**Skywire - Skycoin Meshnet Project**

Objectives:

* Increase broadband competition. Provide an alternative to existing ISPs. Span last mile.
* Enable communities to build ISPs running on user operated infrastructure.
* Start network deployment by July

Skywire is a new darknet protocol.

* Low Latency (as fast as TCP/IP and theoretically faster on native network)
* High Performance (designed for video, file sharing and high throughput applications)
* Privacy Preserving
* Supports Operation Over Wifi (Meshnet)
* Supports Clearnet Operation (Darknet/Overlay)

Skywire solves the incentive and leaching problem for network deployment

* Users receive Skycoins for providing network resources
* Users consume coins for consuming network resources

Skywire is open access.

* Anyone with a client is able to connect to any Skywire node
* Objective is to create a global open access meshnet

Skywire is privacy preserving

* The traffic passing through your node cannot be traced back to your IP address
* Nodes forwarding traffic can only see the previous and next hop
* Third parties passively observing cannot link individual packets to a stream or user
* Third parties and forwarding nodes cannot read contents of traffic

Skywire is designed for efficient ASIC, FPGA and hardware implementation.

====================

=== Skywire Meshnet ===

====================

The Skywire meshnet uses a source routed store and forward protocol.

**Routing: Overview**

The core of the overlay network is a series of Nodes.

* Each node is identified by a public key hash
* Each node receives messages and forwards messages
* Nodes receive coins for forwarding traffic

To communicate between nodes A and C, through node B

* Node A connects to node B and establishes a Route
* Node A extends the route to C
* Traffic sent over the route on A will arrive at C

In a route A -> B -> C -> D

* Nodes only know previous and next hop in route. C will know that the message passed through B and is addressed to D. C will however not know the identity of A.
* B cannot infer that A is the origin of the route.
* C cannot infer that D is the final destination of the message
* B and C cannot read the content of the message (end to end encryption)
* A passive observer not participating in the routing of a particular message cannot gain any information about the contents of the message (link layer encryption).
* Multiple messages from multiple routes to the same destination may be bundled, reducing ability of passive observer to perform traffic analysis.

In the simplest implementation a route is a 128 bit prefix. Each node reads the prefix and does a lookup in a table, to determine the next node to forward the packet to.

The source has complete control over routing.

* Each node can upgrade its routing protocol independently to suits their needs
* source can optimize network paths for lower latency network paths for VOIP or gaming
* source can optimize network paths for throughput for video and file sharing applications
* source can bundle multiple routes to destination for redundancy, reduced latency and throughput

Some applications will bundle multiple parallel routes at application layer for

* privacy (gatekeeper nodes, tor type gateway/anon services)
* throughput
* reduced latency
* redundancy

This is the core of the Skycoin overlay network. It is very simple, but extremely powerful. Technical and implementation details will be discussed later.

Skywire just prefixes a packet with a route ID.

* Routing is a very simple table lookup
* Overhead per packet is constant and does not increase for long routes

Notes:

- The destination does not know the identity of the origin. Identity is no longer at routing layer, but application layer. Identity must be confirmed through public key infrastructure.

- Man-in-middle attacks are not possible. Source can verify destination through their public key.

- Privacy is significantly improved from IPv4, where anyone handling a packet can see the destination, the source and contents of the packet.

- Performance is superior to IPv4/BGP because ISPs use hot potato routing

- End-to-End encryption eliminates packet injection attacks and spoofing. Spoofing traffic requires the private keys for each end of the connection tunnel.

- Encryption is fast. Objective is 10 Gb/s throughput on FPGA hardware, 200 Mb/s on ARM

**Incentives: Payment Protocol**

Nodes want to forward traffic and receive coins. This is the equivalent of “mining” in Skycoin and how many users will get their first coins.

Payments for transit should not reveal the identity of the source node. Skycoin will use blinded escrow payments through a third party until a better protocol is developed.

Each node in route records traffic and the origin node records traffic. They settle bandwidth payments periodically.

The origin holds coins in escrow with a third party. A pseudonym account is created with the third party. Each node can verify reputation of the origin and payment ability through the third party, without learning identity of party. To the third party, each origin will appear as multiple unlinked pseudonym accounts. Each transit node will appear as multiple unlinked pseudonym accounts.

