A universal architecture to anonymize any application or protocol and turn it into an independent decentralized p2p network inside browsers and servers, with browsers acting as servers

CONVERGENCE

1. Description

This proposal is a complete redesign of our initial Convergence proposal from 2015 (http://www.peersm.com/Convergence.pdf) which was intended as a research study for an EU call

By "Convergence" in this proposal we don't refer to a specific network or node but to methods using the Convergence principles

1.1 Background and rationale

The initial Convergence proposal was written based on the observation that we must invent one network/system per need if we want to evade big data centralization and protect privacy/anonymity: to browse, to chat, to email, to exchange files, to do social networking or cooperative work, to do crypto currency, to protect the users from their connected objects, to handle peer identities.

So it did envision the support of any type of applications and protocols on top of a secure anonymization system, inside browsers and servers

The first part is very exactly what IPFS did, including the crypto currency concept in our proposal to sustain the network (Filecoin)

But IPFS is not at all designed for privacy, the IPFS team knows that they will have to address the issue but it's not even part of their roadmap

And IPFS adoption is not for all protocols, many other networks will not use it and lack privacy too

That's why we are proposing the Convergence concepts, which are still very up to date, with a major novelty: the introduction of *Evented Pipes Methods* (https://github.com/Ayms/node-Tor#phase-4-and-phase-5 and https://github.com/Ayms/node-Tor/blob/master/docs/README.md**)**

This implies a more generic architecture usable by any protocol or application, without imposing a format as initially foreseen, the example of IPFS demonstrates that it's useless to propose an architecture that duplicates things, but very useful to propose a privacy by design architecture usable by everybody

We propose to implement a **decentralized and secure architecture using the most widely spread unified/open/standard system (not platform/device dependent): browsers, that would allow any solution/protocol to work on top of it, without requiring any specific installation or skills to use it**.

Browsers can run javascript applications as standalone applications and act as autonomous nodes to relay data

The installation process of Convergence and related applications will be as simple as browsing a site that will deliver the code without requiring any effort from the users, the applications will then execute inside browsers with the possibility to continue to run in background when they are closed, in addition the code can be stored in a secure way inside a bookmarklet and checked via a third party (a hash on github for example) eliminating the need to load it from the outside

The integration by any protocol or application will be as simple as chaining pipes to Convergence

The whole system is using hybrid nodes whether inside browsers and servers

1.2 Main concepts and implementation

The anonymizer networks are thousands of nodes, the bitcoin network is the same, the bittorrent network is hundreds of millions of peers, **browsers are billions**.

So browsers are a good candidate to build p2p decentralized network, they allow to easily use applications without any installation on any device, fix or mobile

In addition browsers have all required features: browsers can discuss between each other (WebRTC), browsers can discuss with networks (WebSockets, XHR), browsers can store data and manipulate files (indexedDB, File API), browsers can stream (built-in solution and Streams API), browsers can handle work in background (Workers), browsers can handle crypto (WebCrypto), browsers can proxy (SOCKS), browsers can handle modules as independent applications (Web Components)

One past drawback of using a browser for applications was that it was required to let it open to run them. This was solved with the Service Workers that allow to run a browser instance and associated applications in background.

The implementation will use the **node-Tor project** (https://github.com/Ayms/node-Tor), the initial development has been founded by Naïs (two years), phases 1 to 3 (refactor/modernize the code, split into modules, release open source) were funded by NLnet under NGI PET call

We propose first to implement phases 4 (partially developed), 5 and 6 which consists in implementing the *Evented Pipes Methods* (allowing typically *ipfs.pipe(node-Tor)* or *bitcoin.pipe(node-Tor)*), update the Tor protocol to elliptic crypto and implement the WebRTC transport

The node-Tor project is the only existing implementation of the Tor protocol in Javascript, it exists as a nodejs module which has been browserified as a standalone secure webapp also

It should not be misunderstood with the Tor network, even if compatible with it, this is a complete implementation of the Tor protocol that can be used to anonymize any protocol/data transfer

