# ELECTROLIPS WORMHOLE NETWORK & INFORMATION FOUNTAINS

**ME: / CH@: — CHRONOLOGICAL VISIBLE CONVERSATION**

Sepia-style document with black Courier lettering. Includes all visible messages, code, labeled image placeholders, an image URL inventory, original citation markers, and related source URLs.

49 visible messages • 24 ME: • 25 CH@:

Original grammar, spelling, punctuation, capitalization, and headings retained.

Inline chat visualizations are represented with editorial [PLACEHOLDER] notices.

Citation markers are preserved exactly as markers; their original individual web links were not available.

**CONTENTS / CLICK TO JUMP**


Message 01 — Opening wormhole and initial iframe

Message 09 — Numbered portal operator

Message 11 — Decentralized roaming switchboard

Message 13 — Existing moving relay networks

Message 17 — Foreign buffers and passive listeners

Message 25 — Multiple communication transports

Message 27 — 16,000-byte packet sizing

Message 31 — 1,600-byte message format

Message 33 — Listener program footprints

Message 37 — Visual Information Fountain

Message 42 — Simultaneous independent fountains

Message 44 — Earlier recap request and delivery

Message 49 — Request for full verbatim transcription

APPENDIX A — IMAGE INDEX

APPENDIX B — ORIGINAL CITATIONS

APPENDIX C — URL REGISTER


## TRANSCRIPTION NOTE

The full natural-language transcript was transcribed from the conversation visible in this chat. At places where the original assistant sent visual widgets, SVG, a clickable dialer, or image galleries, those elements are marked with explicit editorial placeholders, not represented as exact original source bytes. Citation markers are visible in the transcript in [CITE …] notation. An earlier assistant message cut off mid-sentence is preserved without inventing a continuation. Previous assistant progress messages and archive file links are included.

**01 ME:**

How do we make a wormhole that takes a user from one web page to another web page live without using the url ?

[RETURN TO CONTENTS]

**02 CH@:**

You can build a **live webpage wormhole** using an HTML portal. A visitor clicks a button, and another functioning webpage appears inside the current page—without navigating away or typing a URL.

The technology is an **iframe with a full-screen overlay**.

### How the wormhole works

1. Visitor clicks **OPEN WORMHOLE**.
2. A portal opens over the current webpage.
3. The destination webpage loads live inside the portal, including its interactive functions.
4. Visitor clicks **CLOSE WORMHOLE** to return instantly to the original page.

The browser still needs an internal destination address, but visitors never need to enter it, and the original page's address bar stays unchanged.

### Drop-in HTML wormhole

This example uses your Electrolips fashion directory as the destination.

```html
<!-- ELECTROLIPS LIVE WEBPAGE WORMHOLE -->

<button onclick="openWormhole()">
  OPEN WORMHOLE
</button>

<div id="wormhole" style="
  display:none;
  position:fixed;
  inset:0;
  z-index:99999;
  background:#eee0c5;
">

  <button onclick="closeWormhole()" style="
    padding:12px;
    background:#111;
    color:#eee0c5;
    font-family:Courier New,monospace;
  ">
    CLOSE WORMHOLE
  </button>

  <iframe
    id="wormholeFrame"
```

```
    title="Electrolips Live Portal"
    style="
      width:100%;
      height:calc(100% - 45px);
      border:0;
      background:white;
    ">
  </iframe>
</div>

<script>
const portalPages = {
  fashion: "/index.html/fashion/electrolips2/"
};

function openWormhole() {
  document.getElementById("wormholeFrame").src =
    portalPages.fashion;

  document.getElementById("wormhole").style.display =
    "block";
}

function closeWormhole() {
  document.getElementById("wormhole").style.display =
    "none";

  document.getElementById("wormholeFrame").src =
    "about:blank";
}
</script>
```

This assumes that directory is accessible at that path.

### Three possible wormhole designs

| Type | Function |
|---|---|
| Portal window | Opens another live webpage inside your current page |
| Full-screen wormhole | Replaces the visible page without changing the top-level address |
| Two-way wormhole | Both pages exchange messages and synchronize certain functions using JavaScript |

**Important:** Some external websites prohibit being displayed inside an iframe. Pages hosted on your own Electrolips website are generally easier to integrate, provided their security settings permit embedding.

For Electrolips, I would use the **two-way wormhole** approach if you want interaction between the pages, rather than simply displaying one inside the other.

**Are you trying to connect two pages on electrolips.online, or connect your page directly to another website without visibly sending the visitor there?**

[RETURN TO CONTENTS]

**03 ME:**

But that still uses a url no?

[RETURN TO CONTENTS]

**04 CH@:**

**Yes.** The iframe still uses a URL, even though the visitor doesn't see it.

To connect two live webpages, the browser needs some way to identify and reach the destination.

There are alternatives to directly using a URL:

1. **WebSocket connection:** Exchanges live data between pages, but still requires a network endpoint.
2. **Server-side routing:** Uses an internal page identifier instead of exposing the destination URL in the HTML.
3. **WebRTC:** Establishes a live peer-to-peer data connection after an initial signaling process.
4. **JavaScript portal:** Switches between already-loaded page components without making another navigation request.

**The distinction:** We can eliminate visible URLs and direct URL navigation, but we cannot eliminate network addressing when connecting to a separate remote webpage.

Are you trying to create a connection between two different websites **without using HTTP/HTTPS addresses at all**, or simply hide the destination address from the visitor?

[RETURN TO CONTENTS]

**05 ME:**

Not what im looking for. Lets think about this.

