# Indranet White Paper

Programmable onion routing distributed virtual private network protocol with anonymised payments to create scaling incentives.

David Vennik September 2022

> Markdown format of this document is created with Typora which renders the sequence and other graphs found in this document correctly. The PDF format may lag from the current state of the markdown document.

## Abstract

The state of counter-surveillance technologies has remained largely unchanged in the 20 years since the inception of the Tor network. The primary use case has always been obscuring the location information of users from clearnet sites, and the more it has been used for this purpose, the more hostile clearnet sites have become towards this network, due to its frequent use to launch attacks on web services.

With the increasing amounts of value being transported in data packets on the Internet since the appearance of the Bitcoin network, the need for eliminating the risks of geographical correlation between payments and user locations continues to rise.

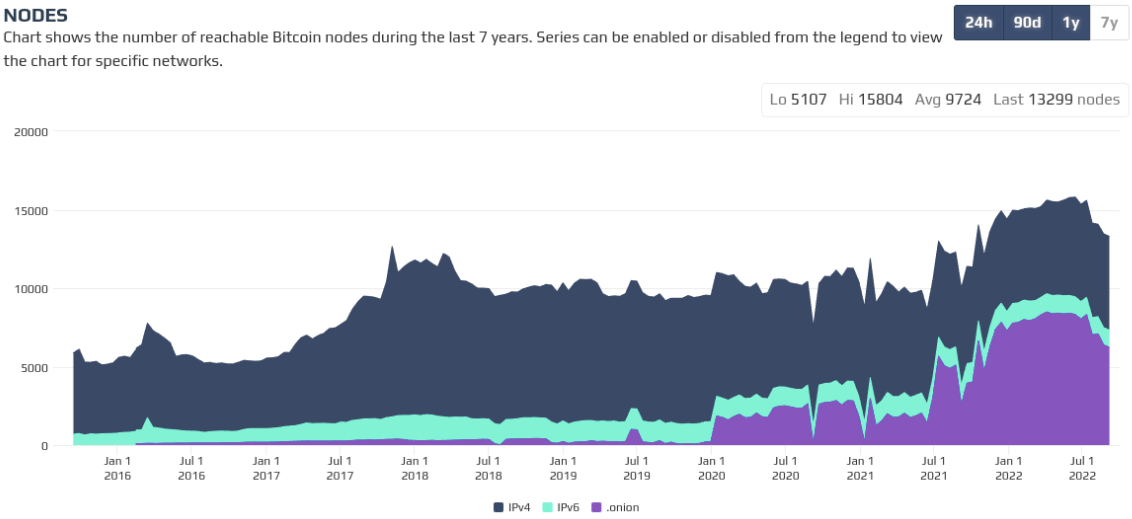
However, without any way for users to pay routers without creating an audit trail, the networks have a severe scaling problem in that in anonymising data, there is an increase in privacy with the larger number of nodes and users, and thus well heeled attackers have largely been able to keep pace and pluck off high value targets, such as the Carnegie Mellon University .

Thus, it is the central thesis of this paper to demonstrate how decorrelation between payments and session usage can be achieved and create a marketplace in routing services which can economically increase to a size that is beyond the capabilities of a state sized actor to fund an attack.