It allows to create Tor protocol nodes as servers or inside browsers acting as nodes also, using the module as an onion proxy, onion router or both

The project is already usable but partially in fact, right now what starts everything is always a simple call to the *node-Tor* function, typically *node-Tor(request)*

Even if simple, the problem is that properties of the *request* object must be set depending on the protocol used which is not easy to do, and the initial design was more request oriented, ie send a request and get the response, which could be a video stream, then not really fully bidirectional

This is the purpose of the design of the *Evented pipes methods* that will simply allow to pipe (and chain) any protocol to node-Tor both way

Example of pipe and chaining:

http.pipe(parser).pipe(gzip).pipe(TLS).pipe(node-Tor) (https over node-Tor)

or using the RendezVous protocol

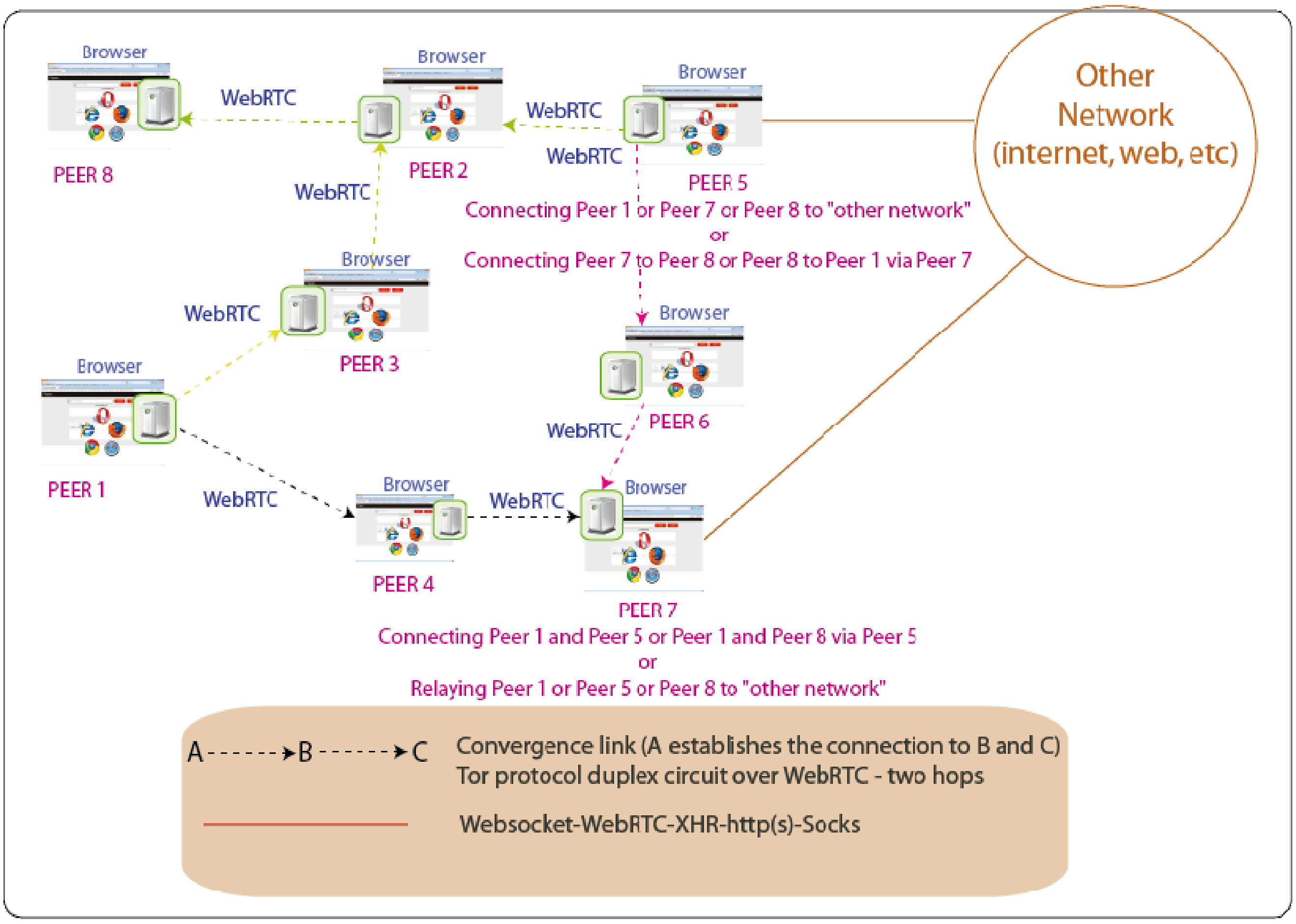
bitcoin/ipfs/webtorrent/etc).pipe(node-Tor).pipe(RDV peer)

