WebSockets

WebSockets

and browser-based real-time communications

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

1990

W3C produces the DOM1 specification

1998

WebSocket and WebRTC implementations become available

2011

1996

Microsoft and Netscape introduce different mechanisms for DHTML

2004

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

CROCODILE

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

Alexander Graham Bell patents the telephone

From the 1960s onwards digital exchanges start to appear

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

1792 1837 1876 1919 1960s > 1990s > 2011

Rotary dial enters service

First commercial electrical telegraph created by Cooke and Wheatstone

1963: DTMF enters service

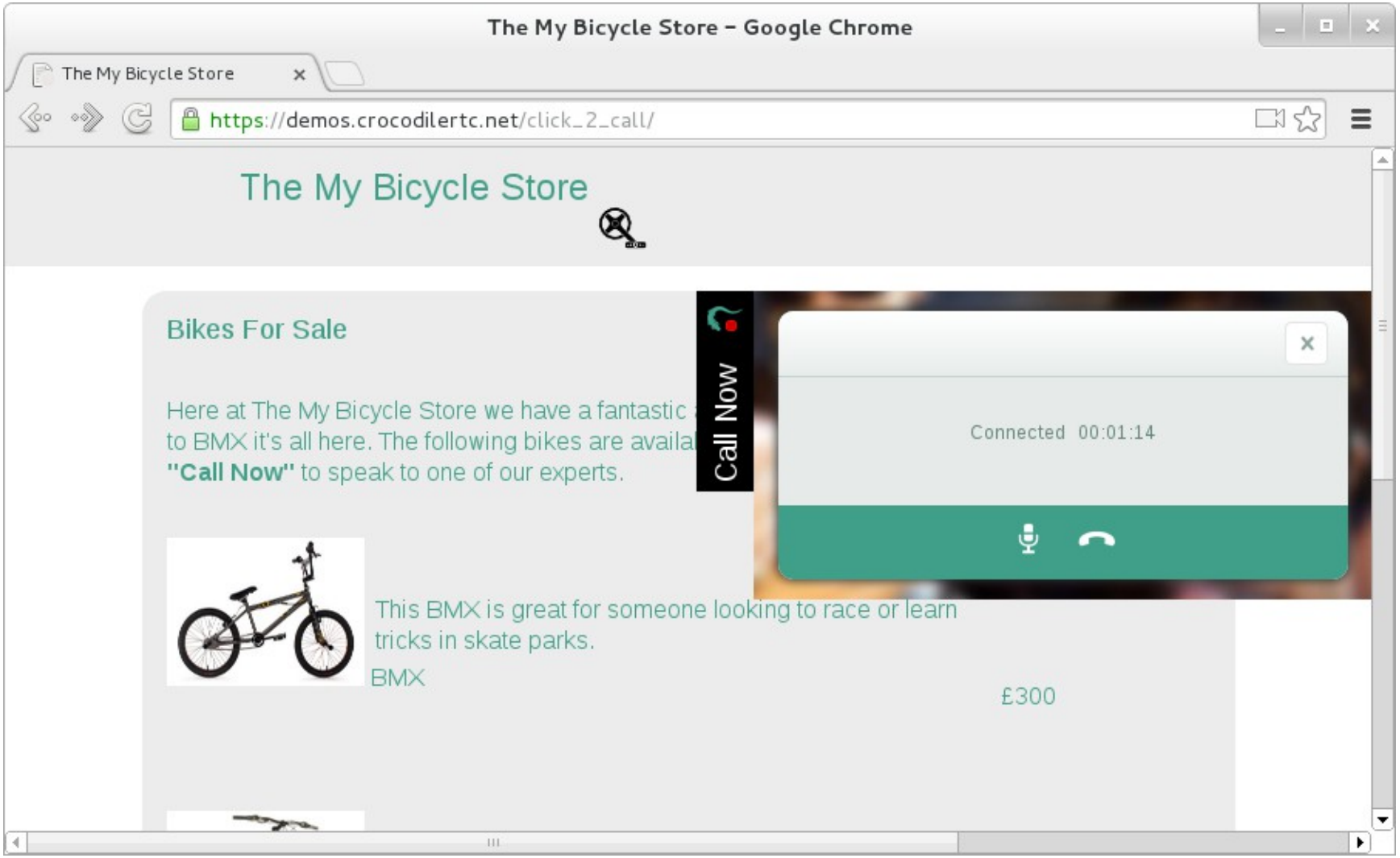
WebSocket and WebRTC implementations become available

