DevCon5

HTML5 & Mobile App Developer Conference

December 10-11, 2013
Hilton Los Angeles/Universal City, California

www.devconfive.com

@DevConFive
#DevCon5

WebRTC & WebSockets

Peter Dunkley, Technical Director, Crocodile RCS Ltd

Email: peter.dunkley@crocodilertc.net
Twitter: @pdunkley

Evolution on the web

Sir Tim Berners-Lee creates HTML. Web -pages are static

W3C produces the DOM1 specification

WebSocket and WebRTC implementations become available

1990 1996 1998 2004 2011

Microsoft and Netscape introduce different mechanisms for DHTML

Google uses Ajax in Gmail (W3C releases 1st draft in 2006) – the dawn of web-apps

Revolution in telecoms

The revolution

Before today the operators (big and small) had full control over real-time communications because it was hard to do and substantial infrastructure investment was required.

Claude Chappe invented the optical telegraph

First commercial electrical telegraph created by Cooke and Wheatstone

Alexander Graham Bell patents the telephone

Rotary dial enters service

From the 1960s onwards digital exchanges start to appear

1963: DTMF enters service

From the 1990s onwards voice started to be carried on technologies developed for data networks such as ATM and IP

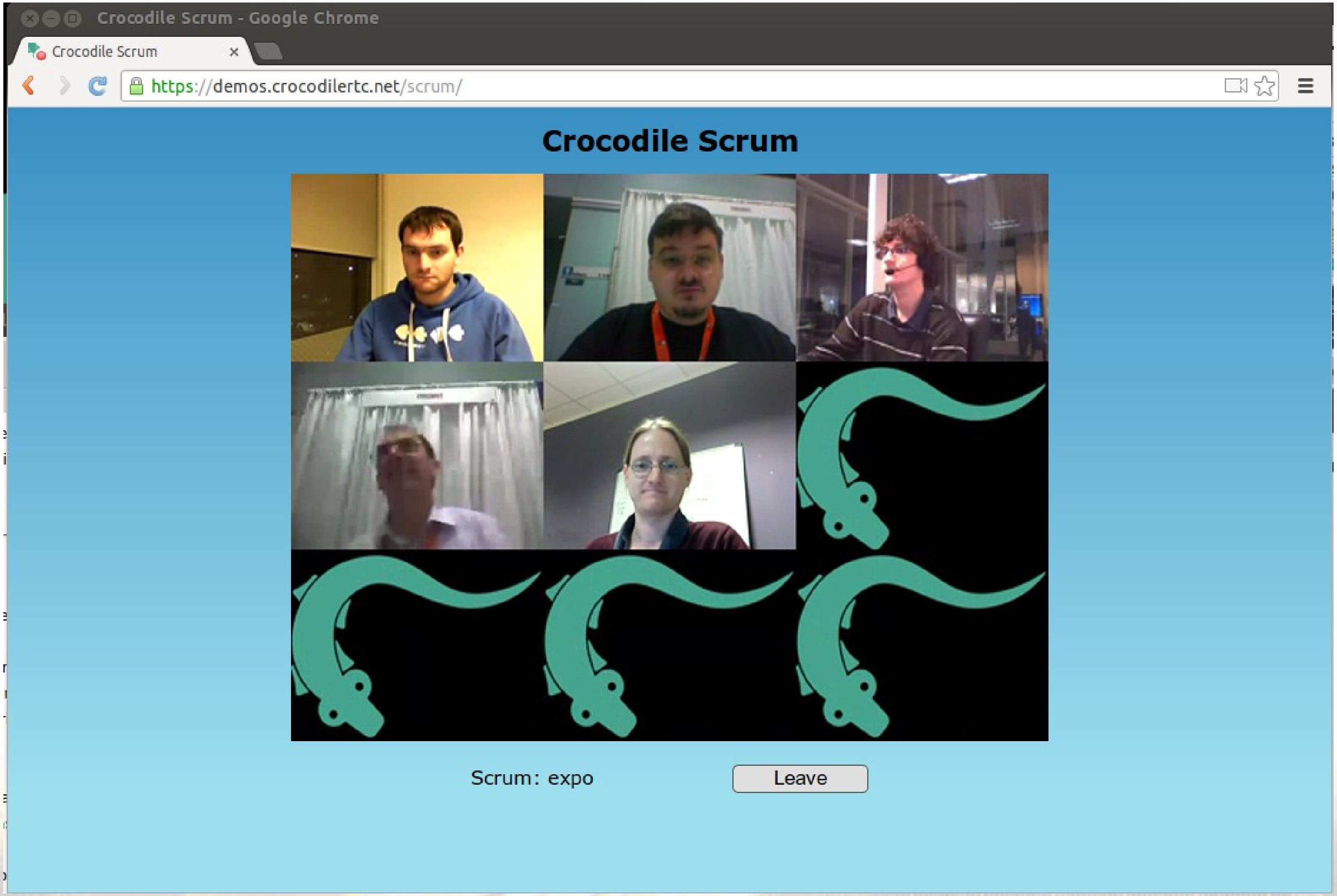
WebSocket and WebRTC implementations become available

1792 1837 1876 1919 1960s > 1990s > 2011

Demo #1: Crocodile Scrum

- Opera and Google Chrome only for now
  - Works on Google Chrome for Android
  - Mozilla Firefox support coming soon
- Anonymous ad-hoc conferencing
- Makes use of WebRTC and WebSockets
- Join the “devcon5” scrum

https://demos.crocodilertc.net/scrum


Crocodile Scrum - Google Chrome
Crocodile Scrum
https://demos.crocodilertc.net/scrum/
Crocodile Scrum
Scrum: expo
Leave

There are a number of proprietary implementations that provide direct interactive rich communication using audio, video, collaboration, games, etc. between two peers' web-browsers. These are not interoperable, as they require non-standard extensions or plugins to work. There is a desire to standardize the basis for such communication so that interoperable communication can be established between any compatible browsers.

Real-Time Communication in WEB-Browsers (rtcweb) 2013-03-13 charter

http://tools.ietf.org/wg/rtcweb/

The mission of the Web Real-Time Communications Working Group, part of the Ubiquitous Web Applications Activity, is to define client-side APIs to enable Real-Time Communications in Web browsers.