Another issue is that currently "old" RSA crypto is implemented, and it does not fit very well with modern projects (like IPFS) so we propose to implement the MESSAGE2 Tor protocol messages using elliptic crypto

The *Evented Pipes Methods* will allow to pipe any protocol whether from a nodejs platform, browser, but also from other platforms, for example using the unix pipes or sockets

Example: *bitcoin* | *node-Tor* | *bitcoin*

The network architecture is briefly summarized in figure 1:


Browser
WebRTC
PEER 8
PEER 2
PEER 5
PEER 3
PEER 1
PEER 4
PEER 6
PEER 7
Other Network (internet, web, etc)
Connecting Peer 1 or Peer 7 or Peer 8 to "other network"
or
Connecting Peer 7 to Peer 8 or Peer 8 to Peer 1 via Peer 7
Connecting Peer 1 and Peer 5 or Peer 1 and Peer 8 via Peer 5
or
Relaying Peer 1 or Peer 5 or Peer 8 to "other network"
A------➤B------➤C
Convergence link (A establishes the connection to B and C)
Tor protocol duplex circuit over WebRTC - two hops
Websocket-WebRTC-XHR-http(s)-Socks

Each peer is connected to several other peers via two hops who are serving the requesting peer directly or are acting as Rendez-vous points to relay the messages between the different peers, similar somewhere to the Tor hidden services mechanisms

Each peer maintains 5 anonymized circuits between browsers (WebRTC) and/or between browsers and servers (WebSocket) and/or between servers and servers

The peer discovery mechanism is supposed to be implemented by the protocol piping to Convergence, nevertheless we will implement a default one called the *facilitator* (part of phase 6) handling peers information (onion keys and addresses) both for browsers and servers and being able to relay the introduction messages for WebRTC (SDP)

The hybrid system will allow any combination of connections between browsers and servers, this will be implemented with phase 7

In addition relays are peers and peers are relay (dual peers), so they implement both the Tor protocol and the RDV protocol, which means that they can serve directly the requesting peer or serve it via a RDV peer, in addition peers can extend the hops to other peers (for example A is connected to C via B, C can't serve A but is connected to E via D who knows G that can serve A)

The dual peers and hops extensions capabilities will be implemented in phase 8, this is experimental and might be subject to changes, this does not substitute the RDV and dual peers concept but extends it

The RDV protocol is similar to Hidden services but without using onion addresses, instead it uses a simple hash of "*something*", it can be a reference to a file or a reference to a protocol, for example by standard it could be defined a hash of "Satoshi Nakamoto" for bitcoin that any peer implementing a bitcoin node would advertise to the RDV points, or a reference to "whatever" in fact, just giving the ability to any peer to advertise anonymously what it does

Right now the RDV protocol uses specific Tor protocol extended messages specified by us, we believe that it is useless and should be reverted to the Tor protocol standard messages, maybe using *padding* messages to advertise the hashes or with just one additional message (currently *RELAY_DB_INFO*), this will be implemented with phase 9

And to finish we will implement phase 10 piping Convergence via unix pipes

This will represent at the end a module usable by any protocol anonymizing any projects and offering them a p2p/user (human centric) control using browsers and modern web transports not involving necessarily third parties or servers