CROCODILE

Demo #1: Click-2-call

- A simple addition to any commercial web-site
- Makes use of WebSocket and WebRTC
- Can be enhanced to make the calling experience richer
  - Context aware communication
  - A fun queuing experience

Demo #1: Click-2-call



CROCODILE

What are WebSockets?

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

CROCODILE

Aren't there other options?

- Ajax
  - You need to open a new connection each time you refresh
  - The web-server needs to process (and run scripts, lookup databases, etc) for each connection
  - With WebSockets you open the connection and keep it open as long as you need it
- BOSH
  - Not an IETF standard
  - Less widely supported than WebSocket
  - WebSocket requires a single network connection, BOSH uses two

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

CROCODILE

Controlling connections

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

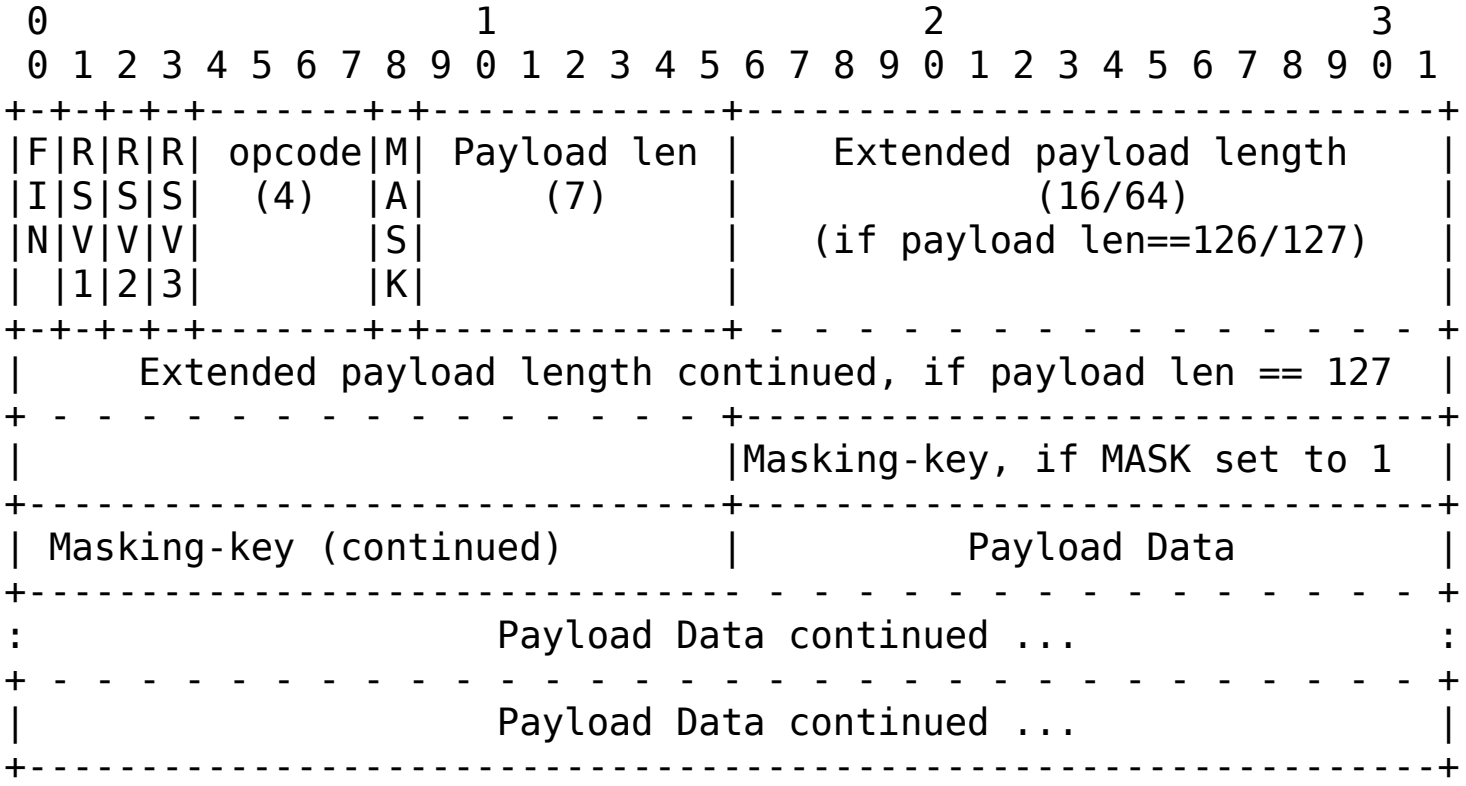
      If you receive a close on a connection that you have not send a close on, send a close on that connection
    - Ping