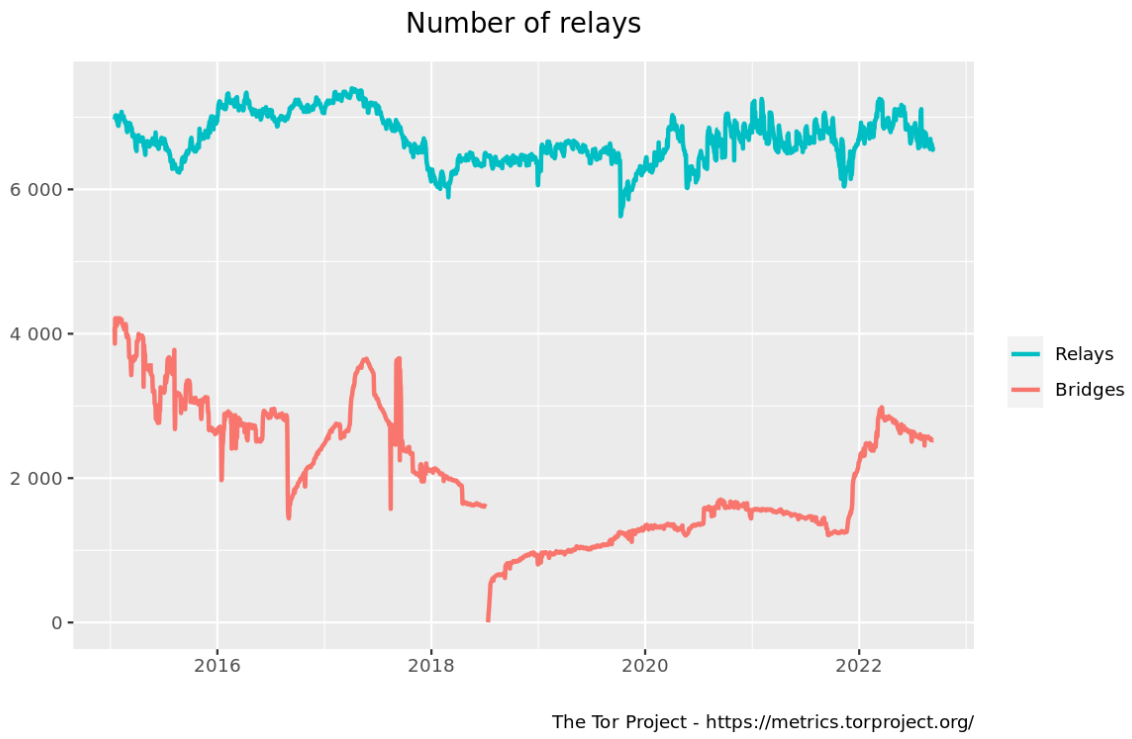
Indra creates mechanisms for anonymous purchase of chaumian vouchers used to initiate traffic sessions with router nodes, which then compensates routers for their running costs, and further, focuses on hidden services and Bitcoin/Lightning (and potentially other Bitcoin related systems) in order to reduce the attack surface from large actors who have thus no open justification for censoring the network.

## Tor isn't Scaling, but Bitcoin Needs Onion Routing

For comparison, this is Bitcoin's node count:




Versus Tor in a comparable period:




It is not hard to see: Tor is not growing, it's flatlined. Bitcoin is growing. Not only that, you can also see that onion routing is forming an increasingly large component of Bitcoin connectivity.

# Session Initiation

Initially, there was some idea that the Sphinx inspired protocol found in `lightning-onion` and used in `lnd` might be ok for this process, but it doesn't provide strong guarantees about progress of the execution of the purchase nor provide a secure return path for the token.

After that was discarded, there was the idea of using blind signatures and a purchase/spend flow, but that's unnecessary with this protocol.