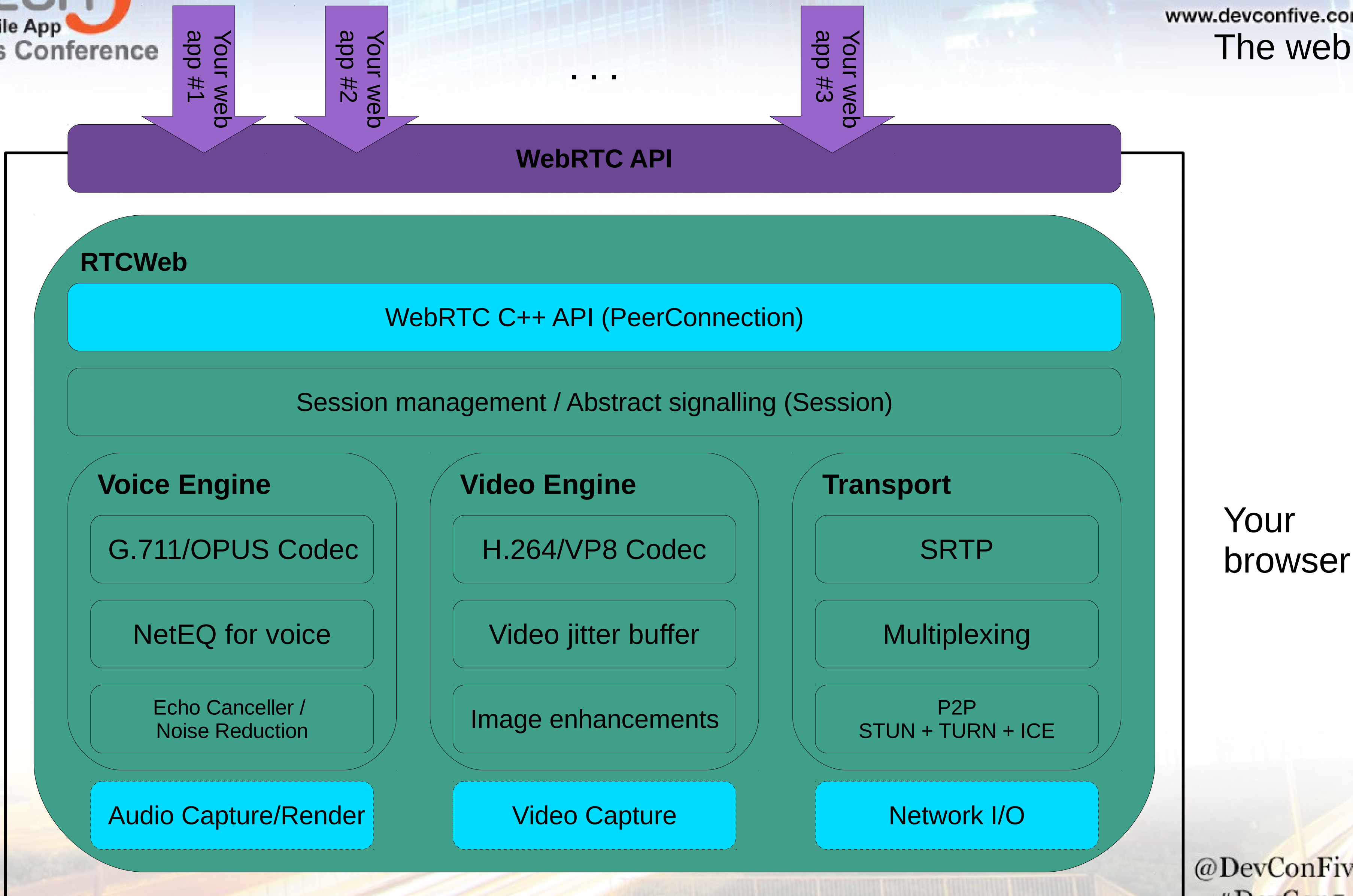
These APIs should enable building applications that can be run inside a browser, requiring no extra downloads or plugins, that allow communication between parties using audio, video and supplementary real-time communication, without having to use intervening servers (unless needed for firewall traversal, or for providing intermediary services).

Web Real-Time Communications Working Group Charter

http://www.w3.org/2011/04/webrtc-charter.html

RTCWeb and WebRTC: not the same thing

- RTCWeb is the on-the-wire protocol as defined by the IETF and may be used in many applications and systems
  - Within VoIP phones
  - On network servers
  - Includes MTI codecs for audio and video
- WebRTC is the browser API as defined by the IETF



Based on the diagram from http://www.webrtc.org/reference/architecture

Audio Codecs

- RTCWeb has two MTI (Mandatory To Implement) audio codecs:
  - G.711 narrowband
    - free to use and unencumbered by patents
    - trivial to implement
    - widely supported on legacy equipment
  - OPUS wideband
    - free to use and unencumbered by patents
    - complicated to implement – but there are open-source versions
    - fantastic quality audio and excellent handling of packet loss

Video Codecs (the great debate)

- There is a big argument over which of H.264 and VP8 should be used
- The arguments are commercial not technical
  - H.264 and VP8 are comparable in terms of performance and quality
  - H.264 and its licensing terms are unacceptable to many small companies and open-source projects
    - Cisco's offer helps some but not all
  - The IPR situation around VP8 is unclear
    - Large (and rich) companies cannot risk using VP8 – it makes them a target
  - Mandating both will not solve this
- Out of desperation older codecs are being suggested including H.261, H.263, and Theora

Codecs are not limited to the MTI

- Apps and browsers can offer any codecs they are capable of
- The MTI is just the base set of codecs you must support to ensure interoperability between endpoints

WebRTC has a rich API

- Media Capture and Streams
  - Audio, video, and screen-sharing
  - http://www.w3.org/TR/mediacapture-streams/
- MediaStream Recording
  - http://www.w3.org/TR/mediastream-recording/
- WebRTC
  - Data can be exchanged too
  - http://www.w3.org/TR/webrtc/

Available (to varying degrees) in Chrome, Firefox, and Opera

What do these APIs let you do?

- Capture audio and video streams from microphone and webcam
- Exchange the captured audio and video with a peer in real-time
- Record local and remote audio and video streams
- Reliably exchange data with a peer in real-time

Screen sharing

- Google Chrome has experimental support for screen sharing
  - You need to turn on a hidden flag to use it
- Screen sharing is (more) dangerous (than video calling)
  - It only takes one frame to reveal any personal details on your screen
- In the future screen sharing will be restricted
  - Only available to Chrome apps (not web-pages)
  - Will require you to explicitly select the screen or window you want to share