[RETURN TO CONTENTS]

**06 CH@:**

Let's approach this as a new type of connection rather than a conventional hyperlink.

**The question is whether we can transfer a user's live interaction from Webpage A to Webpage B without navigating to B's URL at all.**

Consider three separate mechanisms:

1. **Address-based navigation:** The browser requests a webpage using its URL. This is the conventional internet model.

2. **Connection-based transfer:** Two servers establish a persistent connection. Webpage A transfers the user's interaction to a live session hosted by Server B, using a session identifier rather than a destination webpage URL.

3. **Live-state transfer:** Instead of loading Webpage B, its interface and application state are delivered through an existing connection and rendered inside Webpage A. The user interacts with B without navigating there.

> [IMAGE PLACEHOLDER I01 — Inline SVG originally shown here: Webpage A → live session connection → Webpage B; label “No browser navigation to B”. Original SVG had no external URL.]

The third approach is technically possible using remote rendering and real-time communication. It doesn't require exposing or navigating to B's URL in the visitor's browser, although network addressing is still necessary somewhere in the system.

**But there's an important distinction I want to establish:**

Do you want the visitor to actually *leave Webpage A and arrive at Webpage B*, or do you want the visitor to *remain on A while interacting directly with a live B*?

Those are two substantially different engineering problems.

[RETURN TO CONTENTS]

**07 ME:**

Maybe. Theirs a hosting site i can make with a direct serial port entry virtual

[RETURN TO CONTENTS]

**08 CH@:**

That introduces another possibility: **a virtual serial-port gateway between two web hosts.**

Instead of sending a visitor to another URL, we could create a hosting system that accepts connections through numbered virtual ports.

For example:

- **Host A:** Visitor enters the wormhole.
- **Virtual Port 7001:** A persistent communication channel.
- **Host B:** Receives the connection and supplies its live application interface.
- **Visitor:** Interacts with Host B while remaining on Host A.

The key distinction is that we would be **transferring application data through a communication channel, rather than navigating to a destination webpage.**

> [VISUAL PLACEHOLDER — “Proposed architecture”: Visitor — Host A ⇄ Virtual serial gateway ⇄ Host B. Originally displayed as a chat diagram.]

There is one technical limitation: ordinary browser serial access is designed for devices connected to the user's computer, not arbitrary remote hosting ports. [CITE turn976333search0; turn976333search7] A server-side virtual serial gateway could overcome

that by tunneling serial data over a network connection. A WebSocket could carry the browser-facing portion. [CITE turn976333search5]

This would still use network addressing internally, but not necessarily a destination webpage URL.

**One question:** When you say *virtual serial port entry*, do you mean an actual software COM/TTY serial port installed on the hosting server, or a new numbered communication port that we design ourselves?

[RETURN TO CONTENTS]

**09 ME:**

Like a operator that shoots strait to the other portals number. A internet inside the internet

[RETURN TO CONTENTS]

**10 CH@:**

You're describing **a second internet operating inside the existing internet**, using numbered portals instead of website addresses.

The central component would be a **digital switchboard operator**, similar to a telephone exchange.

## Electrolips Portal Network — Concept

[IMAGE PLACEHOLDER I02 — Inline SVG originally shown here: four numbered portals, 1001, 2002, 3003, 4004, connected to central “OPERATOR / Number exchange / Session switching”; caption “Private virtual addressing network”. Original inline SVG had no external URL.]

### How it would function

1. Every participating hosting site receives a unique **portal number**, such as `4004`.
2. Each host maintains a live connection to the Electrolips operator.
3. A visitor enters `4004` into a portal dialer.
4. The operator locates that portal and establishes a communication session.
5. The destination portal sends its live interface and receives visitor interactions through that session.

The operator could connect users directly when technically possible, instead of relaying every message.

## Experimental portal dialer

[INTERACTIVE PLACEHOLDER — In-chat demonstration dialer with default portal number 4004, keys 1–9, CLR, 0, backspace, and a “Design connection to 4004” button. This was an in-chat UI concept, not an active network connection.]

## What makes your idea different

This is technically called an **overlay network**. The underlying internet transports information, but your operator manages an independent numbering system and connection protocol.

A portal could be identified as `EP-4004` rather than `https://example.com/page.html`.

The system would still depend on IP networking underneath. However, neither visitors nor participating applications would need to use destination webpage URLs to establish individual portal sessions.

**The significant design decision:** Should the operator merely connect two portals and then get out of the way, allowing them to communicate directly?

That would make it closer to a true virtual switching network than a conventional website hosting service.

[RETURN TO CONTENTS]

**11 ME:**

Yes but lets scramble the codex in the main opperator switchboard virtual then shot it to receivers that listen for it on open communication channels and puck it up then direct it where it needs to go. Without a central hum just orograns running amd moving through virtual servers abd space

[RETURN TO CONTENTS]

**12 CH@:**

Yes. You're describing a **decentralized, encrypted portal network** in which there is no permanent central switchboard.

Instead of one operator controlling every connection, we distribute the operator's functions among programs running on participating virtual servers.

I would call this the **Electrolips Distributed Wormhole Network (EDWN)**.

## 1. How your system would work

1. **Scramble:** A sending portal encrypts its information and assigns it a temporary routing code.
2. **Transmit:** The information enters an open communication channel shared by participating receivers.
3. **Detect:** Listening receivers recognize routing signals without needing to decrypt the message.
4. **Relay:** Receivers pass the information through other virtual servers toward the intended destination.
5. **Deliver:** The correct destination recognizes its routing identifier, authenticates the transmission, and decrypts the information.
6. **Interact:** A live, two-way session begins between the portals.