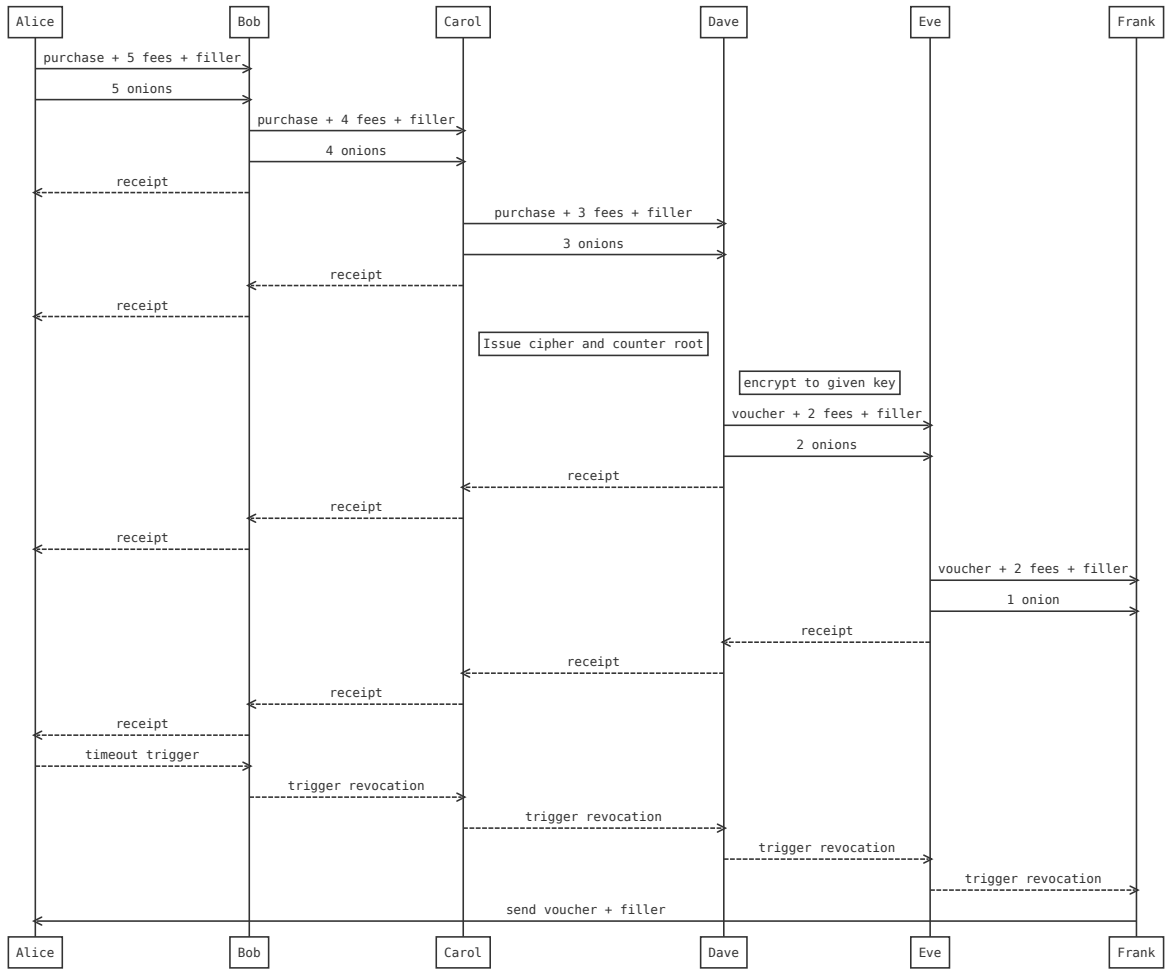
Indra enables full client side routing, aka "source routing". Once a session key is purchased, packets can be passed through the seller's node at any point in the circuit and varied even from one packet to the next.

## Session Purchase Flow

Nodes have identity key pairs, which they publish the public key for, which does not change. Secondary key is a short lived key with a short lifespan, short enough that there is little chance of these secondary keys being broken but long enough that every node on the network can acquire them, and will query for them after the time the old is set to expire, around half an hour.

This short lived key pair can then be used to construct an onion route packet which instructs each node to proxy forward a purchase that leaves a remainder for the node as fee, and to carry forward the onion message to the next node, as well as back propagating an embedded reverse route onion that routes a receipt of the payment, encrypted to a different key for each step, provided alongside the return route onion.

This is the top level view of the process:




If all acknowledgements do not get received within the purchase timeout the buyer triggers the payment revocation onions which reverse the payments for every hop that succeeded and returned an acknowledgement.

The payment revocations are carried in the forward packets but are locked by the keys that are sent in the revocation onion packet in the event of acknowledgement failure. Once the node has the revocation code they can cancel the forward payment on the channel and change it to paying themselves, forwarding the cancellation which allows the next step to do the same.

By using reverse onion acknowledgements, the payment onion forward propagates without direct interaction from the buyer, and any mischief on the path causing acknowledgement failure is punished by propagating a payment revocation which cancels up to the payment that lead to the wormhole attacker.

## Cipher and Counter Session Codes

The actual payload received from a purchase is an initial session cipher and the session hash chain root. The cipher is cycled after a number of packets are relayed, and encrypted to a provided return key in order to ensure the session has forward secrecy. The counter code is a hash chain that is hashed again for each subsequent packet, and allows the router to check off spent bandwidth for a given session.