Small transactions will be settled internally, in off-blockchain transactions. The off-blockchain transactions can be withdrawn into a newly generated, never used before address once the balance exceeds a threshold (currently 1 Skycoin). This reduces blockchain bloat and encourages microtransactions to be performed off-blockchain.

**Source Routing: Link Layer Encryption**

There is default link layer encryption between hops and default end-to-end encryption. A typical application will use link layer encryption, end-to-end encryption and then appropriate application layer encryption.

Encryption between nodes should be fast. FPGA implementations must support 10 Gb/s operation at line speed. ARM processors must be able to support 250 Mb/s.

The current best candidate is ChaCha20 with ECC secp256k1 ephemeral key exchange.

ChaCha20 only uses simple arithmetic operations, is faster than AES for embedded devices and is more resistant to timing channel attacks than AES.

A modern CPU can perform 6000 secp256k1 ECDH operations per second. Session key rotation should be once per second or twice the round trip latency between nodes. There should be a separate key for each direction of communication.

The previous session key should be accumulated into the secret received via ECDH.

The session key, established through public key cryptography (ECC) is used to encrypt communications using a faster asymmetric encryption algorithm (AES, ChaCha20). This the basic link layer encryption between nodes

Example Protocol: Nodes A and B

- Node A wants to generate session key for sending encrypted data to node B

- Node B has public key P, with private key p. P is a point on the ECC sep256k1 curve. p is a 256 bit integer. P is the basepoint b, raised to the power of p with the curve addition operation.

- Node A generates an ephemeral public key Q, with private key q. (Node A randomly generates a 20 byte integer. This is the private key q. Node A raises the base point to the power of q, to generate the private key Q, which is a point on the curve).

- Node A sends, P\*q (the point on curve P, which is B’s public key, raised to power of q)

- Node A sends P to node B

- Node B receives P and computes P\*q, Node A can compute p\*Q, which are equal. This is the shared secret, which is hashed to generate session key.

- P = b\*q, so P\*q is equal to (b\*p)\*q. P\*q = (b\*p)\*q = (b\*q)\*p = Q\*p, since Q=b\*q. A knows q,Q and P, and B knows p,P and Q. So A and B can both compute P\*q and Q\*p and use this as their secret. However, a third party does know know the private keys q for A or know the private key p for B, so a third party cannot compute this “secret” and therefore cannot read anything encrypted under the secret.

- Node B confirms receipt of the session key update. Node A begins transmitting under the new session key as soon as the confirmation is received from B.

- Node A, sends messages to Node B by encrypting them using ChaCha20 using the session key, as the asymmetric encryption key.

Possible Improvements:

- Frequent session key updates. ECDH key exchange every few seconds or minutes

- Hash old session key with new ECDH secret to generate new session key

- Add nonces to packets and hash secret into nonce to generate key for each message. Same key is never reused. Reduces impact of known plaintext attack.

- Eliminate known plain text in messages.

- Pad messages to multiples of 16 or 32 bytes

**IPv4 Gateway: Bypassing Existing ISPs**

Many people have only a single choice for ISPs. This briefly describes how Skywire can increase competition

Some applications can run natively on the Skywire address space. Some applications such as Bittorrent, file syncing and communication applications strongly benefit from the Skywire infrastructure and will be modified to run natively on it.

Legacy applications, such as Netflix, Facebook, Twittter require a network gateway interfacing the Skywire overlay network with IPv4 and IPv6 networks.

A user chooses a Skywire gateway running on a server in a local colocation center. The users IPv4 traffic will be tunneled through the gateway (similar to a VPN). The users IP will appear as the IP of the gateway server . The server has a gigabit connection to multiple fast internet backbones, on providers which do not rate limit Netflix. The user has multiple choices of providers for Skywire IPv4 gateways. The gateway provider will be paid in Skycoin on a metered basis.

The Skywire node in the users home connects to the gateway over all possible routes. The Skywire node tunnels the IPv4 traffic from a router to the gateway in the colocation center. The IP address of the gateway node is the IP address that appears to the user.

Example1:

A user has a 10 Mb/s cable modem. They install a Skywire router. The router is plugged into their computer, into a Skywire Wifi node and into their cable model. The router is configured with Skywire as a IPv4 tunnel. They plug their computer into the router .