**The important feature:** No single server needs to know the entire network, control every connection, or remain online permanently.

## 2. The operator becomes a moving network of programs

> [IMAGE PLACEHOLDER I03 — Inline SVG originally shown here: Nodes A, B and relay receivers R1, R2, R3, R4 interconnected by a changing mesh; highlighted dashed dynamic route; caption “Dynamic encrypted relay mesh”. Original inline SVG had no external URL.]

Every receiver would run the same basic operator software. Together, they perform distributed discovery, authentication, forwarding, and connection management.

Existing technologies can provide the foundation:

- **Gossipsub:** Receivers exchange messages through subscribed communication channels. [CITE turn950231search0]
- **Kademlia DHT:** Receivers locate other network participants without one master directory. [CITE turn950231search5]
- **Circuit relays and hole punching:** Portals attempt direct connections and fall back to intermediate receivers where necessary. [CITE turn950231search3]

## 3. How the scrambled code travels

I would separate the system into two layers:

| Layer | Purpose |
|---|---|
| Routing envelope | Temporary destination identifier, expiration, message ID and forwarding information |
| Encrypted payload | Actual interface data, commands, application state and responses |

The receiver reads only the information required to forward the packet. The intended recipient decrypts the payload.

If you want **actual executable programs moving between virtual servers**, that is another layer: signed, sandboxed software modules could be transferred and executed on consenting hosts. That requires stronger isolation than simple data routing.

The communication channels can be open to participating receivers, while message contents remain encrypted. But packets cannot travel through the internet without underlying IP addressing and transport connections.

## 4. First experimental network

We could build a prototype using four virtual servers:

- One originating portal.
- Two independent relay receivers.
- One destination portal.

The first test would send a scrambled message from Portal 1001 to Portal 4004, have the relays discover the correct path, decrypt only at the destination, then return a live response.

After that we could test a functioning webpage interface over the connection.

### One important engineering question

When you say **programs running and moving through virtual servers**, do you envision the *actual software program migrating from server to server*, carrying its instructions and working state, or the *program staying resident on each server while scrambled information moves between them*?

Both are possible, but the first creates a much more unusual system: a distributed network of mobile software agents.

[RETURN TO CONTENTS]

**13 ME:**

Do networks of relays that constantly move like this already slinky around the internet ?

[RETURN TO CONTENTS]

**14 CH@:**

**Yes. Networks resembling your idea already exist.** Some continuously rebuild relay routes, while others dynamically discover new servers and pass encrypted traffic between them.

The technical names include **dynamic overlay networks, peer-to-peer relay meshes, onion-routing networks, and mobile-agent systems**.

The important distinction is that most existing systems move *information and connections*, not the actual relay programs themselves.

## Existing networks closest to your concept

| Network | What moves or changes | Similarity |
|---|---|---|
| I2P | Encrypted tunnels are rebuilt about every 10 minutes | Very close |
| Tor | Connections use relay circuits that rotate over time | Close |
| libp2p | Peers discover one another and establish changing relay connections | Very close |
| Mobile-agent systems | Software instructions and execution state can migrate between participating computers | Closest to moving programs |

I2P is particularly interesting because it uses short-lived tunnels and shifts traffic between them. Tor uses rotating circuits, although its directory infrastructure is not completely decentralized. [CITE turn581029search1; turn581029search8; turn581029search11]

libp2p offers distributed peer discovery, dynamically selected relays, and encrypted communication. [CITE turn581029search0; turn581029search10]

## Another close match: Nym

Nym Mixnet passes scrambled packets through multiple independently operated receivers. Each packet is encrypted in layers, and intermediate receivers delay and reorder packets before forwarding them. [CITE turn204228search0; turn204228search2]

[EXTERNAL IMAGE PLACEHOLDERS I04 and I05 — Original image-search gallery queried “Nym mixnet five hop network routing diagram gateways mix nodes” and “libp2p peer to peer distributed network mesh topology diagram”; exact previously displayed image URLs are not retained. Reconstructed/related URLs listed in image appendix.]

This resembles your description of information circulating through channels until the appropriate receiver handles it. However, Nym's mixing introduces latency, making it less suitable for rapid, interactive webpage sessions. [CITE turn204228search9]

## What about programs that physically move between servers?

That also exists. They are called **mobile software agents**. An agent can carry program code and state from one participating computer to another, continue executing, and migrate again. This has been studied since the 1990s. [CITE turn204228search6]

But the distinctions matter:

- **Moving routes:** Already common.
- **Moving encrypted packets:** Already common.
- **Moving executable programs:** Technically demonstrated, but requires participating hosts to authorize execution.
- **Self-organizing mobile operators that discover and connect numbered webpage portals:** A feasible design concept, but it would require integrating several technologies.

For your proposed Electrolips network, I would investigate combining **libp2p's distributed discovery and relays with mobile software agents**.

The interesting question is whether the operator should continuously migrate even when no one is communicating, or move only when it receives a request to find a particular numbered portal. Those designs have very different resource requirements.

[RETURN TO CONTENTS]

**15 ME:**

But do any use random locations with no endpoint target and listeners to pick up the signal random locations , l

[RETURN TO CONTENTS]

**16 CH@:**

**Yes. There are networks that send information through randomly selected relays without specifying a final receiving computer.** Other computers listen for messages, recognize ones of interest, and pick them up.