With this arrangement the negotiation of key changes does not require interactivity and is carried along with the path of the routing created by the clients.

## Preferring Long Lived Nodes

There is an attack potential in the creation of nodes that attract purchases and then go offline before their services are delivered. For this reason, nodes will prefer to buy vouchers from the longest lived routers and new routers will have a longer lead time in acquiring regular work.

In addition, nodes will also propagate nodes to other nodes in order of their first seen and best liveness averages. Liveness averages are the result of cumulative ping and usage history. The node propagates nodes to other nodes as best first, thus increasing their chances of others buying sessions from them.

This functions as an eventual consensus for reducing the incidence of spending on traffic and not receiving the service for it. Better performing nodes get more advertised.

The reality of network service is always that one gets a tiny fraction less than what is paid for no matter what, connections fail, systems crash, and sometimes they are malicious. Some cost always prevents full efficiency but this is the unavoidable consequence of entropy in all systems.

# Prices

Indra provides a set of tools for maintaining bandwidth prices, that allows the accounting of their costs and maintains a margin of profit. Routers connection properties are entered in, the billing cycles are configured, and based on these cost inputs, and using exchange rate data, nodes then offer a given price which is current until changed again.

Routers with a pay per byte scheme will usually have higher cost per byte than nodes with pay per line capacity. In general the flat fee for line capacity is the best arrangement for the network anyway, since very often pay per byte services are higher latency mobile networks.

### Uniform Packet Sizes

One key thing to understand about Indra is that packets are all the same size, 8 Kb, which is also a common maximum size used in UDP based protocols, which is also the transport used by Indra. When a packet also contains acknowledgement onions, these consume more space and thus reduce the per packet payload, but are needed for path failure diagnosis or latency guarantee packets, and these return packets must also contain hash chain entries which indicate the current state of the session as well as proving payment, providing a direct indication of bandwidth status of the session.

### Clients

The clients have to pay these costs, so in the client they will see market pricing charted against network latency to the nodes, and can pick a threshold level where their routes will aim to target the price they are comfortable with. There will be a trade off between anonymity set size for a given price point and the smaller set (lower price) the greater the chance of route failure and the latency disruption that brings with it.

### Routers

When setting up a router, in the configuration a user can set the cost of bandwidth based on the parameters of their internet service costs, and define a margin. It can be the case that there is also bandwidth limitations, so this can also be part of this setting. If a node is running low on its time limited allocation (daily, monthly), it can raise the price of routing or simply cease to sell vouchers until the period elapses.

Ultimately, the idea will be that many users will be providing router service in addition to consuming it. In this way, the fees charged for the various elements of the service amortize the usage costs, and preventing the free rider problem, which retards network scaling.

# Routing Patterns

In the Tor network there are two predefined routing patterns used, the three hop to exit, and the 6 hop rendezvous path.

Because Indra is pure source routed, that is, clients construct the paths, the possibility of alternative constructions are entirely at the discretion and budget of the client.

## Reliability, Latency and Obfuscation

These three properties can be improved via the structure of the onion construction.

### Reliability

Reliability can be produced by expanding packets with Reed Solomon Forward Error Correction (FEC), and sending these packets in parallel. Any balance of N of M redundancy can be specified to the onion construction engine, most likely patterns of 2, 3 and 4 parallel paths would be used.

Fan out/redundancy patterns need to be understood by the endpoint, also. Thus, in the implementation, endpoint nodes will require queues to aggregate multiple message paths and unpack the single packet inside them.

### Latency

Latency can be improved also, by using parallel paths with two instead of three hops. Instead of, or in addition to redundancy, packet data is split into segments using Shamir's Secret Shares, and N of M must be received over a fan out/fan in two hop path for each side of the Rendezvous. The reliability can be tuned in parallel with this when packet drops occur.

Again, the endpoint of such a path must reconstruct the pieces from what packets it gets from the multiple intermediary routers. The pieces are collected and then when valid, forwarded onwards. The forwarding from endpoints is atomic by nature.