The Skywire wifi node connects to their neighbors Skywire node over wifi, which is connected to a 10 Mb/s cable modem. The neighbor also has a 200 Mb/s 5 GHz wifi with directional point-to-point antenna connected to a business running a Skywire wifi node down the street.

The users Skywire router does a recursive breadth first search for nodes with clearnet connectivity and establish routes over

- his cable modem

- wifi -> his neighbors cable modem

- wifi -> 5 GHz point to point -> 100/30 Mb/s business/fiber drop

The user will be able to access and aggregate bandwidth over all routes, in connecting to the IPv4 tunnel. In communities where aggregate bandwidth and reliability have reached a certain level, the user no longer needs the cable modem for connectivity.

Example2:

The business down the street has a 100/30 Mb/s fiber drop with an SLA. The business pays a fixed rate for internet. Any bandwidth they do not use is lost. The business hooks up a Skywire router. The router has 3 ports. The first port is their WAN connection, the second port is their internal network, the third port goes to their Skywire wifi nodes on roof. The router buffers and prioritizes traffic on the internal network and allocates unused capacity to the Skywire traffic. The operator receives Skycoins for transit which, subsidizes the cost of the fiber drop.

**Skywire Daemon Implementation:**

<code example>

<link to source code>

Skywire Daemon Services Architecture:

* Each Skycoin node has a Secpk256k1 pubkey
* Each Skycoin node has a Skycoin address identifying it. The address is a hash of the node’s public key. This public key hash is the equivalent of an IP address on the network.
* Each Skycoin node has a connection pool of peer it is connected to. These can be peers over TCP, UDP clearnet connections, physical connections through direct Ethernet and Wifi peer (meshnet operation). A connection can also be a “virtual connection” which is tunneled through a physical or clearnet connection and will be described later.
* Each connection instance with a peer has “channels”. A channel is a 16 bit integer which is similar to a “port” in TCP.
* All messages sent and received have a 32 bit length prefixed and 16 bit channel.
* Channel 0 is reserved for communication between Skywire Daemons, exposing meta-information about the services running on the Daemon and other data required for network operation.
* A Skywire daemon may expose “services” on a channel. A service is a process that handles data messages received on a channel and originates data messages addressed to remote peers and services.

Example Service: Blockchain Sync

This example is for the Golang daemon implementation, but the daemon architecture is language agnostic.

You want to sync two different personal block chains of people with public keys A and B. You initiate two “blockchain sync service” instances, configure them with the respective public keys and associate them with the Skycoin Daemon. These services run on your local daemon, each on a particular channel.

Finding Peers:

The blockchain sync daemons the hash the public key and do a DHT (Distributed Hash Table) lookup to find other peers syncing the blockchain. Once peers are found, the peers can be introduce each other to additional peers through PEX (Peer Exchange).

Sending a Receiving Messages.

Services on creation register a list of messages they respond to and can send. Messages are Golang structs. The message struct data is filled out and then sent. The data arrives and the .Handle() method is called on the corresponding message struct.

**Multi-Home Routing and Link Aggregation:**

If you have a 2 Mb/s cable modem and your neighbor has a 2 Mb/s cable modem and each of you are running Skywire nodes, then your Skywire node can connect to his node and aggregate the bandwidth across both connections. Packets may now take routes through your cable modem and routes through his cable modem. The cable modems are the choke point. To get a 4 Mb/s connection, traffic has to pass in parallel paths through both modems.

Applications like Bittorrent will be able to aggregate bandwidth across all available connections, because they natively open up a large number of connections, which will take district routes.

**Meshnet Routing: Store and Forward**

For nodes at the edge of the network communicating over mesh, there are a few issues.

If you have an eight hop network over Wifi and 50% of packets drop on each hop, then only 1 in 256 packets will make it through. Packet drop is normal on Wifi, but traditional TCP treats packet drop as congestion and throttles the connection speed back.

At the network edge, Skywire will use store and forward along routes. This imposes a memory requirement on Skywire nodes, but significantly improves network performance.

For the route A->B->C

- Each route has a buffer,

- Each node keeps sending the messages until they are received and acknowledged

- If the Buffer from B->C is full for the route, then A will know this and will stop transmitting data until the sent data has been acknowledged there is room in the buffer

Therefore there are two acknowledgements between nodes at the link layer. One acknowledgement is an affirmative acknowledgement that transmitted data segments were received. Another is an acknowledgment that data from the buffer has been sent to and acknowledged by the next node in the route.