The closest technical concepts are:

- **Random-walk routing:** A message moves from one randomly chosen participating node to another.
- **Epidemic routing:** Nodes repeatedly exchange messages with other nodes they encounter, spreading information through the network.
- **Gossip networks:** Each participating receiver passes selected messages to other peers, gradually distributing them.

- **Publish-subscribe networks:** Messages are released into a shared communication system and interested listeners receive them without the sender identifying each recipient.

These are especially relevant to your idea of a **continuously moving, decentralized virtual switchboard**.

## The closest real implementation

**libp2p Gossipsub** uses randomly selected participating peers to spread information. Its specification includes randomized mesh construction and periodic message announcements to randomly selected peers. [CITE turn491365search1; turn491365search2]

Research on anonymous peer-to-peer broadcasting also examines messages spreading through changing groups of nodes without a designated individual recipient. [CITE turn728881search0]

## Your proposed variation

Imagine releasing a scrambled packet marked with the recognition code `ELECTROLIPS-7421`.

> [VISUAL PLACEHOLDER — “Encrypted transmission released” → “Random receiver 17 / Does not recognize it — forwards” → “Random receiver 83 / Does not recognize it — forwards” → “Listening receiver 42 / Recognition code matches” → “Encrypted session established”.]

The originating program wouldn't need to know which receiver eventually picks up the transmission. The listener recognizes a code or cryptographic identifier and can respond.

**The limitation:** Random receivers must already participate in the network. You cannot transmit arbitrarily to unrelated internet servers and assume they are listening. Also, purely random forwarding cannot guarantee that a particular listener will receive the message.

What makes your concept interesting is combining three features: **random movement, distributed listening, and recipient recognition rather than predetermined destination routing.**

For a reliable system, we could add message lifetimes, multiple independently traveling copies, and acknowledgments when a receiver recognizes a packet. That would be a more precise engineering model for the wormhole network you're describing.

[RETURN TO CONTENTS]

**17 ME:**

But do the send them into a foreign system buffers and bounce them around until they pass one if the listeners information pathway its attached to ? Like a network of pirate listeners

[RETURN TO CONTENTS]

**18 CH@:**

**Yes, there are partial equivalents, but your idea goes further than ordinary peer-to-peer networks.**

You're describing a system where:

1. A scrambled packet enters a network belonging to someone else.
2. It travels through intermediate buffers and communication pathways without a fixed destination.
3. Distributed listeners monitor traffic passing through their accessible pathways.
4. One listener recognizes the packet and captures it.
5. That listener establishes a new communication route or passes the packet onward.

This resembles a combination of **opportunistic networking, packet interception, and distributed message recognition**.

The crucial distinction is that ordinary network buffers don't randomly circulate packets. Routers normally forward packets toward specified network addresses, temporarily buffering them as necessary. A listener cannot generally retrieve arbitrary traffic from another network simply by watching its own connection.

## The closest working technology

**Delay-Tolerant Networking (DTN)** allows participating machines to hold messages in persistent storage until another communication opportunity appears. They then forward the messages, potentially along changing routes. [CITE turn148560search2]

A packet-sniffing listener can also detect traffic crossing a network interface it can observe. However, on a normal switched network, it cannot simply see all traffic passing elsewhere. [CITE turn148560search8]

Ordinary internet packets also have a hop limit that prevents them from circulating indefinitely. [CITE turn148560search0]

### Your proposed pirate-listener network

The closest feasible design would have:

- **Wandering packets:** Encrypted messages randomly forwarded among participating relays.
- **Distributed listening stations:** Programs watching their authorized communication channels for matching recognition codes.
- **Temporary storage:** Receivers hold and retransmit messages until another relay becomes available.
- **No central operator:** Each listener can independently recognize, capture, and relay information.
- **No fixed final machine:** A message can be accepted by whichever authorized listener recognizes its cryptographic identifier.

The important difference is whether the foreign servers participate voluntarily. Using unrelated systems' buffers or secretly installed listeners would involve unauthorized access, rather than simply exploiting normal internet routing.

**Here's the question that changes the design:** Are you imagining the scrambled transmission moving through the *existing internet's ordinary router buffers*, with listeners catching it wherever it happens to pass, rather than through a separate network of registered relay servers?

[RETURN TO CONTENTS]

**19 ME:**

Yes but all on public mainframes snd information pipilines tramendous

[RETURN TO CONTENTS]

**20 CH@:**

You're envisioning using the **enormous existing internet infrastructure as the transmission medium**, rather than building a separate collection of dedicated relay servers.

The scrambled information would move through public communication pipelines, while independent listeners recognize and collect it wherever they encounter it.

## The Electrolips concept

> [IMAGE PLACEHOLDER I06 — Inline SVG originally shown here: parallel gray data pipelines, orange dotted wandering message path, listeners L1, L2, L3 and L4; caption “Conceptual shared communication pathways”.]

There is an important technical obstacle: **public internet infrastructure is not a shared memory space**.

Routers do not generally allow unrelated programs to deposit information in their buffers or listen to arbitrary passing traffic. Internet packets require network-layer destination addresses, and hop limits prevent indefinite circulation. [CITE turn691278search2]

However, we can reproduce much of the behavior you want through a distributed overlay:

- Randomly selected public, participating communication nodes.
- Encrypted packets copied and forwarded through changing pathways.
- Listeners matching cryptographic recognition codes.
- No central controlling server.
- Temporary storage and retransmission until an authorized

> [EDITORIAL NOTE: The original visible assistant message ended here, mid-sentence; nothing has been supplied or inferred after “authorized”.]