### Obfuscation

In addition to these simple parallel path patterns, it is also possible to open multiple sessions with a larger number of routers and vary the onion path in each packet, in addition to also potentially using short path for latency, in a way that further obscures the traffic's pathways.

### Notes

These features may not be as useful as they sound in practice, but the means to implement them should be available.

Note that parallel paths incur a proportional bandwidth cost, which should reasonably match up with the benefit of lower latency, increased reliability or higher security.

# Rendezvous, Forwarding and Inbound Routing

Because of the interference of routers and especially Network Address Translation, it can be that a node may not be able to directly receive inbound traffic. One of the big advantages of running a very large scale distributed VPN is that there is usually many infrastructure nodes that can proxy inbound access to nodes that cannot.

Peer to Peer network systems all have this difficulty of negotiating inbound routing in order to provide services. Thus, there is always a need to enable this proxying of inbound routing. There needs to be nodes on the network with routeable addresses, and these nodes get the benefit of the extra traffic for service provision within the network.

Normally this is done simply through Rendezvous routing, for hidden services, but because this inbound routing issue can be a problem, the programmability of the routing paths in the previous section also means it can be simple for nodes to create "open" rendezvous points that do not attempt to hide the location of the server. This still results in traffic on the network that adds the anonymity set for the anonymising services, and can be charged for the same way. There is a definite extra cost in enabling inbound routing, and this thus rationalises the extra earning capability for these nodes.

The user pays, and the user's client software can be programmed to perform the routing as requested. Thus, where normally a rendezvous is a 3 hop circuit, it can be 2 or 1, and effectively simply be forwarding, like a reverse proxy. The cost of each hop is a factor and thus if traceability is not a concern for the service (for example, its services are relatively low value) it can simply directly advertise rendezvous points of one hop and save on routing fees, and still increase the anonymity set of the network, and work around a lack of available inbound routing for the endpoint.

## Advertising Rates for Network Exit

As part of the peer to peer node advertising system, nodes also list available services that exit on their router. Normally by default this will be lightning and bitcoin nodes without charge, but it can be anything at all. These two services are required for all nodes to operate and the traffic is free because it is already known that the node has paid for the route leading to the exit, and thus is a reciprocal relation for all nodes. This provides privacy for location of traffic both on Bitcoin and Lightning networks, and is thus a collective benefit.

Tunneling out to clearnet can be made available too. In addition to the name of service, and port number related to it, a price per byte on this traffic can also be levied. The spending of sessions will proceed the same way as with others, through the use of hash chain counters based on the session seed. Purchases likewise will pass through the Purchase protocol outlined previously.

This fee is managed in the same way as simple internal network routing services, the buyer will want to use an onion route to purchase it if they are going to onion route to the exit as well. But again, one could skip the onion route for the payment, and go direct to the exit, and save on routing fees while adding to overall network traffic and the anonymity set.

Making the route construction modular, and enabling potentially arbitrary paths, and charging for this with potentially anonymised payments, means that the bigger picture for Indra is that ultimately it can become in band payment layer for ALL internet traffic, *in potentia.*

# Creating Circuits

In most descriptions of Onion Routing protocols, it is usual to discuss the circuit creation protocol. Indra does not need to use telescoping to form circuits. Nodes can simply send packets out and the header contains a hash chain value which indicates the position of the packet in the overall data allocation paid for.

The biggest benefit here is that this enables the recomposition of routes in the case one of the hops goes offline.

# Session Encryption

In order to reduce the potential for capturing encryption keys, all sessions, on each onion layer, use the Double Ratchet key negotiation algorithm. This key change will be triggered by the sender at a rate of around once per 4 seconds, to reasonably bound re-keying overhead. This will mean on average 1 key change per second per onion layer for exit paths and a little more for circular paths.

The individual message segments, composed of one or several sequential UDP packets, will be identifiable by a hash chain sequence which is used by the nodes and clients to keep an account of the bandwidth remaining in a session.