**Store and Forward: Capacity Utilization**

In traditional IP networks, as network links are utilized towards capacity the efficiency drops. A network running at 80% capacity faces the risk that a short term burst of data causes a router to go over capacity and network packets are dropped.

TCP interprets dropped packets for any reason as congestion and throttles back speed. The dropped packets also require retransmission under TCP and introduce latency as the application waits for timeout and retransmission before it can process the rest of the packet stream.

With store and forward operation, the route buffers fill up and then nothing happens. When the buffers fill, the incoming node stop sending data until notified of space in the buffer.

This store and forward operation is especially important for practical Wifi meshnets. There are only three non-overlapping channels in the 2.4 GHz band. Packet loss increases very quickly and very early compared to the bandwidth saturation point for a Wifi networks. Wifi packet loss is inevitable and does not reliably indicate congestion or capacity limits.

Store and forward allows us to run the Wifi nodes at full capacity and saturate all available bandwidth, without triggering TCP congestion controls.

Practical networks will require

- Software Defined Radio

- MIMO

- Beam Forming

- Directional Antennas

- Cooperative temporal and geophysical coordination of transmission time, broadcast power and frequency usage

- 801.11af whitespace frequencies

**Store and Forward: Examples**

Each node, for each route keeps track of

- buffer size for receiving node for route

- predicted buffer size (acked and unacked datasegments transmitted)

- acked buffer size

- offset, size and sequence of each transmitted message that has not been acked

- circular buffer of bytes of outgoing datagrams that have not received an ack

A data segment on the link layer may contain concatenated messages, from multiple routes addressed to the same node. This frustrates traffic analysis and improves performance by allowing larger datagrams on networks which support higher MTUs.

For each transmitted message, there are two acks. The first ack is that the datagram has been received by the next node in route. This is an ack for the datagram, which may contain multiple messages, each corresponding to different routes. After receiving this ack, the node no longer needs to retain the datagram. If the datagram is not acked, then it needs to be resent.

The second ack are updates about remaining free bytes in the incoming buffer for a route. If the free bytes in the buffer for a route is large enough, then additional messages for that route can be transmitted.

Another possible approach, maintains a buffer per sender instead of per route, with the receiver sending block messages to the sender for congested routes. This reduces the number of route hash lookups required by sender and is something that may have to be experimented with.

Example1: normal operation

* Route: A->B->C
* B has 1024 KB buffer for route
* A sends 512 KB to B
* B Acknowledges the 512 KB to A
* < A receives the Acknowledgement (and clears first 512 KB, no longer needs to be stored) >
* B forwards 512 KB to C
* C acknowledges receipt of the 512 KB
* C acknowledges to A that the 512 KB has been forwarded

Example2: congestion

* Route: A->B->C
* B has 1024 buffer for route
* A sends 512 KB to B
* A sends 256 KB to B
* A sends 256 KB to B
* < A stops sending because pending, is already enough to fill buffer at B >
* B acknowledges to A the receipt of 512 KB and 512 KB
* B sends 256 KB to C
* C acknowledges receipt of 256 KB to B
* B acknowledges to B, forwarding of 256 KB to A
* < A can now send, up to 256 KB additional KB >

Data is assumed to be received in the order sent for Wifi and direct ethernet connection

Example3: dropped packets

* Route: A->B->C
* B has 1024 buffer for route
* A sends 512 KB to B
* A sends 256 KB to B
* B acks the 256 KB
* A infers that the 512 KB was not received
* A retransmits the 512 KB
* B acks the 512 KB
* <B can now continue sending stream in order to C>

**Store and Forward: Bandwidth Latency Product**

In store and forward a storage requirement is imposed on the transmitting node equal to the product of the round trip latency and the transmission rate times the round trip latency. 1 GB of ram is enough for 8000 ms round trip latency at 1 Gb/s transmission rate.

Store and forward should be default, but optional.

**Store and Forward: Capacity Utilization, Quality and Service**

Video, Audio and file downloads are buffered. Absolute averaged throughput over a time window of seconds matters, while latency is irrelevant. Other traffic such as website requests, video game and voip is real time and should be delivered as quickly as possible.