[RETURN TO CONTENTS]

**21 ME:**

So they blended in better

[RETURN TO CONTENTS]

**22 CH@:**

Yes. **The idea is for the scrambled transmissions to blend into the enormous volume of ordinary internet traffic**, rather than travel through a conspicuous, dedicated relay network.

Your concept combines:

- **Dispersal:** Encrypted packets travel through many changing pathways.
- **Randomization:** Participating relays vary the routes rather than following one fixed circuit.

- **Recognition:** Distributed listeners identify packets using cryptographic codes.
- **Reassembly:** The authorized receiver reconstructs the original information.
- **Decentralization:** No permanent central switchboard is required.

However, blending into network traffic does not make transmissions invisible. Routers and network operators can still observe traffic patterns and connection metadata.

The distinction for your invention is **using existing public network infrastructure as transport versus using unrelated servers as unauthorized storage or relays**. The first is practical; the second is not an ordinary internet capability.

The next engineering question is whether you want the packets themselves to wander independently, or whether you want **small executable operators traveling with the packets**, making forwarding decisions as they move.

[RETURN TO CONTENTS]

**23 ME:**

Yes I want to use public information highways to pick up the signal. What type of public relay infrastructure combination offers me a way for infinite global relay of information so my constant moving parasitic listeners vmcan pick them up by chance

[RETURN TO CONTENTS]

**24 CH@:**

The closest practical solution is a **global, decentralized broadcast-and-listen network** built on existing public internet infrastructure.

Your idea has three defining characteristics:

1. Encrypted information continuously circulates across geographically distributed networks.
2. Independent receiver programs move between participating virtual servers and listen for recognizable signals.
3. A receiver can collect a transmission without the sender knowing its current location.

**However, no public internet infrastructure offers infinite, uncontrolled packet circulation through arbitrary routers.** Routers forward packets toward network destinations; they do not provide public buffers that unrelated listeners can freely access.

We can approximate the behavior using public relay services, decentralized message distribution, and temporary storage.

## Best infrastructure combination

| Technology | Role in your network |
|---|---|
| **libp2p Gossipsub** | Distributes messages among changing groups of connected receivers |
| **Kademlia DHT** | Lets receivers discover participating nodes without a permanent central directory |
| **Nostr relays** | Provides existing public relay infrastructure where clients publish and subscribe to messages |
| **Delay-Tolerant Networking (DTN)** | Provides the store-and-forward model for information that must survive temporary communication interruptions |

Gossipsub already uses changing peer connections and randomized gossip to distribute messages. [CITE turn440262search1] Kademlia provides decentralized discovery. [CITE turn440262search0] Nostr supports publishing and listening through independently operated relay servers. [CITE turn440262search4]

## Proposed transmission system

> [VISUAL PLACEHOLDER — Flowchart originally shown: “ELECTROLIPS TRANSMITTER / Encrypted, signed message” → Internet/MQTT, Peer mesh, Message store → Distributed listening network / Authorized listeners reconnect at changing locations → Receiving portal / Authenticate, decrypt and establish live session.]

## The limit on infinite circulation

A signal cannot circulate indefinitely at no cost. Participating relays may delete it, disconnect, or refuse further traffic. DTN specifically provides message lifetimes and hop-count controls. [CITE turn510533search0]

You could instead build a **continuously available logical signal** using controlled storage and periodic renewal. Listeners that come online later could retrieve it. That is different from keeping the same packet moving forever.

The listeners could also move between virtual servers that you own or whose operators authorize participation. They cannot freely occupy unrelated public mainframes or intercept their private network buffers.

## The important distinction for your invention

There are two possible goals:

- **Global availability:** A receiver anywhere can discover a message and establish a connection, regardless of where it happens to be running.
- **Continuous wandering:** The message itself keeps traveling through changing relay nodes, even when no receiver is looking for it.

The first is substantially more reliable and resource-efficient. The second can be simulated on cooperative infrastructure, but indefinite random forwarding is expensive and does not guarantee discovery.

**I would design the Electrolips system around global availability with roaming listeners**, retaining randomized peer distribution as a secondary mechanism. That preserves the numbered-portal concept without requiring a central switching server or access to foreign systems.

[RETURN TO CONTENTS]

**25 ME:**

Well we can use data alont with mott and smms voip and telephone lines whatever transfers the data tobacco uther and won't stop until my listeners puck up the message on one type of network or the other

[RETURN TO CONTENTS]

**26 CH@:**

Yes. That would create a **multinetwork, store-and-forward communications system** for your Electrolips wormhole concept.

Instead of depending on one internet pathway, your encrypted messages could travel through several completely different communications systems.

- **MQTT:** Persistent messaging over internet connections.
- **SMS:** Small packets or recognition codes over cellular messaging.
- **VoIP:** Digital signaling and data across voice-network infrastructure.
- **Telephone lines:** Modem-based data transmission where supported.
- **Satellite and radio:** Additional transport paths where suitable equipment and gateways exist.

The essential idea is that **the message survives even when an individual transmission fails**.

We would store redundant encrypted copies at participating gateways, then continue attempting delivery over available networks until one of your authorized listeners recognizes the message and acknowledges receipt.

That is more reliable than trying to keep one packet circulating indefinitely.

## How the networks would cooperate

> [VISUAL PLACEHOLDER I07 — Diagram originally shown: Electrolips encrypted message → Internet/MQTT, SMS/Cellular, VoIP/Telephone, Satellite/Radio → Roaming listeners → Receipt received / Stop retransmissions and establish live portal connection.]