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

The browser API handles this for you

CROCODILE

Sending/receiving frames



RFC 6455, section 5.2

The browser API handles this for you

CROCODILE

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

Opensource projects

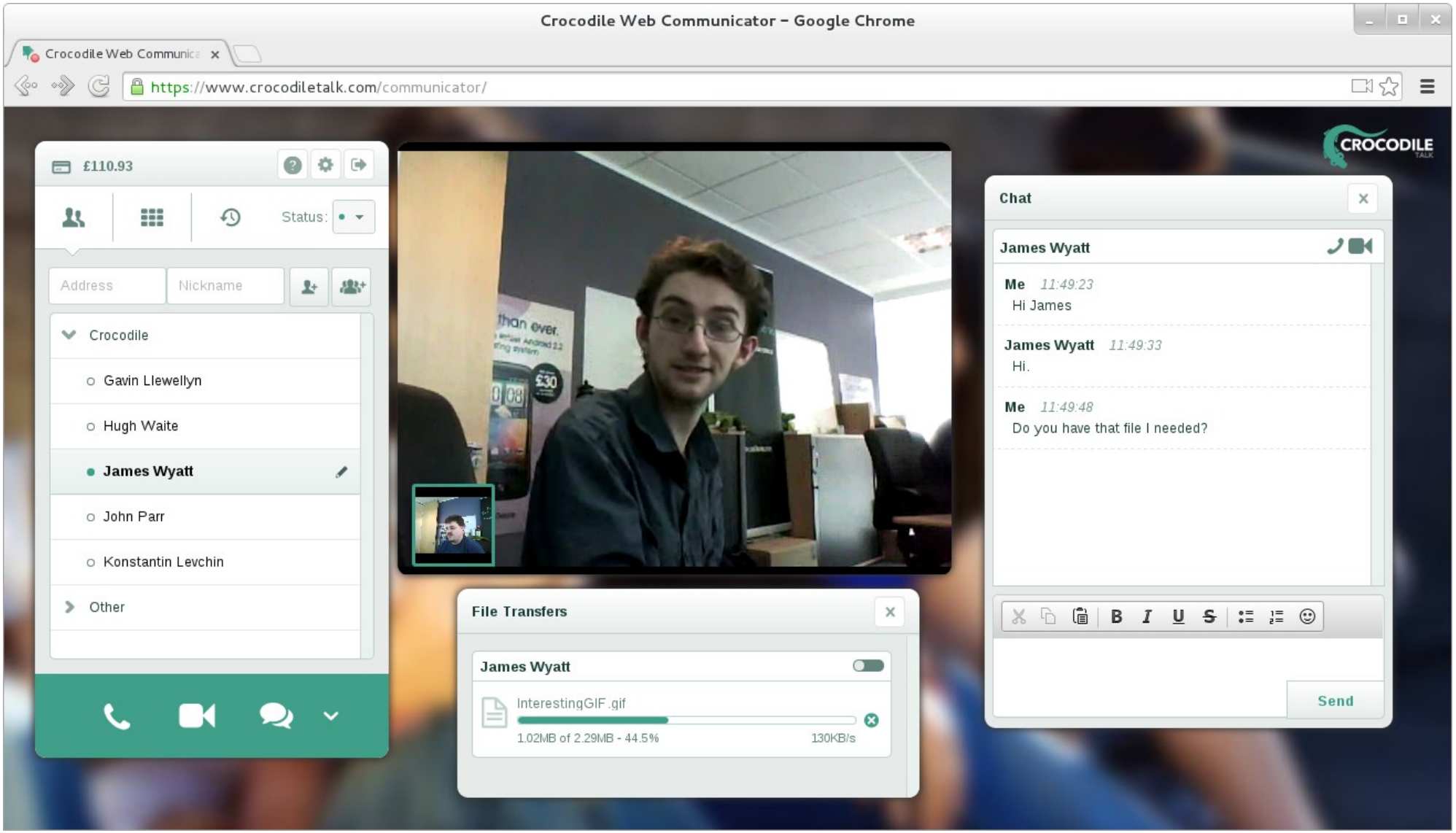
- Node.js
  - Many WebSocket libraries available
- SIP Servers
  - Asterisk, Kamailio, OverSIP, reSIPprocate
- SIP Clients
  - JAIN-SIP-Javascript, JsSIP, QoffeeSIP, sipml5
- XMPP Servers (and connection managers)
  - ejabberd-websockets, node-xmpp-bosh, openfire-websockets
- XMPP Clients
  - JSJaC,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