We have insisted here on the pipe and chaining capabilities within the Convergence architecture it can also easily bridge with other networks (for example the Web with Exit nodes)

The whole system will constitute **an encrypted serverless anonymous P2P network that safeguards the confidentiality and privacy of the user's information for any protocol and application, on any device fix or mobile**

1.3 State of the art

Many networks exist (Tor network, freenet, bittorrent, i2p, ipfs, cjdns, Tribler, WebTorrent, IPFS, Gnu net, Diaspora, etc)

To take a few, the Tor network has been designed for browsing only (unlike some people think, using wrongly hidden services for any purpose), is centralized and far too small to envision any P2P application on top of it, I2P and freenet have been designed to share content (although they propose services), with a level of complexity not accessible to normal users, the bittorrent network is used to share content only too, is quite efficient but not designed at all to protect privacy , same applies to IPFS, cjdns proposes the same type of architecture than Convergence but is more designed to prevent DDOS attacks and does not protect privacy/anonymity, the Tribler project is a Tor project-like implementation of the Tor protocol with bittorrent protocol not compatible with browsers.

So, some of them are including privacy/anonymity features but the common point of all those projects is that they are difficult to integrate with other projects, risky (for example we saw recently a bitcoin security breach while using the Tor project code) and/or just failed to sustain themselves as p2p networks, mainly because they are too small and there are no incentive to sustain them

As already stated node-Tor is the only project implementing the Tor protocol inside browsers, but that's not all, if we compare with Tor network projects' Flashproxy and Snowflake (still not even completely released) it is much more sophisticated since it does not only relay Tor messages but really handles the Tor protocol messages at browser level and it does implement the RDV protocol

To summarize, regarding the state of the art, nothing exist to allow what we are proposing

In addition, whether on browser or server side **the code is only ~1 MB not minified** (so 500 kB minified), which is quite small for what it does, making it easy to integrate compared to the sprawling Tor project code, and unlike the Tor project code it includes a very few dependencies making the code more robust, secure and easy to adapt

1.4 Proof of concept and test configuration

The proof of concept with browsers has been demonstrated by the Peersm project (http://www.peersm.com), which implements all the crypto of the Tor protocol inside browsers (the demo link http://peersm.com/peersm2 is for now disabled because the project is between two steps) and was the first p2p project able to stream videos live inside browsers from bittorrent

The test and phases validation configurations are:

- **Phases 4 (evented pipe methods) and 5 (elliptic crypto)**

  <browser> → <node-Tor node> → Tor network nodes → Tor network Exit node

  And

  <browser> → <node-Tor node> → Tor network nodes → <node-Tor RDV point>

  Where → are Tor protocol circuits piping the protocol(s)l used, it's of course bidirectional but shows the way the anonymized circuits are established

Then from the browser an URL can be entered and this will fetch the related resource via the Exit node (*http.pipe(node-Tor)*) or a reference to a file (hash) can be entered and this will transfer the data from a peer that has it (*ordb.pipe(node-Tor*), indexedDB is used here for browser storage

This demonstrates the compatibility with the Tor network (ie correct implementation of the Tor protocol with elliptic crypto), the p2p capabilities and how easy to use and efficient are the *Evented Pipes Methods*