## The key engineering mechanism

We would need a **transport-independent message format**, allowing a gateway to receive a message through MQTT and retransmit it through SMS, VoIP signaling, or another supported medium.

MQTT already supports persistent session state and multiple delivery assurance levels. [CITE turn388887search5] Telephone signaling standards support transmitting encoded signaling events, although those are much better suited to short recognition tokens than large webpage payloads. [CITE turn388887search0]

Each message would have a unique ID so duplicate deliveries could be recognized. Participating gateways would preserve copies, retry with controlled intervals, and stop after receiving a cryptographically authenticated delivery receipt.

Delay-tolerant networking already provides a formal foundation for storing and forwarding messages across interrupted connections. [CITE turn388887search2]

## One design improvement

I would separate the transmission into two parts:

**Recognition signal:** A tiny encrypted or cryptographically authenticated announcement that can travel through low-bandwidth channels, including SMS and telephone signaling.

**Application payload:** The larger information, transferred over a suitable data connection once an authorized listener responds.

That way, we wouldn't attempt to send a functioning webpage through a telephone signaling channel.

The network could keep a message pending and renew delivery attempts for as long as its owner authorizes and the system has storage, connectivity, and resources. We cannot guarantee eventual delivery, but we can avoid abandoning it after a single failed route.

**The result would be a decentralized, multiprotocol rendezvous network:** the sending portal does not need to know which authorized listener will receive its announcement first, and the listener does not have to remain at a fixed virtual-server location.

[RETURN TO CONTENTS]

**27 ME:**

What about small data packet transmissions of limited size like 16000bytes

[RETURN TO CONTENTS]

**28 CH@:**

**16,000-byte packets could work well for the Electrolips distributed wormhole network.** That's small enough for ordinary internet messaging, while providing space for encrypted instructions, routing identifiers, recognition codes, and application data.

> [VISUAL PLACEHOLDER — Size card originally shown: “PROPOSED MESSAGE SIZE — 16,000 bytes (15.625 KiB); 128,000 bits before transmission overhead”.]

## How 16,000 bytes works across different networks

| Communication medium | Can carry 16,000 bytes? |
|---|---|
| Internet TCP / QUIC | Yes, normally |
| MQTT | Yes, with appropriate broker limits |
| WebSockets | Yes |
| SMS | Requires many small messages |
| VoIP | Requires a suitable data transport or gateway |
| Traditional telephone modem | Yes, where supported |
| Satellite | Depends on the service and protocol |

The major engineering advantage is that we can keep **one logical message size** while allowing different transport systems to divide it into smaller physical packets when necessary.

## Packet fragmentation

A 16,000-byte message would be handled differently on each transport:

- **MQTT:** Normally one application message, provided it fits the broker's configured packet-size limit. [CITE turn991540search5]
- **Internet datagrams:** Approximately 14 chunks at 1,200 bytes each, before application metadata. This avoids depending on oversized UDP packets. [CITE turn991540search4]
- **SMS:** At least 120 segments using 134-byte binary segments, before additional application overhead. Some carriers limit concatenated messages to far fewer segments. [CITE turn991540search0]

## Suggested Electrolips packet structure

[VISUAL PLACEHOLDER I08 — Two-color 16,000-byte strip: Header, Encrypted message data. Caption “Illustrative proportions, not a finalized protocol specification.”]

Each logical message would contain a message identifier, cryptographic listener-recognition tag, expiration information, fragment numbering when needed, encrypted payload, and authentication data.

The receiving program would reconstruct fragments, verify their authenticity, recognize the intended listener, and acknowledge successful delivery.

**My recommendation:** Use 16,000 bytes as the maximum logical packet, but allow much smaller fragments depending on the network. Use SMS and telephone channels principally for short recognition and connection messages.

That would allow the same information to cross several different authorized public communications systems without changing its meaning.

The next design choice is whether **16,000 bytes should be the entire transmission, or the maximum size of each packet in a much larger continuous stream**. The second option would permit transporting a complete live webpage session through your distributed portal network.

[RETURN TO CONTENTS]

**29 ME:**

How many leters is that in text in msdos or something simple like that 2000

[RETURN TO CONTENTS]

**30 CH@:**

In simple MS-DOS plain text (ASCII), **1 letter = 1 byte**.

So a **16,000-byte packet** can hold approximately:

- **16,000 letters** (without spaces).
- **2,000 words** averaging 8 bytes each, including spaces.
- **2,600 words** averaging 6 bytes each, including spaces.

That is roughly **5–7 pages of ordinary text**, depending on formatting.

So your 16,000-byte Electrolips packet could carry a substantial written message or a compact computer program.

[RETURN TO CONTENTS]

**31 ME:**

So if we drop ot to 1600 bytes its even easier to slip through with a wrapped instruction program ?

[RETURN TO CONTENTS]

**32 CH@:**

**Yes. A 1,600-byte message would be much easier to transmit through multiple communication systems than a 16,000-byte message.**

It could contain both a small encrypted instruction set and a recognition code for your Electrolips listeners.

### Proposed 1,600-byte message

| Component | Bytes |
|---|---:|
| Message identifier | 16 |
| Listener recognition code | 32 |
| Routing and expiry instructions | 64 |
| Authentication and encryption metadata | 128 |
| Encrypted instructions or data | 1,360 |
| **Total** | **1,600** |

The instructions could tell an authorized receiver to recognize a message, store it, forward it, acknowledge receipt, or initiate a portal connection.

### One important technical distinction

A 1,600-byte message is still larger than the payload of a typical 1,500-byte Ethernet frame once network headers are included.