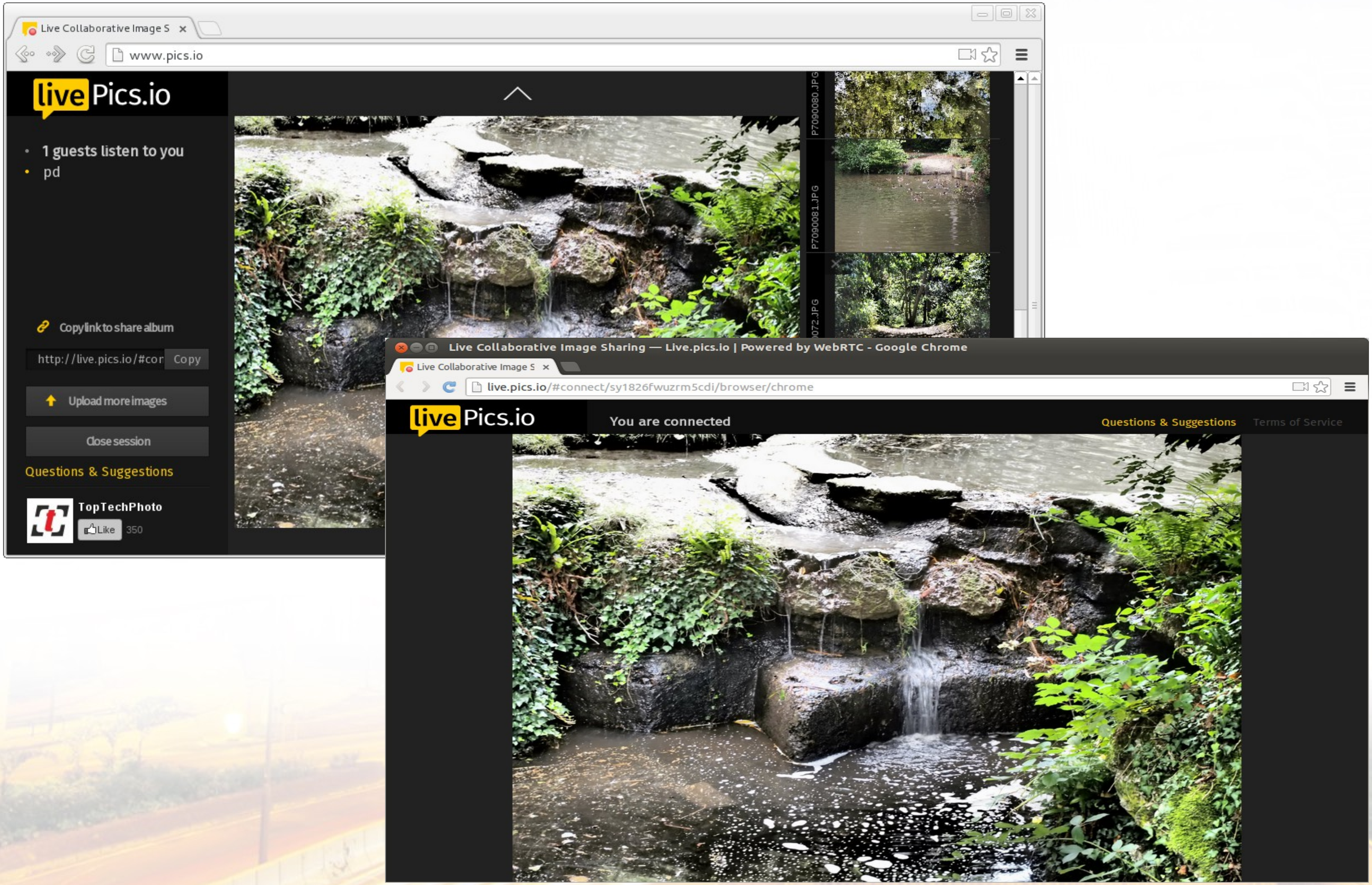
The DataChannel

- The WebRTC DataChannel uses SCTP over DTLS
  - SCTP means reliable, in-order, frame delivery
  - DTLS means UDP packets (so the same NAT traversal mechanisms can be used for audio, video, and data) that are encrypted
- There are already peer-2-peer file-sharing applications implemented using the WebRTC DataChannel

Demo #2: live.pics.io

- Produced by pics.io
- Uses WebRTC DataChannels for “Live Collaborative Image Sharing”
- Drop your images into the browser
- Share the link (to multiple people)
- Talk people through your slide-show

http://live.pics.io/


Live Collaborative Image S
www.pics.io
live Pics.io
1 guests listen to you
pd
Copy link to share album
http://live.pics.io/#con Copy
Upload more images
Close session
Questions & Suggestions
TopTechPhoto
Like 350
P7090080.JPG
P7090081.JPG
Live Collaborative Image Sharing — Live.pics.io | Powered by WebRTC - Google Chrome
Live Collaborative Image S
live.pics.io/#connect/sy1826fwuzrm5cdi/browser/chrome
live Pics.io
You are connected
Questions & Suggestions
Terms of Service

WebRTC applications

- WebRTC is not about making phone calls in a browser – although this is one possible use case
- WebRTC allows you to make communicate in a contextual way
- A phone call is an activity of its own – but that's not how humans communicate face to face
- A phone call is a disruptive (rude) demanding event

WebRTC is about context

- Talk to someone while collaborating on a document
- A better way to access customer services
  - Already authenticated
  - Use a web-form instead of an IVR
- A truly virtual PBX
  - Web-based phone and operator console
- Many gaming and entertainment related applications
  - FPS without centralised servers (DataChannel) and where you can see and hear your opponents
  - Online gambling (for example, poker) where you can see your opponents
  - Online dating, after dinner speaking, and so on
  - It's not necessarily about the real-time audio and video, but they enhance the experience

Demo #3: Cube Slam

- Produced by Google to showcase WebRTC
- WebRTC DataChannel enables multi-player gaming
- WebRTC media enhances the game but is not part of it

https://www.cubeslam.com/


Cube Slam
https://www.cubeslam.com/kemclh
CUBE SLAM
LEVEL 1
YOU
FRIEND
P1
TECHNOLOGY
TERMS & PRIVACY
SOUND ON
51ms
13:34
02/12/2013

