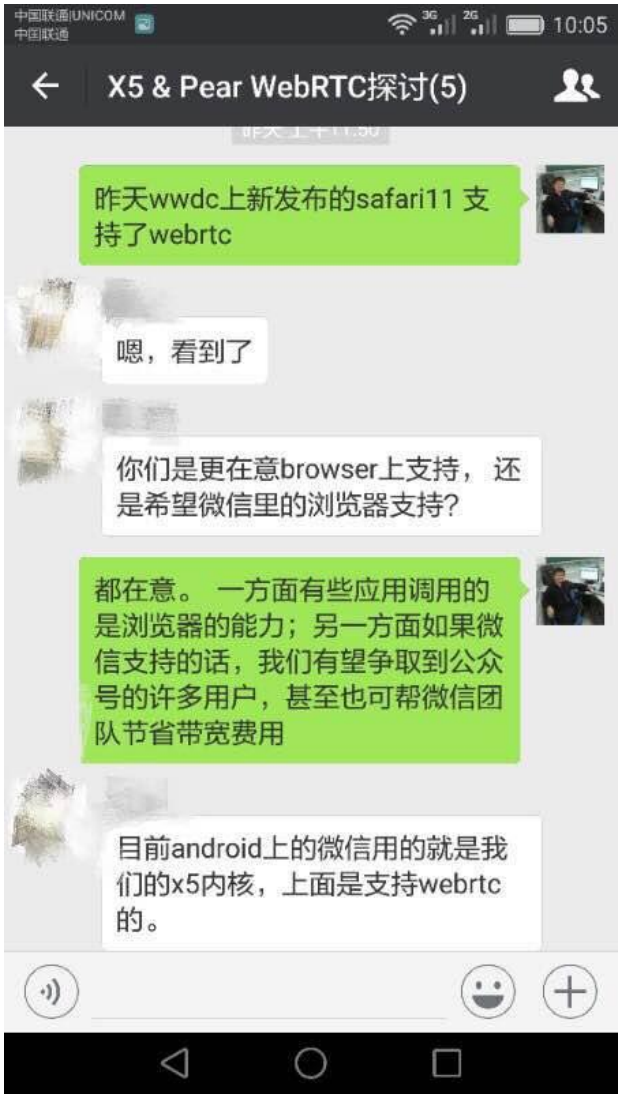
Please check where/what you're testing against. Anyone running 28.0 is running something multiple years out-of-date.

cheedoong (Reporter)
Comment 7 • a year ago

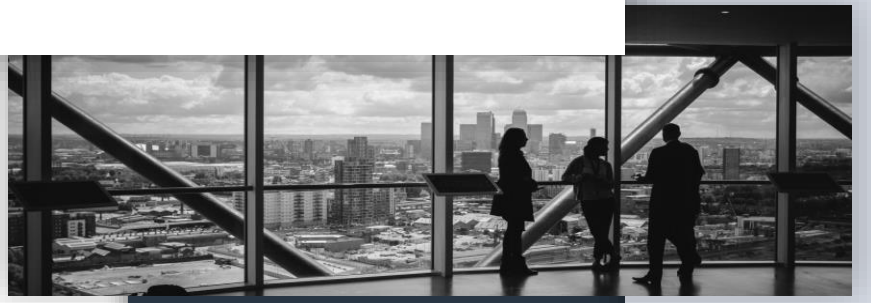
John, ekr, Randell,
thank you all for the quick response!