In order to enable all this, there is a difference in the signaling patterns in Indra. That is, all traffic must be prompted. All return packets come via codes delivered in an initial request packet outbound. This guarantees a very noisy pattern to routing data that does not easily correlate to data volume, and most especially not to endpoints of the paths, unless they exit, and the exit protocol is immediate. Exiting messages to Bitcoin and Lightning don't have immediate and direct outbound signals, they are mediated by validation steps and their outbound direction is random depending on the state of nodes' current peers.

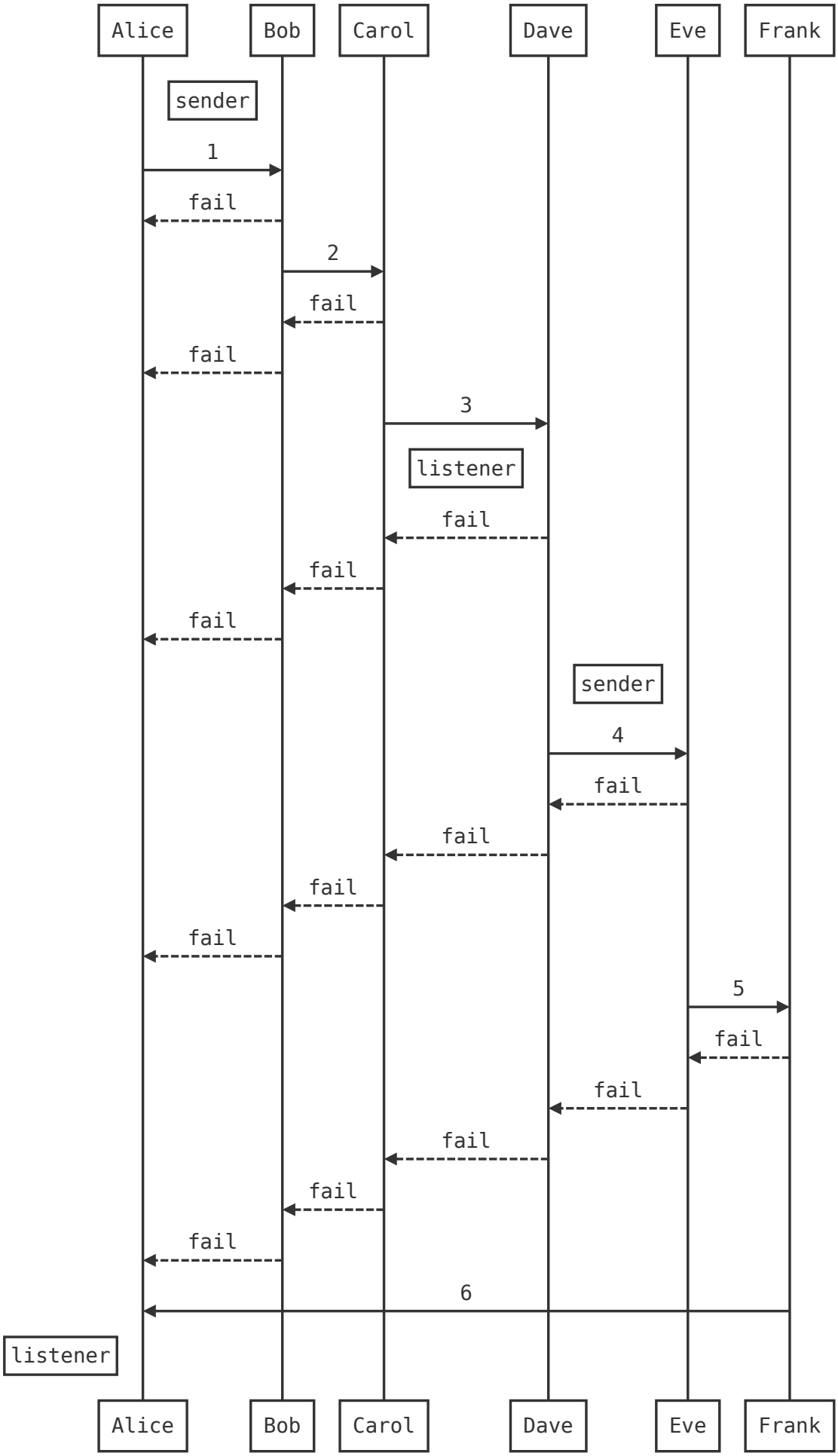
The ratchets are triggered by the forwarding on of new encryption values by the endpoint (hop 3) in a standard circular circuit. These are carried along with the forward return packets, encrypted only to the sender.