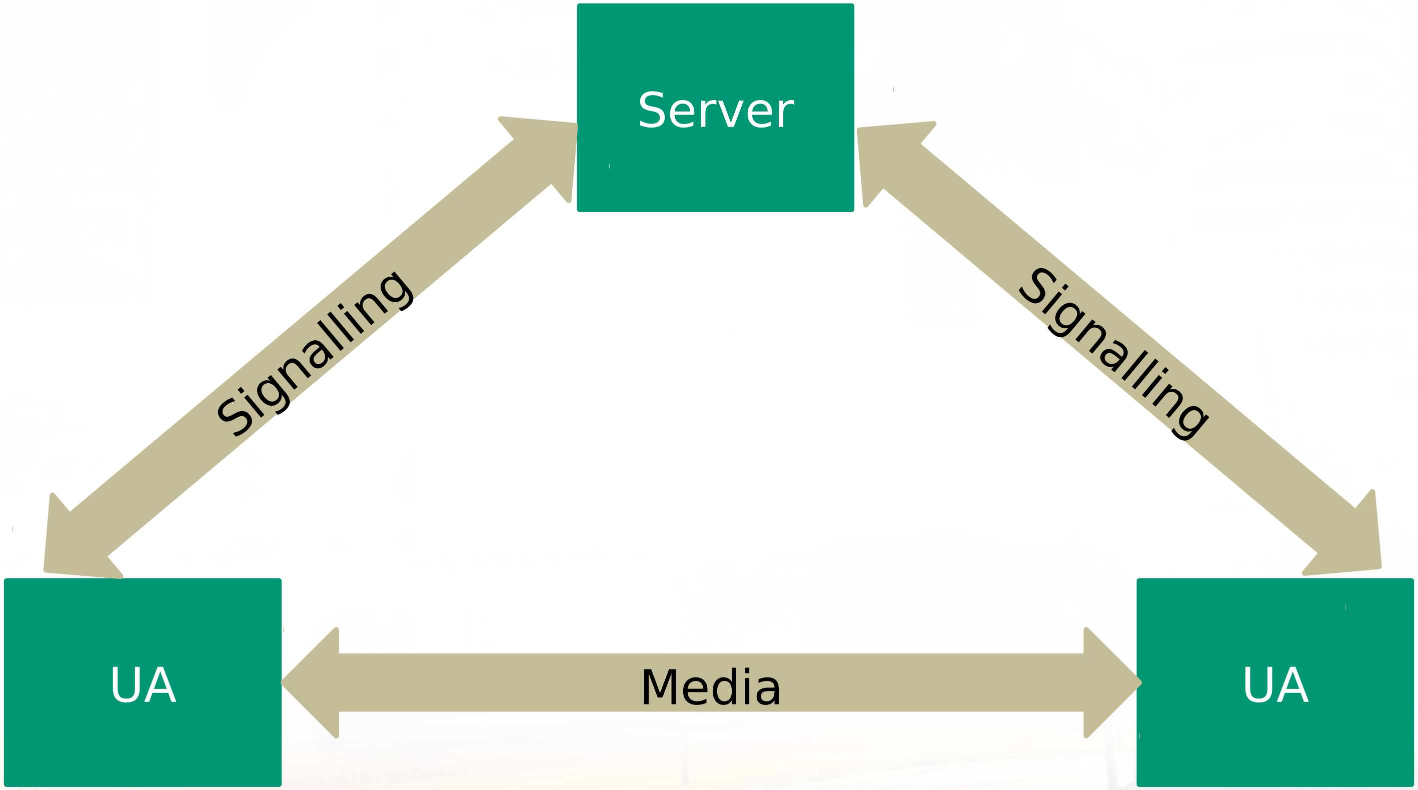
The WebRTC APIs are not enough

- Google made a controversial (but very wise) decision not to specify how the signalling should work
- Signalling is required
  - To discover who to communicate with
  - To exchange information on what the communication should be (audio, data, video, and codecs)
  - Even the simplest, proprietary, RESTful exchange is signalling

Interoperability is not always required

- Interoperability may negatively impact the business case
  - For example:
    - Document collaboration – you want to keep people in your application
    - Online dating – you want to keep people on your site
    - Online gaming – there is no point in different games interoperating

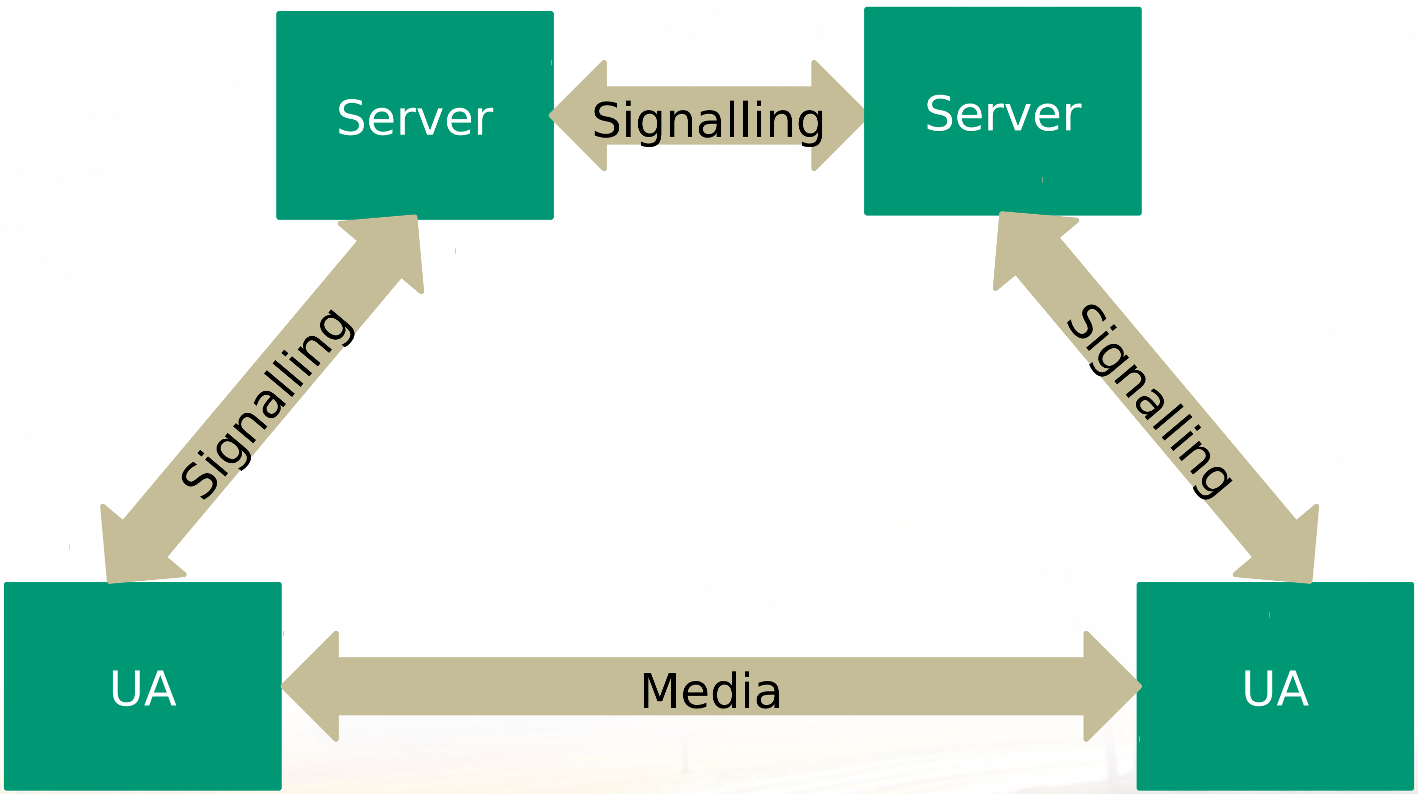
The signalling triangle



Interoperability is sometimes required

- These are typically ones where the point of the application is communication
  - For example:
    - Conferencing – calls in and out of legacy networks are required
    - Call Centres – calls in and out of legacy networks are required
    - Virtual PBX – calls in and out of legacy networks are required