对WebRTC标准的持续跟进





PART 4



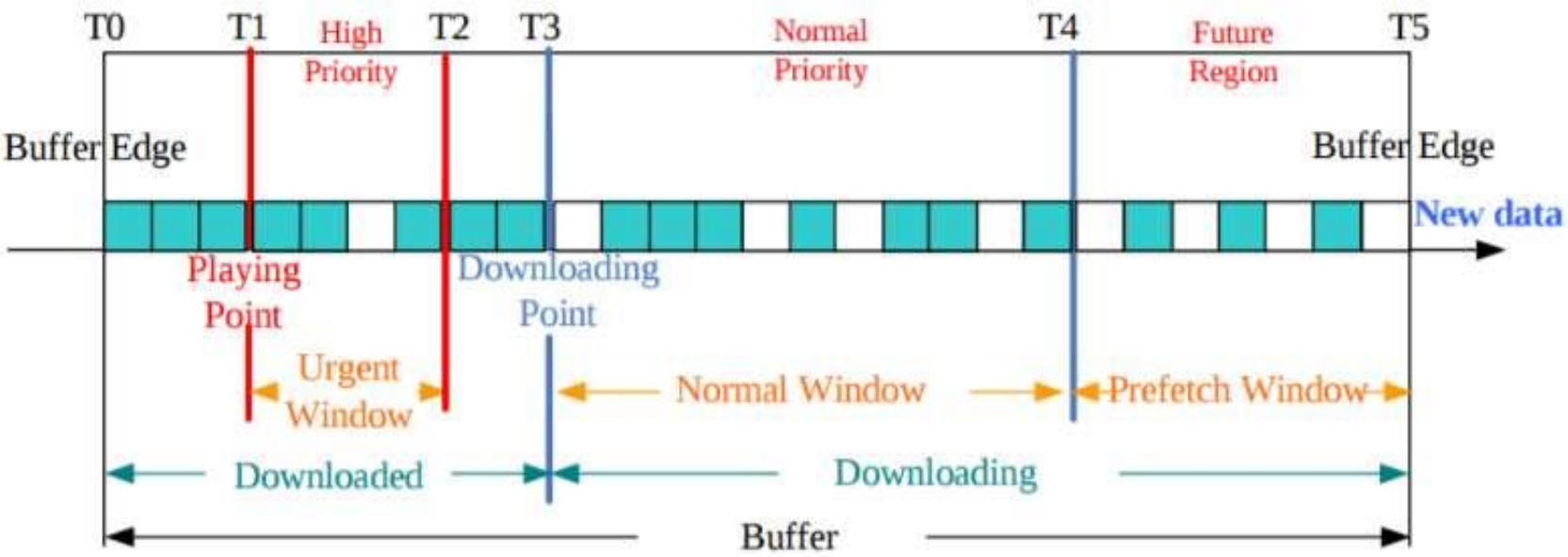
P2P流媒体

- WebTorrent
- First-Aid
- FastMesh
- Push-Pull
- PearPlayer.js

WebTorrent—pull-based的P2P算法

- WebTorrent是用于 Node.js 和浏览器的流 Torrent 客户端，完全使用 JavaScript 编写。
- WebTorrent 是个轻量级，快速的开源 BT 客户端，拥有非常棒的用户体验。
- 在浏览器中，WebTorrent 使用 WebRTC (数据通道)进行点对点的传输，无需任何浏 览器插件，扩展或者安装。
- 支持 Chrome, Firefox 和 Opera
- 存在“带宽饥饿”和“内容饥饿”问题

First-Aid算法——pull-based的P2P算法增强版



FastMesh——push-based的P2P算法

- push-based，延迟低，适用于直播
- 能根据每个节点的上行带宽动态调整网络拓扑结构，充分利用了网络的现有能力
- 根据一项对比试验，FastMesh可能是目前众多P2P算法中效果最好的
- 缺点：对于网络结构的变化比较敏感，不适合节点变化较大的情况

Push-Pull算法——综合两大优势

- 用pull的方式从父节点获取优先级最高的buffer，由父节点以push的方式为其提供后续的buffer
- 根据新加入节点的上行带宽和服务能力来动态调整拓扑结构
- 混合HTTP、WebRTC、WebSocket等多种协议，在优先保证用户体验的前提下最大化P2P率
- 具备低延迟、高带宽利用率、高P2P率、对网络拓扑结构变化鲁棒性强等优势

PearPlayer

✓ WebRTC | https://qq.webrtc.win/tv/pear001.mp4 | 播放

Fog Speed 0 Kbps
Cloud Speed 0 Kbps

96 %

(58 M)

Fog Ratio

Algorithm	FirstAid
ChunkSize	1 M
Interval	5000 ms
Timeout	5000 ms
SlideInterval	10 s