# Routing

Since other than Bitcoin and Lightning networks, Indra does not provide, by protocol, in its core implementation, exit nodes to tunnel out of the network, the third and equally important endpoint for a 3 hop circuit is a rendezvous.

On each side of the rendezvous, the client and the server create a 6 step circuit, not reusing the intermediary, so involving 5 other nodes, the middle point being the rendezvous, in a similar topology as the voucher purchase and session initiation.

The topology of the onion is the same as the Voucher Purchase and Session Initiation, except each layer only contains a session hash chain sequence, the onion route, a time to live, and the payload.


Alice
Bob
Carol
Dave
Eve
Frank
sender
1
fail
2
fail
fail
3
listener
fail
fail
fail
sender
4
fail
fail
fail
fail
5
fail
fail
fail
fail
fail
6
listener
Alice
Bob
Carol
Dave
Eve
Frank

The last 3 layers of the onion only provide the directions, but are encrypted to the sender's key, with the pre-negotiated hash chain sequence header and symmetric key for the next hop, concealing whether the packet is outbound or inbound, as their format is the same.

On the other side for the hidden service, the same pattern is provided, and circuits complete when one side sends and then the other side completes its half way journey, and vice versa. The rendezvous side has an encryption key provided where the two circuits meet the same node, which is used to encapsulate the packets that are sent on the other side via the forward path.

## Failure modes

Note the "fail" paths - these are messages that propagate backwards if the before the configured TTL the sender does not get an acknowledgement from the next step in the path. These allow the customisation for the type of traffic, for interactive versus bulk transfer. They enable only when timeouts are triggered, in order to diagnose the failure point in the path causing a transmit/receive failure.

If the failure return cascades are triggered, it means a node in the circuit is either offline or congested, and the onion routing will be reconfigured with a different node in the failure point, which returns as the hash chain counter value, which identifies the node that failed to relay within the time limit.

It is an extra cost in data overhead to encode the failure onions so they back propagate, but this small reduction of bandwidth is compensated by the way that with this information clients can construct a new path rapidly thanks to the pre-initiation of sessions in the purchase/initiate steps.

For latency sensitive applications, it is possible for every onion to carry a set of acknowledgement onions, which are sent backwards and unwrapped to notify the sender of successful route hops. This can also instead be only triggered after a circuit timeout, conserving bandwidth but providing path diagnostics to enable constructing of a route through a known live router.

## Client side path creation and benefits of gossip pattern

In order to enable this dynamic path changing, this network has a constant chatter. For every message that is desired to return to the client, there must be an onion constructed to shepherd the payload back from the rendezvous point or exit. Thus a ping will come with an onion that routes the pong back to the client, for example.

There is only a fixed size to the packets, so in order to perform a receive of data, the client must continuously send requests for more. There is no passive routing here because this would break the security of the paths provided by the Double Ratchet key negotiation. In the Tor protocol, these return paths are only created in the circuit extension stage, but in Indra every forward packet contains the round trip path to enable dynamic changes in paths and route around congestion and offline failures.

Obviously this incurs a substantial cost in AES encryption for the onion layers, and forces a constant signaling pattern, but this improves the anonymity set anyway. The volume/structure of onion circuit construction has a signature in the traffic, which Indra works around by separating the voucher purchase and session initiation, which means that large scale surveillance operators do not have very much useful timing data as packets are largely uniform in size and constant frequency. This also dictates that the maximum payload size an onion can carry is limited, in order to ensure there is a uniform packet size and frequency, removing timing data of what is carried by the onions.

Shorter routes can be programmed by users if their need for privacy of a signal is not high. Standard 3 hop paths implicitly are 3 times as expensive to use as single hops.