- **Phase 6 (WebRTC):**

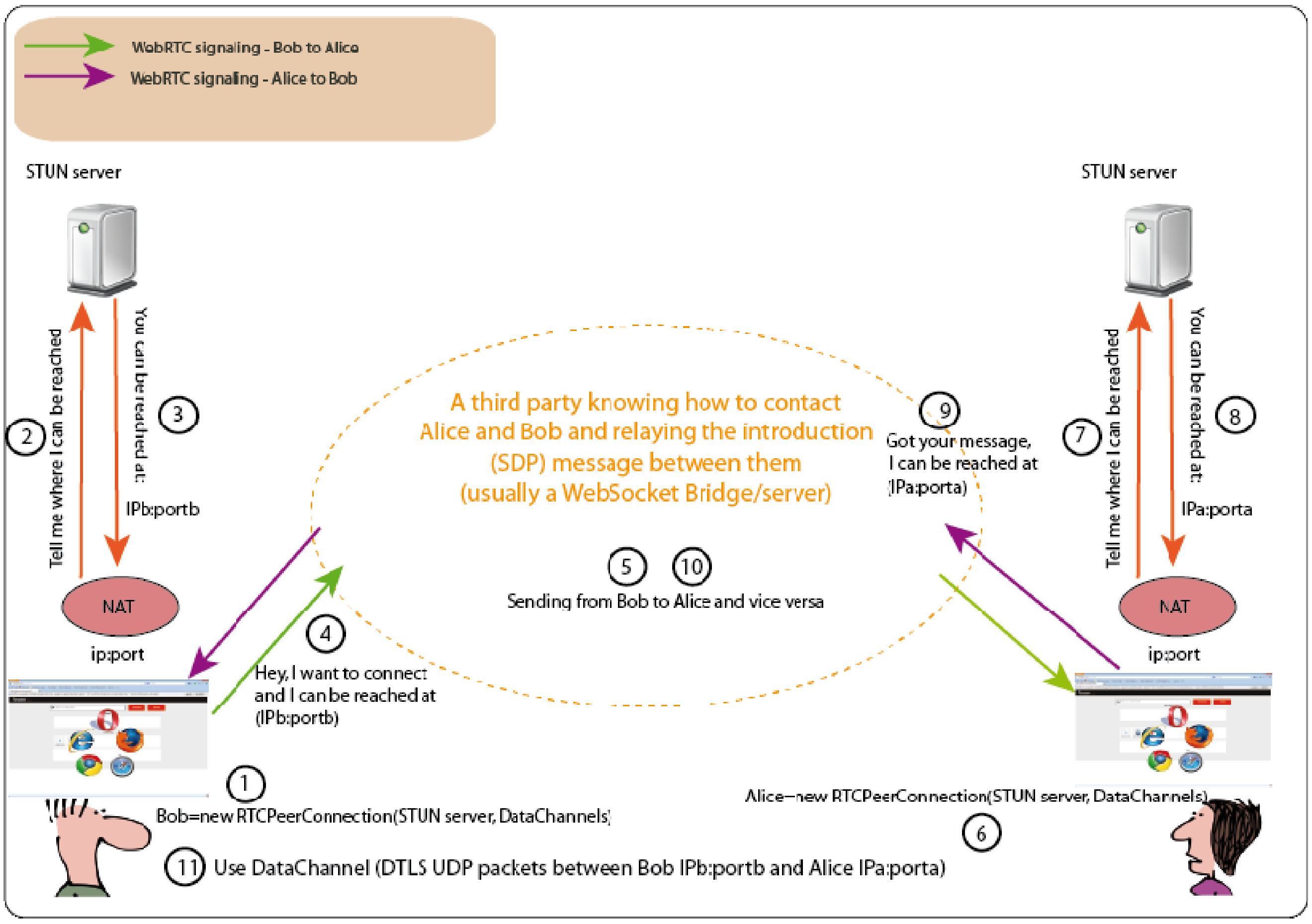
<browser1> →<browser2> →…→ <browser n>

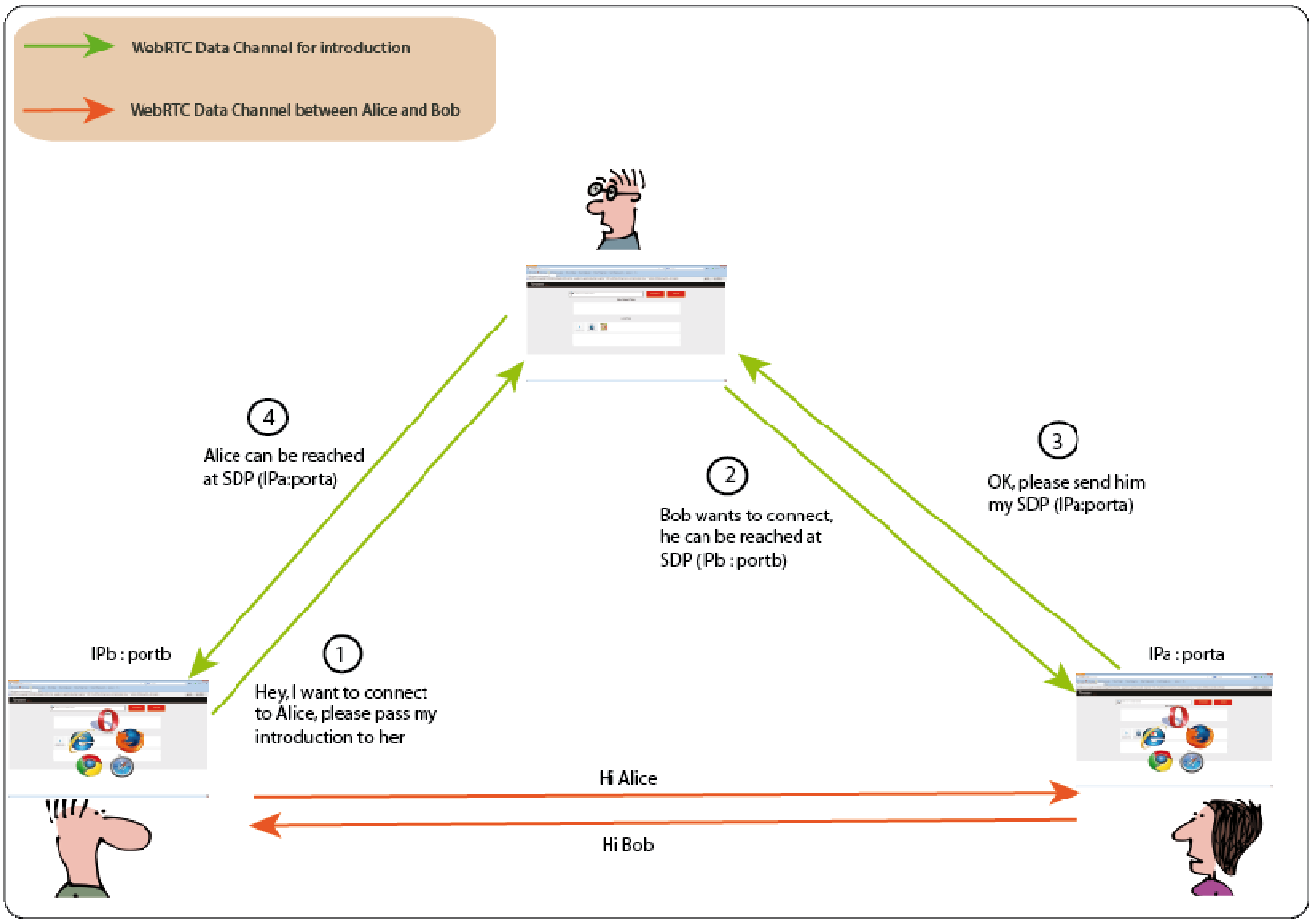
Or

<browser1> →<browser2> →<browser3 (RDV)>←<browser4>←<browser5>←<browser6>

Where browser can be a RDV point or a Tor protocol node, the test configuration here will use peer to peer download (what we call the *ordb* protocol)

Note that once bootstrapped the system can sustain itself alone since peers can introduce each other without requiring the *facilitator* (A knows B that knows C, B can send to A the B SDP introduction message, see below), a quick reminder how WebRTC works is illustrated below:


WebRTC signaling - Bob to Alice
WebRTC signaling - Alice to Bob
STUN server
STUN server
Tell me where I can be reached
You can be reached at:
IPb:portb
NAT
ip:port
Tell me where I can be reached
You can be reached at:
IPa:porta
NAT
ip:port
A third party knowing how to contact Alice and Bob and relaying the introduction (SDP) message between them (usually a WebSocket Bridge/server)
Got your message, I can be reached at (IPa:porta)
Sending from Bob to Alice and vice versa
Hey, I want to connect and I can be reached at (IPb:portb)
Bob=new RTCPeerConnection(STUN server, DataChannels)
Alice=new RTCPeerConnection(STUN server, DataChannels)
Use DataChannel (DTLS UDP packets between Bob IPb:portb and Alice IPa:porta)
1
2
3
4
5
6
7
8
9
10
11


WebRTC Data Channel for introduction
WebRTC Data Channel between Alice and Bob
4 Alice can be reached at SDP (IPa:porta)
2 Bob wants to connect, he can be reached at SDP (Pb : portb)
3 OK, please send him my SDP (IPa:porta)
IPb : portb
IPa : porta
1 Hey, I want to connect to Alice, please pass my introduction to her
Hi Alice
Hi Bob

- **Phase 7 (hybrid):**

<server node>→<browser>→<browser>→<server node>

Peer to peer download again where → uses WebRTC and WebSockets

- **Phase 8 and 9 (hops extension, RDV protocol and direct peers introduction)**

<browser1>→<browser2>→<browser3>←<browser4>←<browser5>←<browser6>←→<browse r7>←→<browser8>←→<browser9>

Where ←→ can be established both ways, and where browser1 will request a hash that browser3 can't serve but that browser9 can, or just a hash that browser3 knows to demonstrate the hops extension (with redesigned RDV protocol), dual peers capabilities and autonomous peer to peer introduction

- **Phase 10 (pipe/chaining methods from non nodejs/browser platforms):**

bitcoin | node-Tor | bitcoin

<node-Tor>→<node-Tor or Tor network>→<node-Tor or Tor network>→bitcoin node

Where “bitcoin” is based on our bitcoin-transactions module (https://github.com/Ayms/bitcoin-transactions) using stdin/stdout for the protocol (because standard bitcoin nodes don’t do this) and where an exit node is used to reach a bitcoin node via node-Tor and several hops (nodes can be server or browsers)

1.5 Quick technical and security overview

Browsers and javascript are often associated to the word “insecure”, this is somewhere true when we are considering browsing, this is false in the context of Convergence, because Convergence does not include traditional browsing, this section quickly scans some security concerns

- *Global architecture*

  - Crypto inside browsers

  While quite difficult some years ago, crypto inside browsers is now relatively easy, nevertheless we will keep node-Tor javascript crypto because the Tor protocol implements specific progressive hash and encryption supported by nobody (openssl, WebCrypto, NSS)

  - Number of hops

  Since WebRTC with the peer introduction mechanism makes it difficult to create connections, it appears not easy to build the standard three hops for each connection.

  In addition, Convergence does not foresee to use the CREATE_FAST cells, therefore the first node in a circuit can’t know it is the first one, it has to be noted that two hops is not necessarily the number of hops since the circuits are likely to be extended.

  - The Guards concept

  This concept of the Tor network insures to keep the same “well established”/long-running first node during a circuit lifetime which reduces the probability of connecting to someone that controls the first and the last node of a circuit and could therefore correlate traffic.

  We consider that this does not apply to Convergence principles, since browsers are volatile, subject to change and a Convergence based network is not centralized

  - Blocking browsers - obfuscation

  Browsers are difficult to block since users are usually behind a NAT and their IP address change, nevertheless WebSockets and WebRTC protocols have specific fingerprints and could be blocked, there are no signs today that this will happen (for example the Peersm file sharing project easily passes the Great China Firewall). Obfuscation could be used, now it has to be noted that it’s not trivial to determine what protocol implements a node piping to Convergence (can be IPFS, WebTorrent, whatever)

  - Security compared to Tor

For the general architecture if we compare to all the attacks/threats studied in the context of the Tor network, most of them cannot apply to Convergence. Because most of the attacks are linked to the inherent dangerousness of browsing the web and related implications.