The signalling trapezoid

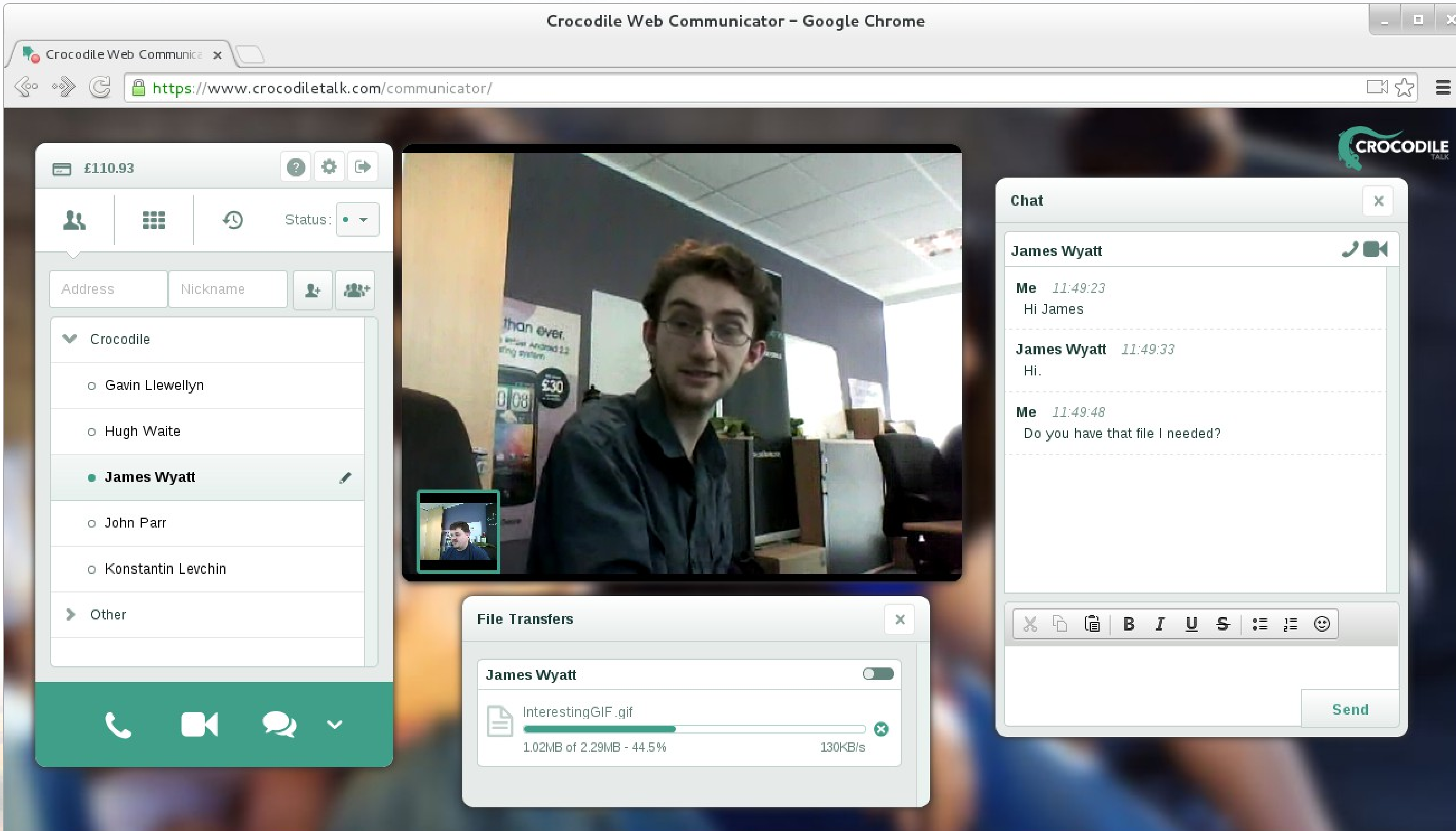


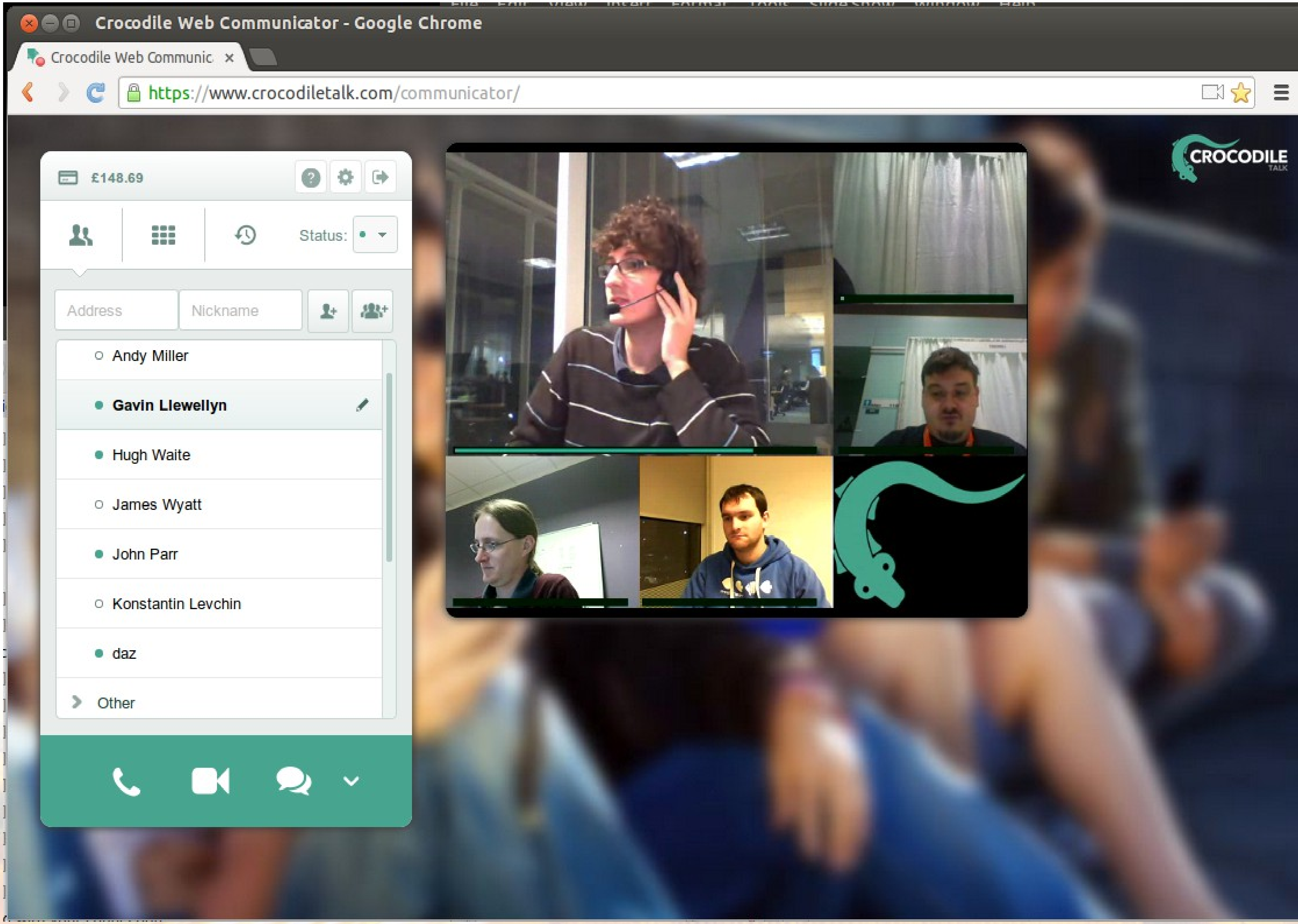
Signalling transport options

- XHR Polling (HTTP Long Polling)
  - Easy
  - Securable
  - Inefficient
- WebSockets
  - Available in all modern browsers (certainly any with WebRTC support)
  - Fast
  - Securable

Demo #4: Web Communicator

- A fully-featured unified communications client
- Makes use of WebRTC and WebSockets
- Multiple WebSocket/DataChannel connections for multiple protocols
  - MSRP (file-transfer), SIP (session signalling), and XMPP (messaging and presence)
- No need to create a new application for every target platform
- Browsers without WebRTC support can still use WebSocket for file-transfer, messaging, presence, and other data


Crocodile Web Communicator - Google Chrome
https://www.crocodiletalk.com/communicator/
CROCODILE
£110.93
Status:
Address
Nickname
Crocodile
Gavin Llewellyn
Hugh Waite
James Wyatt
John Parr
Konstantin Levchin
Other
Chat
James Wyatt
Me 11:49:23
Hi James
James Wyatt 11:49:33
Hi.
Me 11:49:48
Do you have that file I needed?
Send
File Transfers
James Wyatt
InterestingGIF.gif
1.02MB of 2.29MB - 44.5%
130KB/s


Crocodile Web Communicator - Google Chrome
Crocodile Web Communic
https://www.crocodiletalk.com/communicator/
£148.69
Status:
Address
Nickname
Andy Miller
Gavin Llewellyn
Hugh Waite
James Wyatt
John Parr
Konstantin Levchin
daz
Other
CROCODILE
TALK

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455, I. Fette (Google, Inc) et al, December 2011

http://tools.ietf.org/html/rfc6455

To enable Web applications to maintain bidirectional communications with server-side processes, this specification introduces the WebSocket interface.