详细信息

Undownloaded Server Node DataChannel

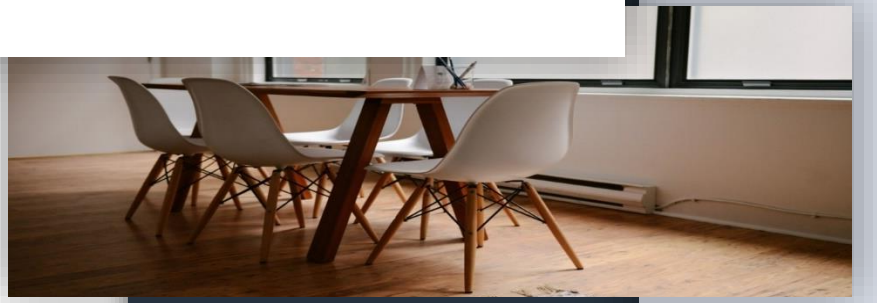
PearPlayer.js API

```
<video id="v1" src="https://example.com/v1.mp4"></video>
<script src="PearPlayer.js"></script>
<script>
    var PearPlayer = require('PearPlayer');
    PearPlayer('#v1',{
        type: 'mp4',
        token: token
    });
</script>
```

PART 5

WebRTC与雾计算

Easing your life



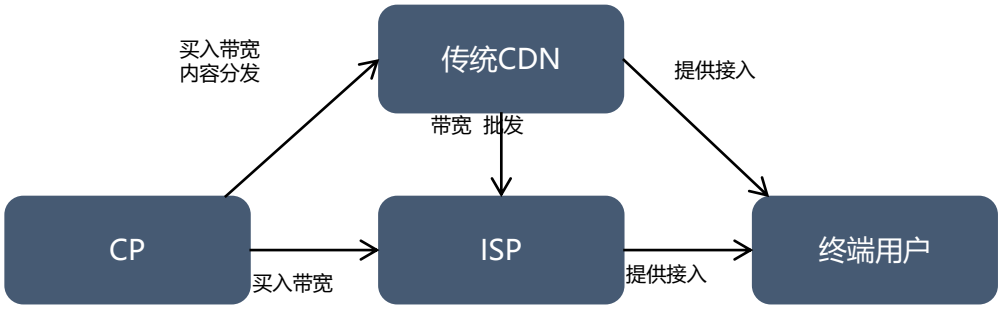
与云计算的区别

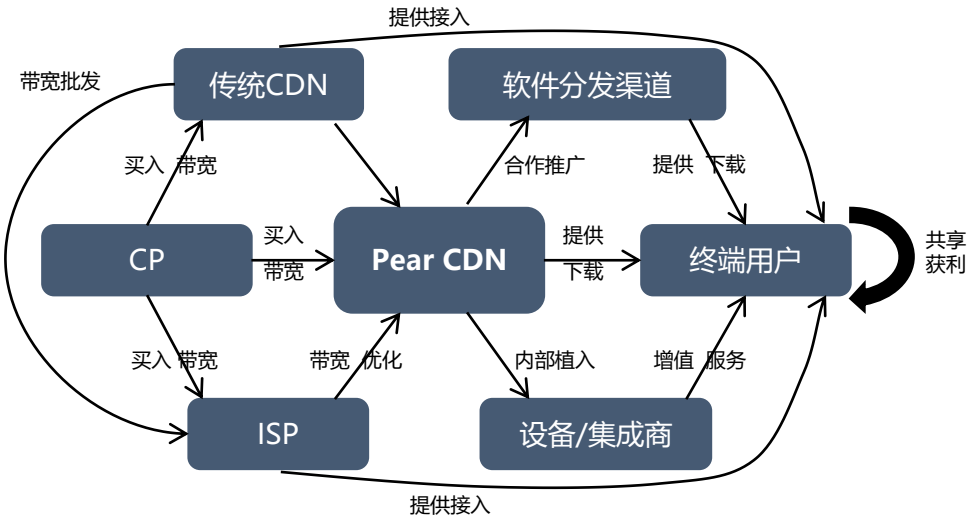
- 云在天空飘浮，高高在上，遥不可及；数据中心距离终端用户较远，用户消息需要经过若干跳才能够到达
- 而雾却现实可及，贴近地面，就在你我身边
- 雾计算并非由性能强大的服务器组成，而是由性能较弱、更为分散的各类计算设备组成，例如智能路由器、网络存储设备等
- 雾能够弥补云的不足，并和云相互配合，协同工作。