Demo #2: Web Communicator

- A fully-featured unified communications client
- Makes use of WebSocket and WebRTC
- Multiple WebSocket 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

Demo #2: Web Communicator



CROCODILE

WebRTC

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/

CROCODILE

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/

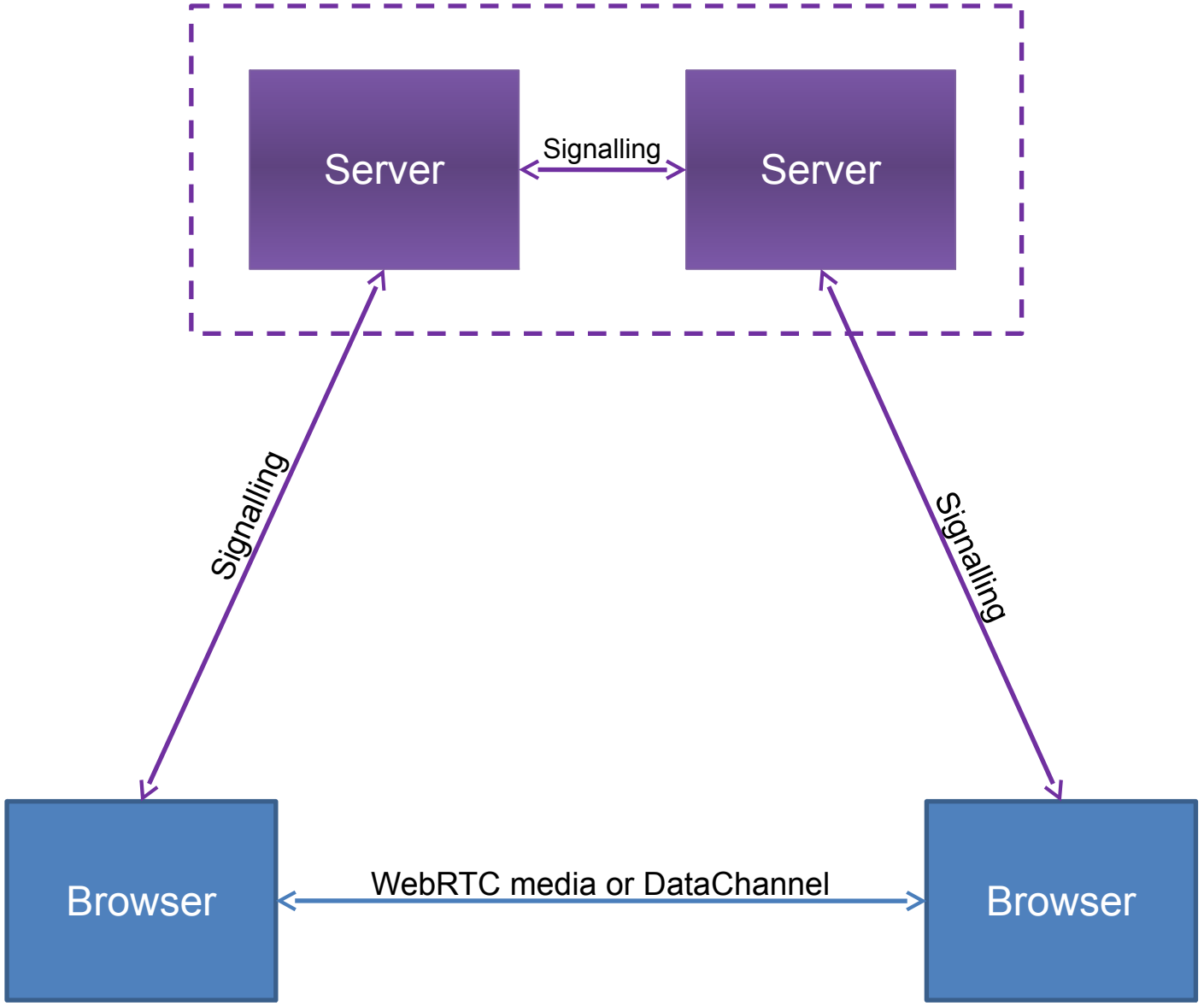
Available (to varying degrees) in Chrome and Firefox stable