The WebSocket API (W3C Candidate Recommendation), I. Hickson (Google, Inc), 20 September 2012

http://www.w3.org/TR/websockets

WebSockets: safe and secure

- A raw TCP/UDP API for Javascript would be dangerous
  - There would be no need to fool users into installing trojans
- The WebSocket protocol is asynchronous
  - Connections can only be established from the client side
- Data from client to server is masked
  - Prevents in-line proxies from mistaking the data for HTTP and modifying it
- Can be secured using TLS

Easy to use

- Very simple API
  - Constructor creates (opens) the connection
  - Methods: *close()*, *send()*
  - Events: *onopen()*, *onerror()*, *onclose()*
- Has the advantages of TCP and UDP
  - Data is framed – no need to parse the stream to work out where messages start and end
  - Frame delivery is guaranteed and in-order
- Interpretation of the frames is based on subprotocol not TCP or UDP port

Opening a connection (Handshake)

Request from client (browser)

```
GET wss://edge00.crocodilertc.net/4m9e4ipsfd8uh0leg0kr HTTP/1.1
Origin: https://www.crocodiletalk.com
Host: edge00.crocodilertc.net
Sec-WebSocket-Key: ywV2YxcaL0DMDVPyeHYj3Q==
Upgrade: websocket
Sec-WebSocket-Protocol: sip
Connection: Upgrade
Sec-WebSocket-Version: 13
```

Response from server

```
HTTP/1.1 101 Switching Protocols
Access-Control-Allow-Origin: https://www.crocodiletalk.com
Connection: upgrade
Sec-WebSocket-Accept: 9H9dBstuq+Y4Be2Ql7WWkV6tnjA=
Sec-WebSocket-Protocol: sip
Upgrade: websocket
```

The browser API handles this for you

Controlling connections

- Two types of frame
  - Data frames
  - Control frames
    - Close

      If you receive a close on a connection that you have not send a close on, send a close on that connection
    - Ping

      If you receive a ping on a connection send a pong on that connection
    - Pong

The browser API handles this for you

Sending/receiving frames

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

RFC 6455, section 5.2

The browser API handles this for you

Proxies and subprotocols

- Proxies
  - In-line proxies may be an issue
    - Masking helps avoid frame corruption, but sometimes the handshake fails
    - Using TLS avoids the issue and is good-practice anyway
  - Configured proxies
    - Must support the CONNECT HTTP request
- Subprotocols
  - http://www.iana.org/assignments/websocket/websocket.xml

The simplest form of signalling

- Exchange json blobs over WebSockets
- But beware, when you go beyond the basic “call” signalling gets hard (especially in the edge cases)
- Standards based signalling schemes have already solved:
  - Session liveness
  - Hold
  - Renegotiate
  - Transfer
  - and more...

Signalling options

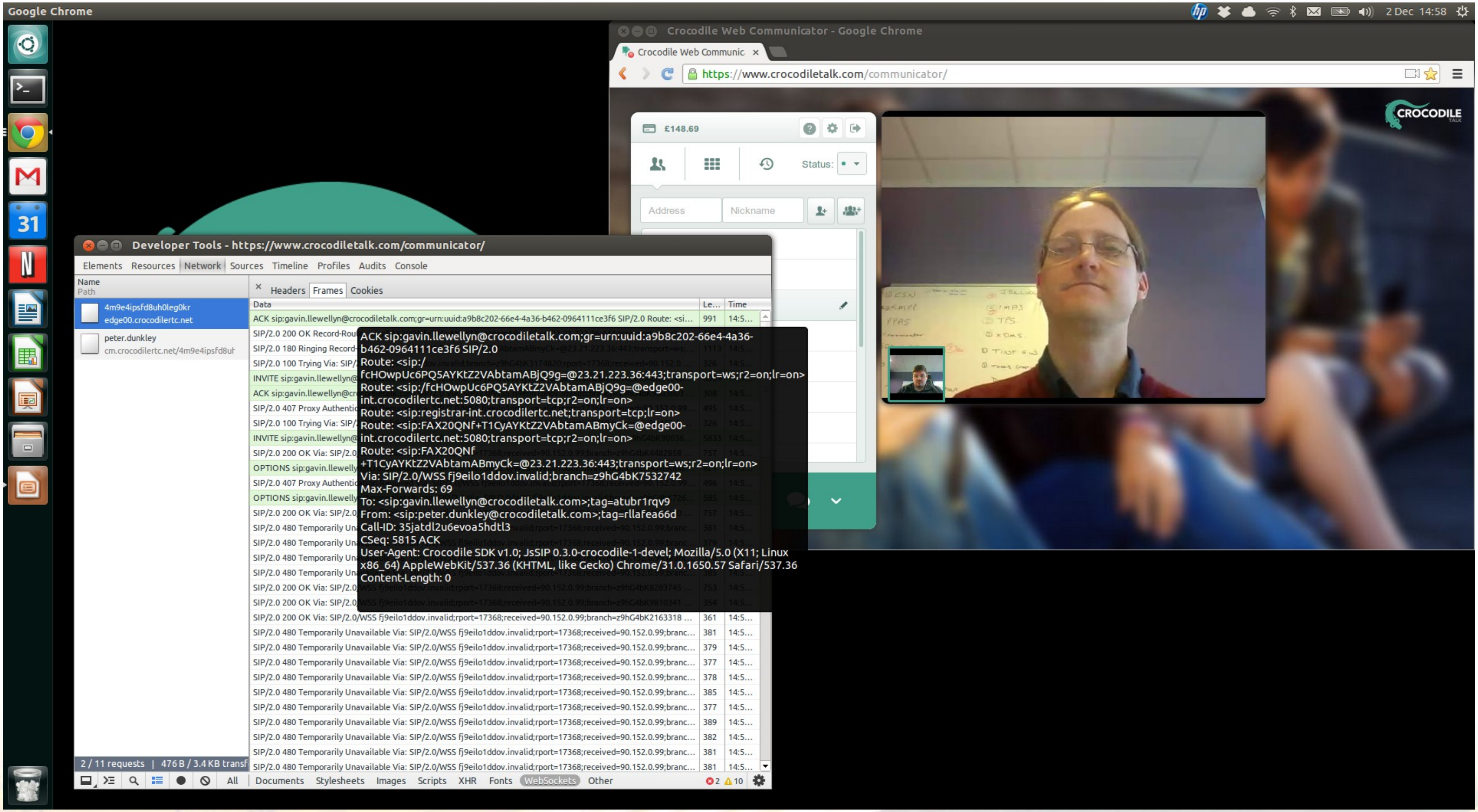
- Open standards are usually best
  - SIP over WebSocket, http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket
  - XMPP over WebSocket, http://tools.ietf.org/html/draft-moffitt-xmpp-over-websocket
  - OpenPeer, http://openpeer.org/
- The WebRTC API is easy but signalling is often hard
  - There are many open-source libraries that do the signalling
  - The library APIs vary in complexity to meet every need
  - Hosted infrastructure lets you add real-time communications to your website without having to build a network yourself

Opensource projects

- Node.js
  - Many WebSocket libraries available (for example, support in socket.io)
- SIP Servers
  - Asterisk, Kamailio, OverSIP, reSIPprocate
- SIP Clients
  - JAIN-SIP-Javascript, JsSIP, QoffeeSIP, sipml5
- XMPP Servers (and connection managers)
  - ejabberd-websockets, node-xmpp-bosh, openfire-websockets
- XMPP Clients
  - JSJaC, Strophe

Demo #5: Chrome Developer Tools

- Google Chrome provides a range of tools to help you create and debug applications
- You can add view code and add breakpoints
- You can modify code and view debug
- You can examine what is happening on the wire
  - The *Network* → *WebSockets* view can be particularly helpful – especially if using TLS
- ...