- Adversaries tactics and tracking

We believe it matters that a node knows it is the first one (like the Tor network bridges), because it can deanonymize those that are connected to it, that's why Convergence will not use the CREATE_FAST cells, the nodes cannot know in what position they are (unlike the Tor network the first node cannot check among a finite known list of relays if the previous one is a relay or an user). In addition, two hops is not necessarily the path that will follow the messages since the peers can extend hops. So at the end it becomes quite difficult to know who is doing what and who is connected to who, because peers are relays and relays are peers

- NAT traversal

Convergence has no other choice than using what WebRTC offers, so using STUN servers for NAT traversal (but TURN relay servers are not considered), we don't think that it matters that STUN servers know about the peers (and therefore could be able to trace a connection).

- Code loading and centralization

The code of Convergence and related applications is loaded from a web site, typically the applications web sites.

But this is not centralized since the code can be duplicated on other web sites in case the above ones are blocked, or can be retrieved from different sources such as social networks, the cloud or blockchains.

Securing the code loading of a web application is an unsolvable problem if we don't involve a third party that can validate the code,

- WebRTC mode and block size

WebRTC empiric uses regarding packet loss possibilities advises a size of 1024B < payload of IP, UDP, DTLS, and SCTP protocols ~1150 B - unreliable mode, so, since the Tor protocol is fragmenting by blocks of 498B it seems logical to keep this size (we could change this but the system must be compatible with the Tor network again)

- WebRTC security issue

WebRTC is using self-signed certificates, the SDP (peer introduction) does include the fingerprint of the certificate, this is not enough to guarantee that there is not a MITM peer in the middle. Therefore the standards foresee to add another mechanism where the fingerprint of the DTLS certificate will be signed by a third party that knows the peer, typically a social network where the peer has an account, this is of course not considered here since the Tor protocol does this much better and eliminates this risk

It is well known that WebRTC will leak the IP address of the browser while browsing the web, this does not apply here since the application is not browsing but embedded inside the browsers

- Browsers issues

  This is a misdesign of the web that we have raised several times to the W3C: security procedures don't allow a https page to downgrade to *ws*, and a Convergence node can't use *wss* therefore a browser node can't use https to load the code, same happen with Service Workers

  Anyway, we have demonstrated in https://peersm.com/wallet a way to secure the code loading even with http

1.6 Open licenses

node-Tor is under a MIT open source license and is using Forge, sjcl, RSA and ECC in JavaScript, Browserify, Terser and other modules from us under a MIT license, we will use also Elliptic for Convergence

1.7 Exploitation potential

We believe that Convergence technology could be integrated natively inside browsers, like Brave is integrating IPFS, Webtorrent and the Tor project code (but the wrong way for us since it is using a localhost socks proxy)

This would allow an infinity of possibilities for private data transfer and storage, removing the inherent risk of loading the code as a webapp from the outside, and could probably revive some dead concepts like abandoned Google uproxy project

On another hand, this just simply gives the possibility for any, on any device to anonymize itself, which currently does not exist at all

1.8 Encouragements over years

node-Tor is well known and starred many times, many people in the past did request to move forward for some of their needs (see node-Tor issues) covered by the Convergence principles (including some proposals to include it natively in FF OS), the latest discussion happened with IPFS (https://github.com/ipfs/ipfs/issues/439) but surprisingly it's not a priority to implement privacy for IPFS, we keep the option to integrate one day node-Tor/Convergence with libp2p but the whole IPFS ecosystem is not trivial and it is still better to have a standalone Convergence module easy to use by everybody

1.9 Challenge

This proposal is ambitious but we are confident to break all possible barriers, probably some parts will be adapted during the project

We are confident that the funding will be well used and the project useful "to anonymize the world", **years after years the same needs keep resurfacing, all leading to a Convergence-like design**