CROCODILE

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

The signalling trapezoid/triangle



CROCODILE

Signalling options

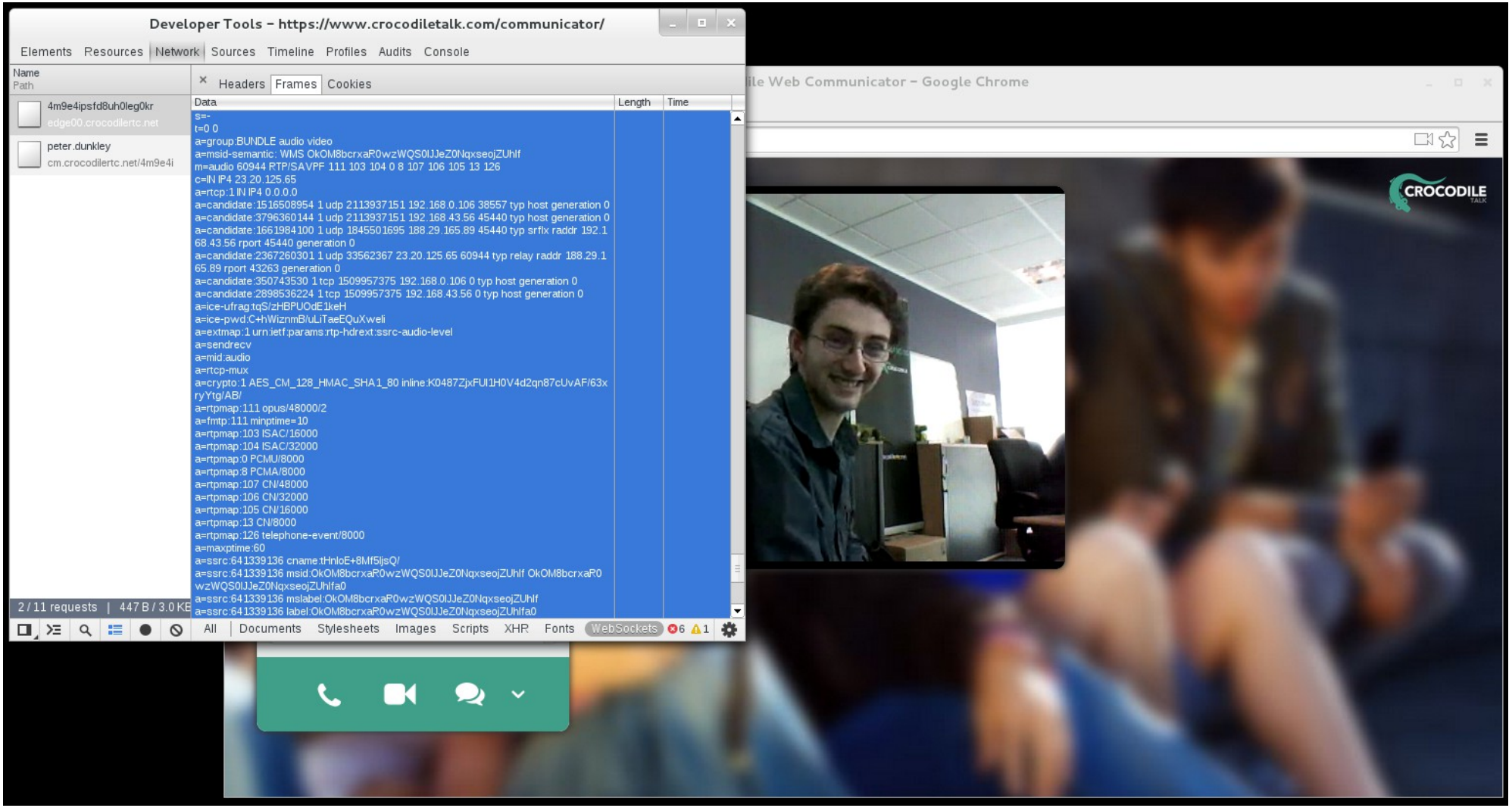
- Open standards are usually best
  - SIP over WebSocket, http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket
  - XMPP over WebSocket, http://tools.ietf.org/html/draft-moffitt-xmpp-over-websocket
  - OpenPeer, http://openpeer.org/
- The WebRTC API is easy but signalling is often hard
  - There 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

Demo #3: 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
- …

CROCODILE

Demo #3: Chrome Developer Tools



CROCODILE

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

Applications of this technology

- Telecommunications
  - Unified communications, corporate infrastructure, call centres
- Distance learning
  - Virtual colleges and universities
- Telemedicine
  - Providing medical services out-of-hours and to remote locations
- Peer-to-peer applications
  - File-sharing, collaboration
- Gaming
  - Multi-player interactive (with and without servers)
- ...

Browser support today

http://caniuse.com/

	IE	*FF*	*Chrome*	*Safari*	*O*	*iOS Safari*	*O Mini*	*Android Browser*	*Blackberry Browser*	*O Mobile*	*Chrome for Android*	*FF for Android*
WebSocket	10.0	6.0	14.0	6.0	12.1	6.0	-	-	7.0	12.1	27.0	22.0
WebRTC	-	17.0***	21.0***	-	-*	-	-	-	10.0***	-*	-**	-

* Partial support for *GetUserMedia* is in some old versions
** WebRTC is known to work in the latest version if you set a flag
*** As WebRTC is still under development later browser versions will be more stable and have more features

CROCODILE

Desktop and mobile apps

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

Questions?

Code: https://github.com/crocodilertc

Email: peter.dunkley@crocodilertc.net

Twitter: @pdunkley

CROCODILE

Crocodile WebRTC SDK and Network

www.crocodilertc.net