With two quality of service levels, “Real Time” and “Bulk” we can transmit VOIP, website and `video game traffic first, reducing latency for this traffic. Latency insensitive traffic such as video, music and file sharing would only flow over link after real time traffic buffer is empty.

We are able to utilize links at near 100% of capacity while, lowering latency for real time traffic. Therefore we propose supporting two quality of service levels for routes.

**Source Routing: Multi-Route Mobile Connectivity**

If connections between nodes are stable, low latency and have high bandwidth then a single route is sufficient for most applications. Some applications like Bitorrent, open a large number of connections and natively can use bandwidth across all available routes.

If the links between nodes are slow, unreliable or connectivity is changing, reliability and performance requires traffic to be multiplexed over multiple redundant routes.

If a Skywire node running on a cell phone is in a car driving down the street, the networks that are accessible will change. Network nodes will come into range and other network nodes will leave range. The node should have continuous connectivity at the application layer, even as the physical connections are created and destroyed.

One approach is choosing a set of reliable nodes on the network backbone as termination points for a route and then proxying the traffic through these nodes, over a set of multiple short term routes.

**Source Routing: Multi-Route Reliability**

If links are unreliable or have highly variable latency, it is desirable to encode application data over multiple paths, such that the data can be recovered if data from any of the paths is received. Fountain coding and other encoding methods exist which may be applicable here.

**Source Routing: Guard Nodes**

For privacy, if a user wants to further weaken linkability between their Skywire node address (public key hash) to their IP address, they can destinate a fixed set of nodes that are advertised as being transit points for traffic destined for their address or act as required nodes on a route from their address.

**Source Routing: Limitations of BGP**

Border Gateway Protocol, the current dominant routing protocol, handles the routing problem by not keeping any state for packets. Instead BGP, allows each network to create a series of ad-hoc rules for each of its routers which look at the source and destination of a packet and decide which network interface to forward the packet to. Routers message each other with connectivity information and another routing algorithm is used for routing within a network domain.

BGP is designed to interface a series of independent autonomous networks. BGP has a homogeneity assumption, the routing within an autonomous domain is assumed to be centrally managed and highly reliable with homogenous routing within the domain. Mesh network and community ISPs will be ad-hoc with heterogeneous device connectivity and routing.

Connectivity in mesh networks, ad-hoc configurations and densely interconnected networks with redundant multi-home routing paths completely violate the hierarchical assumptions BGP.

BGP has several issues that a next gen protocol should address

* BGP is not self configuring. BGP based networks require extensive technical expertise to configure and operate
* BGP systems often require manual configuration to route around damage and are not resilient against bad configurations
* BGP requires manual creation of ad-hoc route filtering rules and increasing complexity for networks with multi-home connectivity
* BGP networks require highly centralized planning
* The NSA has exploited flawed in BGP to route targeted traffic to interception points
* The assumptions of BGP are becoming increasingly strained, especially for ad-hoc, mesh and mobile networks
* The hierarchical, single path assumptions of BGP make implementation of multi-homing and other next-gen networking requirements extremely difficult
* BGP suffers severe issues when network links are unreliable, such as route flapping.
* BGP routing table size grows exponentially as interconnected subnetworks proliferate.
* Multihoming causes a massive explosion in BGP routing table size.
* BGP has difficulty with load balancing and multi-home routing. BGP limits the ability in practical networks to take advantage of parallel connectivity between locations.
* BGP creates an incentive for ISPs to dump network traffic on to other networks as quickly as possible (“Hot Potato Routing”), reducing performance and increasing latency

There is no alternative to BGP. BGP is the best solution within its design constraints.

The successor to BGP must

* be non-hierarchical
* be self-configuring (zero-conf)
* operate well with dense ad-hoc, redundant interconnection between networks

**Virtual Routes: Skywire Network Topology at Scale**

The Skywire routing implementation requires a node to maintain information for each route that passes through it. Individual nodes are unable to handle hundreds of thousands of individual routes and scalability is achieved through another mechanism.

Skywire is experimenting with a non-hierarchical, self organizing routing that natively supports multihoming and non-hierarchical network topographies while scaling efficiently.

Skywire minimizes network diameter as the network scales through the use of virtual routes. Virtual routes allow thousands of connections to be bundled over a high bandwidth backbone connection with the overhead of a single route.