Pear——雾计算的践行者

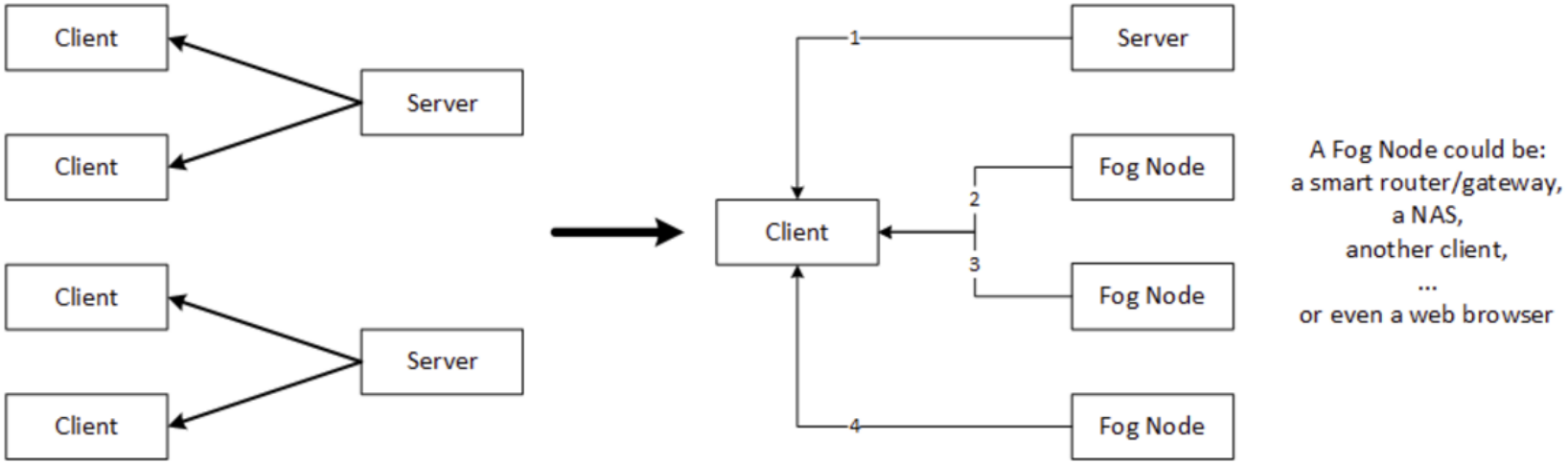
- 拥有海量可持续稳定提供服务的节点。
- 大部分带宽、存储、计算资源通过众包方式收集自终端用户稳定在线的边缘设备。
- 服务能力覆盖全部地域、所有运营商、每处网络边缘。
- 动态、实时的感知和调度，让数据传输距离尽可能接近“零跳”。

站在共享经济的风口上


买入带宽
内容分发
传统CDN
提供接入
带宽 批发
CP
ISP
终端用户
买入带宽
提供接入


提供接入
带宽批发
传统CDN
软件分发渠道
买入 带宽
合作推广
提供 下载
CP
买入
带宽
Pear CDN
提供
下载
终端用户
共享
获利
买入 带宽
带宽 优化
内部植入
增值 服务
ISP
设备/集成商
提供接入

多源，鲁棒


Client
Server
Client
Client
Server
Client
Traditional CDN
(single source & single stream)
1
Server
Fog Node
2
Client
3
Fog Node
4
Fog Node
Pear FogVDN
(multiple sources & multiple channels)
A Fog Node could be:
a smart router/gateway,
a NAS,
another client,
...
or even a web browser

低成本

- 相对于传统CDN依赖于昂贵的机房、服务器和租赁带宽，Pear FogVDN凭借海量的节点和共享经济的模式，价格相比传统CDN显著降低。

捍卫用户的隐私

支持整个传输链路的安全存储与传输，防止数据被篡改，避免版权文件的泄漏及内容劫持等。

- 所有边缘节点支持TLS，HTTP默认使用HTTPS(HTTP2.0)通道
- WebRTC通道传输数据使用SCTP协议和DTLS加密来保护
- 信令通信是通过安全的WebSocket完成的，该WebSocket也使用TLS/SSL加密
- 可选的数据加密，与分段、分片、分块

支持开放的、国际标准的协议

- HTTP2
- WebRTC
- HLS
- DASH

智能内容分发和调度

- 精确解析

精确识别访问用户的真实IP地址（即使在使用VPN的情况下），并99.9%地映射到所处区域、ISP，筛选出距离近乎“零跳”的节点为用户提供优质服务。

- 高效内容分发与分布网络

精确、实时感知数据热度，把内容快速Push到边缘节点。

- 大数据推荐、预测，及内容预取

提供API和相关服务，对于有用户画像和资源属性分析能力的CP，内容预取和传播可精确到用户和“小世界”(Small-world)社交网络粒度。

THANKS

Pear Limited

https://pea.hk