Note that return circuits the endpoint is provided the encryption keys for the three hops back, but these keys are built from the secret knowledge of the client from the session double ratchet, and change with every new message * cycle*. Thus they give no information to the endpoint about the path back to the client. The ratchet is not actuated every message, but periodically, with a data or time limit configured. The ratcheting is a separate message type to the out/inbound data delivery.

# Clearnet and Server Exits

In addition to creating rendezvous paths of arbitrary structure to rendezvous points, there can be "clear" exit points, which essentially amount to connecting to servers running on the same node as the router. This is by default lightning and bitcoind, but could feasibly be anything, the security isolation would be a factor of the protocol's structure and sensitivity and value of the data it handles. In simple terms, it is like port forwarding on NAT.

## Wireless Hot Spot Routing

A second use case that is not related is providing internet service through an " open" hotspot. The hotspot would refuse to relay normally, but has an Indra listener which can be negotiated with to send vouchers or LN payments and then becomes a usable access point. This ends the conflict between open hotspots and abusive users, as all users have to then pay for bandwidth, at minimum, as for one hop in a chain of the Indra network.

Where you can go from there, depends on the policy of the router, which will generally mean you are inside Indra using LN, Bitcoin or Indra messaging. Because the router is running LN and Bitcoin, it can freely provide sync data for the chain and for the user's channels, and thus combined with Neutrino, enables ubiquitous full SPV nodes on mobile devices. Users running these hotspots can also alternatively levy a higher charge for clearnet exiting, but as a general default, the purpose of this network is to enable access to hidden services, not tunneling. Hidden service access costs are lower because the endpoint cannot correlate to the entry point, and thus do not potentially present a security liability for the node operators.

Owners of such networks will then have special owner keys which let them send traffic on their own nodes without paying, and this service can then be exposed on unsecured wireless access points and become a direct source of income for the owner when others use it, while remaining secure by only carrying in-band traffic destined for other onion routers lightning/bitcoin and hidden services.

## Benefits for Other Peer to Peer Protocols: IPFS, Bittorrent

This also can ultimately facilitate more security for IPFS and Bittorrent networks as well, because everything adds to the anonymity set, if it is tuned to work with it well. One of the big problems with Tor is it is tuned to the TCP/HTTP use case and this is only part of network traffic usages. So additionally to LN/Bitcoin there can be specific "exits" for IPFS and Bittorrent ports.

Thus, as a later stage of implementation, these features should be included, and to enable it, an extensible proxy/socket protocol needs to be devised, built for the smallest use case set, and designed to be extensible for these several cases.

It is however of importance because when appliances are built to provide this routing service, they can have optional wireless interfaces that can run open hotspots that provide free seeding (with bandwidth limits) of locally available Bitcoin block data and mempool, and facilitate opening Lightning channels to enable payments.

### Enabling Mobile `neutrino/lnd`

This would mean devices with installed mobile clients, running Neutrino SPV nodes and on-net and Lightning wallet functions will always be able to connect and make payments anywhere a router is installed. This takes the burden of having internet connectivity away from the users, which can be very helpful for payment use case in that access to the payment network is free - and it is essentially quite a low cost additional to running the node, and the protocol compensates the users in accordance with their prescribed fee rates.

## Circuit Parameters

When creating connections, different types of traffic have different requirements for reliability and latency. As such, based on standard TCP Type of Service flags, circuits can be set to have constant acknowledgement cycling or only turn on acknowledgement seeking after a given timeout.

Conventional TCP services have a set of assumptions that don't hold well in the face of forward privacy onion circuit design. Thus, there are parameters that are used for Indra circuits:

- timeout - how long to wait for message return before retry.
- dead circuit retries - how many failed returns to count as triggering defining a circuit as dead, and start probing with acknowledgement back propagation onions to determine the router that is unresponsive, essentially a trigger that limits how long the connection stalls before probing the path. This then allows replacing the dead router with a working one in the circuit for a connection.
- latency guarantee - to do this, each hop will back propagate an onion that carries a packet acknowledgement. These decrease circuit bandwidth by consuming more of the fixed size packet, but prevent long delays from routers in the path failing, thus raising the effective cost of bandwidth.