A “virtual route” creates a tunnel over an existing route

A -> B -> C -> D

The virtual route appears as A->D. B and C may be high bandwidth long distance connections. B and C only incur the overhead of a single route, while A and D incur the overhead of maintaining the routes on the A->D tunnel.

The virtual route may contain traffic from hundreds of bundled routes from A to D, while B and C only experience overhead of a single route. The virtual route may further, bundle multiple redundant network paths between the origin and destination for performance, throughput and redundancy.

Virtual routes allow network capacity to be clustered roughly hierarchically with nodes at each layer having a constant fan in and overhead.

Nodes at the network edge feed into aggregation nodes. Edge aggregation nodes are connected to high bandwidth intradomain transit and feed into gateway nodes which interface between networks. Gateway nodes feed into high bandwidth and long-haul transit.

Virtual routes are a representation of existing interdomain routing relationships, which natively support

* non-hierarchical routing (data centers)
* multiple-homing
* dense network interconnection between domains at different levels of hierarchy
* multi-path routing within and between network domains

Virtual routes obey a triangle equality. If the cost of a route A->B, is C(A->B) then

C(A->B->C) >= C(A->B) + C(B->C)

Origin preference for low latency, low cost, and low hop routes, creates economic incentives to create an efficient network topology. The network is non-hierarchical and self-organizing. The virtual routes that are created are route summarizations that naturally reflect the flow of traffic.

In BGP, networks try to get rid of traffic as quickly as possible (hot potato routing). In Skywire, networks compete to provide transit (to receive coin incentives). Skywire clients will preference, low cost, low hop count and low latency routes. Networks with direct long haul capacity between source and destination have lower latency and lower hop count and therefore receive preference.

For efficiency, the bandwidth capacity and fan-in (number of routes each virtual route is bundling) at each level of the network hierarchy must be constant, in order to achieve a constant network diameter and logarithmic routing table growth in the number of hosts.

**Source Routing: Virtual Routes, SONET Topology**

A multiple-input, multiple-output virtual route, may be physically implemented as a SONET ring, with Skywire nodes in each city the SONET topology passes through. The Skywire nodes act as a gateway router between the Skywire network and the SONET topology.

Nodes are able to queue up large datagrams concatenating multiple messages from the same source to the same destination for efficiency.

The message enters the Skywire node of the SONET ring at a colocation center in one city. The message destination or route is read and the message is encoded for transport over the SONET segment. The message arrives at the destination Skywire node on the SONET segment and continues on its path.

A multiple input, multiple output virtual route is therefore a list of Skywire nodes, with a transit cost, describing a SONET ring or fully connected topology, where any node in list has transit to any other node in the list.

**Source Routing: Asymmetric Connectivity**

Next gen wifi systems will have 4x4 and 8x8 antennas in a phased array MIMO arrangement. Such systems are able to project highly focused directional beams. These systems significantly increase the power and signal strength at the receiver, but do not symmetrically improve antenna gain for return signals.

Similarly, a high powered, amplified wifi signal through a directional antenna may be received at a site fifteen miles away, but reception of the signal from the site cannot similarly be amplified as easily as power can be boosted in transmission.

We propose, Asymmetric routes, for situation where messages can be received by a node, but where the node cannot directly communicate back. In an asymmetric route confirmation messages are relayed over the network by a route, enabling full utilization of asymmetric connectivity over one way communication channels.

Situations where this will become increasingly relevant

- Rural SONET arrangements with amplified Wifi over directional antennas

- Urban connectivity between highly directional and non-directional antennas broadcasting at the same power levels

- Concrete penetration in 802.11af systems

- Non-line of sight LiFi propagation can transmit over 200 Mb/s but its highly asymmetrical

- RONJA type Li-Fi systems have theoretical capacity limits over 10 Gb/s line-of-sight and there is cost/setup advantage for asymmetrical connectivity

Utilizing asymmetric connectivity and routes allowing only one way direct data transmission between nodes has several advances, especially for rural developments and reducing cost of integrating high capacity of next-gen technologies.

**Source Routing: Route Discovery**

IPv4 Gateway and mesh nets for community ISPs, only require breadth first search over paths to clearnet connectivity. The best, most reliable, highest throughput routes have very small depth. Therefore we consider routing solved for this case. Will look at general routing later.