Google Chrome
2 Dec 14:58
Crocodile Web Communicator - Google Chrome
https://www.crocodiletalk.com/communicator/
£148.69
Status:
Address
Nickname
CROCODILE
Developer Tools - https://www.crocodiletalk.com/communicator/
Elements Resources Network Sources Timeline Profiles Audits Console
Headers Frames Cookies
4m9e4ipsfd8uh0leg0kr
edge00.crocodilertc.net
peter.dunkley
ACK sip:gavin.llewellyn@crocodiletalk.com;gr=urn:uuid:a9b8c202-66e4-4a36-b462-0964111ce3f6 SIP/2.0
Route: <sip:/fcHOwpUc6PQ5AYKtZ2VAbtamABjQ9g=@23.21.223.36:443;transport=ws;r2=on;lr=on>
Route: <sip:/fcHOwpUc6PQ5AYKtZ2VAbtamABjQ9g=@edge00-int.crocodilertc.net:5080;transport=tcp;r2=on;lr=on>
Route: <sip:registrar-int.crocodilertc.net;transport=tcp;lr=on>
Route: <sip:FAX20QNf+T1CyAYKtZ2VAbtamABmyCk=@edge00-int.crocodilertc.net:5080;transport=tcp;r2=on;lr=on>
Route: <sip:FAX20QNf+T1CyAYKtZ2VAbtamABmyCk=@23.21.223.36:443;transport=ws;r2=on;lr=on>
Via: SIP/2.0/WSS fj9eilo1ddov.invalid;branch=z9hG4bK7532742
Max-Forwards: 69
To: <sip:gavin.llewellyn@crocodiletalk.com>;tag=atubr1rqv9
From: <sip:peter.dunkley@crocodiletalk.com>;tag=rllafea66d
Call-ID: 35jatdl2u6evoa5hdtl3
CSeq: 5815 ACK
User-Agent: Crocodile SDK v1.0; JsSIP 0.3.0-crocodile-1-devel; Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.57 Safari/537.36
Content-Length: 0
2 / 11 requests | 476 B / 3.4 KB transf
All Documents Stylesheets Images Scripts XHR Fonts WebSockets Other

Dealing with firewalls

- WebRTC is peer-to-peer technology
- Sometimes firewall and NAT devices get in the way
- ICE (Interactive Connectivity Establishment) is mandatory
  - STUN (Session Traversal Utilities for NAT) helps in most cases
  - TURN (Traversal Using Relays around NAT) helps when STUN doesn't
- If a firewall is specifically configured to block real-time communications your options are limited
  - TURN over WebSockets is now under development

https://code.google.com/p/rfc5766-turn-server/

Desktop and mobile apps

- This technology isn't just for browsers
- Native WebRTC is possible
  - Mobile: Android and iOS libraries, and BB10 has support in the OS
  - Desktop: Linux, OS X, and Windows libraries
- Native WebSocket libraries are available
  - WebSockets are a sensible option for mobile app developers who want a safe way to exchange data with servers

HOWTO: A SIP over WebSockets Server

- Download, build, and install Kamailio 4.1
- Create a kamailio.cfg file based on the following code snippets

http://www.kamailio.org/

Handling WebSocket handshakes in Kamailio

```
...
tcp_accept_no_cl=yes
...
event_route[xhttp:request] {
    set_reply_close();
    set_reply_no_connect();

    if ($hdr(Upgrade)=~"websocket"
        && $hdr(Connection)=~"Upgrade"
        && $rm=~"GET") {

        # Validate as required (Host:, Origin:, Cookie:)

        if (ws_handle_handshake())
            exit;
    }

    xhttp_reply("404", "Not Found", "", "");
}
```

WebSocket clients are always behind a NAT

- Javascript applications cannot see the real IP address and port for the WebSocket connection
- This means that the SIP server cannot trust addresses and ports in SIP messages received over WebSockets
- nathelper and/or outbound can be used to solve this problem

Using nathelper on SIP over WebSocket requests

```
modparam("nathelper|registrar", "received_avp", "$avp(RECEIVED)")
...
request_route {
    route(REQINIT);
    route(WSDETECT);
    ...
route[WSDETECT] {
    if (proto == WS || proto == WSS) {
        force_rport();
        if (is_method("REGISTER")) {
            fix_nated_register();
        } else if (is_method("INVITE|NOTIFY|SUBSCRIBE")) {
            add_contact_alias();
        }
    }
}
...
route[WITHINDLG] {
    if (has_totag()) {
        if (loose_route()) {
            if (!isdsturiset()) {
                handle_ruri_alias();
            }
            ...
```

Using nathelper on SIP over WebSocket responses

```
onreply_route {
    if ((proto == WS || proto == WSS)
            && status =~ "[12][0-9][0-9]") {
        add_contact_alias();
    }
}
```

What about web-calls to non-web endpoints?

- Use mediaproxy-ng from SIPWise

https://github.com/sipwise/mediaproxy-ng

- Companion Kamailio module: rtpproxy-ng

http://kamailio.org/docs/modules/devel/modules/rtpproxy-ng.html

- SIP Signalling is proxied instead of B2BUA'd (that is, not broken)

Catch 488 to invoke mediaproxy-ng

```
modparam("rtpproxy-ng", "rtpproxy_sock", "udp:localhost:22223")
...
route[LOCATION] {
          ...
          t_on_failure("UA_FAILURE");
}
...
failure_route[UA_FAILURE] {
     if (t_check_status("488") && sdp_content()) {
          if (sdp_get_line_startswith("$avp(mline)", "m=")) {
               if ($avp(mline) =~ "SAVPF")) {
                    $avp(rtpproxy_offer_flags) = "froc-sp";
                    $avp(rtpproxy_answer_flags) = "froc+SP";
               } else {
                    $avp(rtpproxy_offer_flags) = "froc+SP";
                    $avp(rtpproxy_answer_flags) = "froc-sp";
               }
               # In a production system you probably need to catch
               # "RTP/SAVP" and "RTP/AVPF" and handle them correctly
               # too
          }
          append_branch();
          rtpproxy_offer($avp(rtpproxy_offer_flags));
          t_on_reply("RTPPROXY_REPLY");
          route(RELAY);
     }
}
...
```

Handle replies to the retried INVITE

```
modparam("rtpproxy-ng", "rtpproxy_sock", "udp:localhost:22223")
...
failure_route[UA_FAILURE] {
    ...
    t_on_reply("RTPPROXY_REPLY");
    route(RELAY);
}

onreply_route[RTPPROXY_REPLY] {
    if (status =~ "18[03]") {
        # mediaproxy-ng only supports SRTP/SDES – early media
        # won't work so strip it out now to avoid problems
        change_reply_status(180, "Ringing");
        remove_body();
    } else if (status =~ "2[0-9][0-9]" && sdp_content()) {
        rtpproxy_answer($avp(rtpproxy_answer_flags));
    }
}
...
```