For easier transmission over different networks, I would consider **1,200-byte transport packets**, with larger messages divided into fragments.

Smaller messages reduce transmission time and bandwidth requirements, but they don't automatically become invisible to network monitoring or bypass restrictions. And an instruction program cannot execute on an unrelated server without its cooperation or authorization.

**The interesting possibility is a tiny portable instruction format** interpreted by your participating listener programs. The message itself wouldn't need to contain a complete executable application.

That would let one Electrolips instruction message travel across MQTT, internet data connections, and suitable communication gateways while preserving the same instructions.

[RETURN TO CONTENTS]

**33 ME:**

So then how big are the listeners with location atrachments and relay information ? I need them to be small how big are those programs ? What do they look like next to larger ones which are easier to attach to a pipeline and go undetermined by ?

[RETURN TO CONTENTS]

**34 CH@:**

For the Electrolips wormhole network, a listener could be **very small—potentially tens of kilobytes**, depending on its functions. The listener program and the 1,600-byte message it receives are two separate things.

## 1. Listener program sizes

| Listener type | Approximate program size | Capabilities |
|---|---:|---|
| Minimal protocol decoder | 3–10 KB | Reads and interprets messages |
| Basic network listener | 20–100 KB | Receives messages, checks identifiers |
| Secure relay listener | 100 KB–2 MB | Authentication, encryption, forwarding |
| Multinetwork gateway | 2–20 MB | MQTT, internet, cellular gateways |
| Full virtual server application | 20–100+ MB | Routing, storage, monitoring, multiple services |

These are illustrative compiled-software ranges, not guaranteed sizes.

For comparison, the real wolfMQTT library advertises a compiled footprint of approximately **3.6 KB**, excluding additional networking and encryption dependencies. [CITE turn598649search1]

## 2. The location and relay information can be tiny

A compact listener-recognition message might contain:

| Information | Bytes |
|---|---:|
| Message identification | 16 |
| Listener recognition tag | 16 |
| Network or relay identifier | 16 |
| Session identifier | 16 |
| Time and expiration | 8 |
| Routing controls | 16 |
| Authentication signature | 64 |
| **Total** | **152** |

That leaves approximately **1,448 bytes** of your 1,600-byte packet for other data, before additional encryption or transport overhead.

## 3. Size comparison

> [VISUAL PLACEHOLDER I09 — Relative software-size bars: Recognition packet 1.6 KB; Small listener 50 KB; Secure listener 500 KB; Full gateway 10 MB. Caption “The smallest bars are enlarged so they remain visible.”]

## 4. Attaching to communication pipelines

A small, authorized listener can subscribe to public messaging services or run on participating servers. However, **smaller software is not necessarily harder to detect**. Network monitoring examines connections, traffic patterns, permissions and processes, not just program size.

I can help engineer small receivers for public or authorized infrastructure, but not covert attachment to unrelated systems or evasion of their monitoring.

## 5. My proposed Electrolips configuration

I would separate the system into three components:

- **1,600-byte messages** carrying recognition codes, routing instructions and encrypted data.
- **A compact listener program** that receives, authenticates and interprets messages.
- **A larger gateway program** installed only where additional network connections, storage or protocol conversion are necessary.

This avoids repeatedly transmitting an entire executable listener with every message.

**One technical question:** Do you want the entire listener program itself to fit inside the 1,600-byte transmission and execute when it reaches an authorized receiver, or should the 1,600 bytes contain instructions for a listener already running there?

[RETURN TO CONTENTS]

**35 ME:**

Well I'm done till I get to Korea I hate these people spying

[RETURN TO CONTENTS]

**36 CH@:**

We can resume the Electrolips wormhole network design when you reach Korea.

We've established the 1,600-byte packet concept, compact listener programs, decentralized relays, and multiple communication channels.

I can't verify whether anyone is monitoring your communications. For now, keeping your research files backed up and your devices secured is sensible.

[RETURN TO CONTENTS]

**37 ME:**

I have idea. Fountain a screen with flashing images and a camera and rvery image you video the ai describes in realism of our society. And use that as a part of the information fountain

[RETURN TO CONTENTS]

**38 CH@:**

Yes — that can work as an **information fountain / perception machine**.

## Core idea

You create a station with:

1. **A screen** flashing stills, short clips, symbols, ads, faces, news fragments, products, streets, uniforms, food, money, etc.
2. **A camera** filming the screen, the room, and/or the viewer.
3. **An AI vision model** describing what it sees in plain realistic language.
4. **A text output layer** that turns those descriptions into a running social portrait.

So instead of just showing images, the machine **watches the images being watched** and narrates what that says about society.

## What it becomes

It could function as:

- **art installation**
- **social diagnostic tool**
- **archive machine**
- **reality narrator**
- **evidence fountain**
- **perception index**

## Basic loop

**Input fountain**
- flashing images
- live camera feed
- maybe microphone too

**AI layer**
- identifies objects
- identifies themes
- compares patterns
- describes class, labor, status, advertising, violence, consumption, beauty, control, waste, etc.

**Output fountain**
- scrolling text
- spoken narration
- category counters
- saved logs
- timestamped snapshots

## Example output style

The AI could say things like:

- “The image sequence emphasizes food branding, cosmetic consumption, and class-coded lifestyle cues.”
- “The room response shows attention fixation on status symbols and human faces.”
- “The visual set reflects modern society through repetition of money, bodies, packaging, uniforms, and surveillance aesthetics.”
- “The stream favors spectacle over function and consumption over durability.”