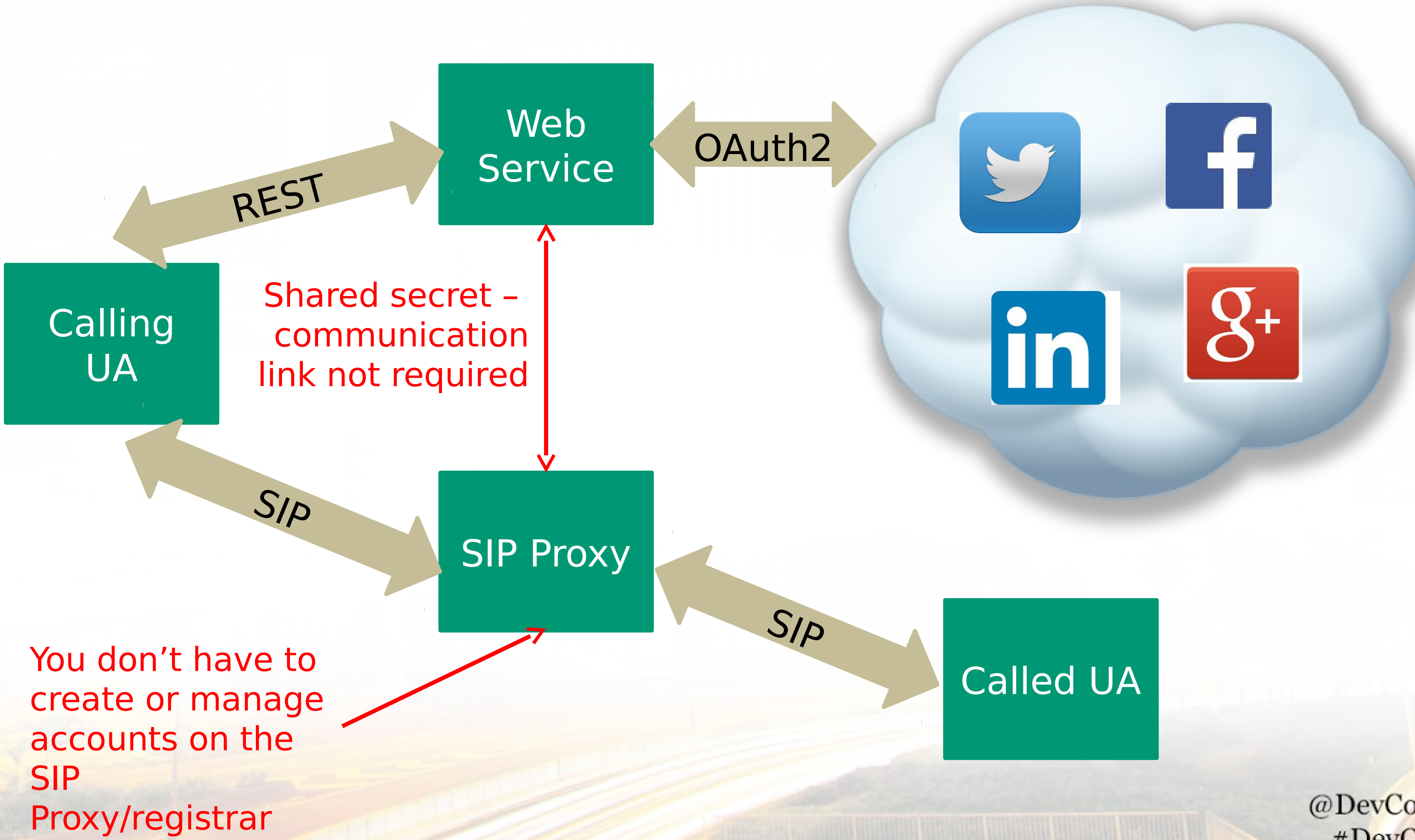
Current mediaproxy-ng limitations

- No support for SRTP/DTLS
  - SRTP/DTLS is a MUST for WebRTC and SRTP/SDES is a MUST NOT
  - mediaproxy-ng works with Google Chrome today (but Google will be removing SRTP/SDES over the next year)
  - mediaproxy-ng does not work with Firefox at this time
- Does not support “bundling”/”unbundling”
  - WebRTC can “bundle” audio and video streams together, but mediaproxy-ng does not support this yet
  - Google Chrome does not currently support “unbundling”
  - You can have an audio stream, or a video stream, but not an audio and video stream at this time

HOWTO: Authenticate SIP using a web-service

- No communication required between authentication server and Kamailio
- Credentials expire (the expiry time is chosen by the authentication server)
- Extract username and password from the “GET” used for HTTP handshake and authenticate there, <u>or</u>
- Use the credentials for digest authentication of SIP requests
- Check the From-URI or To-URI in SIP headers match the user part of the credential

http://kamailio.org/docs/modules/devel/modules/auth_ephemeral.html


Web Service
OAuth2
REST
Calling UA
Shared secret – communication link not required
SIP
SIP Proxy
SIP
Called UA
You don’t have to create or manage accounts on the SIP Proxy/registrar

Authenticating the handshake

```
...
tcp_accept_no_cl=yes
...
modparam("auth_ephemeral", "secret", "kamailio_rules")
...
modparam("htable", "htable", "wsconn=>size=8;")
...
event_route[xhttp:request] {
      ...
            # URI format is /?username=foo&password=bar
            $var(uri_params) = $(hu{url.querystring});
            $var(username) = $(var(uri_params){param.name,username,&});
            $var(password) = $(var(uri_params){param.name,password,&});
            # Note: username and password could also have been in a Cookie: header

            if (!autheph_authenticate("$var(username)", "$var(password)")) {
                  xhttp_reply("403", "Forbidden", "", "");
                  exit;
            }

            if (ws_handle_handshake()) {
                  $sht(wsconn=>$si:$sp::username) = $var(username)
                  exit;
            }
      ...
event_route[websocket:closed] {
      $var(regex) = $si + ":" $sp + ".*";
      sht_rm_name_re("wsconn=>$var(regex)");
}
```

Checking SIP requests

```
...
request_route {
     route(REQINIT);
     route(WSDETECT);
     ...
     if (!(proto == WS || proto == WSS))
          route(AUTH);
     ...
route[WSDETECT] {
     if (proto == WS || proto == WSS) {
          $var(username) = (str) $sht(wsconn=>$si:$sp::username);
          if ($var(username) == $null || $var(username) == “”) {
               send_reply(“403”, “Forbidden”);
               ws_close(1008, “Policy Violation”);
               exit;
          }

          if (!autheph_check_timestamp(“$var(username)”)
                    || (is_method(“REGISTER|PUBLISH”)
                              && !autheph_check_to(“$var(username)”))
                    || (!has_totag() && !autheph_check_from(“$var(username)”))) {
               send_reply(“403”, “Forbidden”);
               ws_close(1008, “Policy Violation”);
               exit;
          }

          force_rport();
          ...
```

Questions?

Code: https://github.com/crocodilertc

Email: peter.dunkley@crocodilertc.net

Twitter: @pdunkley