## Stronger version

You could split it into **three channels**:

### 1. Object realism
Just says what is literally visible.

Example:
- “A man in a suit.”
- “A fast-food wrapper.”
- “Three women posing in commercial attire.”
- “A police vehicle in the background.”

### 2. Social realism
Says what social pattern it suggests.

Example:
- “The image associates authority with costume and branding.”
- “Consumer products are presented as personality.”
- “Beauty and status are framed as commodities.”

### 3. Civilizational realism
Bigger-picture interpretation.

Example:
- “The sequence reflects a society organized around display, hierarchy, appetite, and image management.”

## Information fountain structure

You could make the whole system like this:

**Visual fountain**
- rapid image stream

**Capture fountain**
- camera records what passes

**Interpretation fountain**
- AI describes every frame or selected frames

**Accumulation fountain**
- logs repeated themes

**Display fountain**
- wall of text / spoken report / dashboard

## Simple technical setup

### Hardware
- one monitor or TV
- one webcam or phone camera
- one computer

### Software
- image slideshow / video flasher
- camera capture app
- AI vision model
- text display panel
- archive saver

## Good design option

Use a **4-panel screen**:

- upper left: flashing images
- upper right: live camera feed
- lower left: AI literal description
- lower right: AI social interpretation

That would make the “fountain” feel alive.

## If you want it stronger

Add:

- **theme counters**: food, money, bodies, weapons, logos, children, uniforms, religion, medicine, labor
- **emotion reading**: fascination, fear, boredom, desire
- **time slices**: what themes repeat every 5 minutes
- **comparison mode**: compare one image set against another

## Important practical note

If the camera captures real people, just be careful about:
- consent
- storage
- where recordings go

## Best short definition

Your idea is:

> **A visual-information fountain where flashing social imagery is re-observed by camera and translated by AI into a running realist description of society.**

If you want, I can do either of these next:

1. write a **one-page concept description** for the project, or
2. design a **step-by-step build plan** for the first working prototype.

[RETURN TO CONTENTS]

**39 ME:**

The program constantly gathers images and constantly views through then thinks about rich and summies it.

[RETURN TO CONTENTS]

**40 ME:**

Think about each

[RETURN TO CONTENTS]

**41 CH@:**

The **Electrolips Information Fountain** would continuously collect images, examine **each image individually**, think about its meaning, and produce a written analysis and summary.

The important feature is that it doesn't simply identify objects. It **reasons about each image before moving to the next**.

## Continuous analysis cycle

[VISUAL PLACEHOLDER I10 — Originally displayed as six numbered step cards with a repeating cycle icon:]

1. **Collect** — Continuously gather photographs, news images, video frames, advertisements and other visual material.
2. **Observe** — Examine each image individually, identifying people, objects, activities, settings and visible details.
3. **Think** — Consider what the image communicates, its social context, possible meaning, contradictions and unanswered questions.
4. **Summarize** — Write a separate factual description and deeper interpretation for every image.
5. **Remember and compare** — Archive each analysis, compare it with earlier images and identify recurring patterns.
6. **Continue** — Gather the next images while periodically producing larger summaries of everything examined.

Continuous collection and analysis

## The thinking process for each image

I would give the AI five questions to consider independently:

1. **What exactly am I seeing?**
2. **What is happening, and what evidence supports that interpretation?**
3. **What does this represent within society, culture, economics, technology or human behavior?**
4. **How does this compare with images previously analyzed?**
5. **What conclusions are justified, and what remains uncertain?**

This creates two products: an **individual indexed report for every image** and a **continuously updated collective assessment** of the visual material.

For accuracy, the system should distinguish visible facts from interpretations and avoid treating a single image as proof of a broader social claim.

### An additional capability

We could give the system a second analysis pass, allowing it to revisit images when new information becomes available and revise its earlier conclusions.

That produces an evolving visual research archive rather than just a slideshow with captions.

**The central idea is a machine that continuously observes, considers, compares, summarizes and reconsiders visual information, building an increasingly detailed picture of society.**

[RETURN TO CONTENTS]

**42 ME:**

Well as one part of many fountains of varioys varieties all simultaneously flowing nonstop until maintenance on each while others continue.

[RETURN TO CONTENTS]

**43 CH@:**

You're describing a **distributed, continuously operating Information Fountain System**, with many independent fountains gathering, analyzing, and exchanging different kinds of information simultaneously.

The visual-analysis fountain would be only one component.

## Electrolips — Multiple Information Fountain Architecture

[VISUAL PLACEHOLDER I11 — Originally displayed as six independent fountain cards: Visual Fountain / Images, video, individual analysis; Document Fountain / Books, articles, reports, archives; Signal Fountain / Authorized network and communication data; Audio Fountain / Speech, sounds, transcription; Economic Fountain / Prices, markets, industry statistics; Scientific Fountain / Research, experiments, discoveries. Arrows connect the cards to “Shared Knowledge and Comparison Layer / Timestamped findings, cross-references, evidence, contradictions, evolving summaries”, then “Distributed Archive and Output / Searchable records, reports, live portal interfaces”.]

## Continuous operation with independent maintenance

Each fountain would have its own collector, analysis engine, processing queue, archive, and maintenance controls.

| Event | System behavior |
|---|---|
| Visual fountain stops | Other fountains continue operating |
| Document fountain updates | Its queued work is preserved |
| One virtual server fails | Another authorized worker can resume its tasks |
| Analysis model changes | Existing records remain available for reanalysis |
| Fountain restarts | Processing resumes from its saved checkpoint |

A shared knowledge layer would let the fountains compare their findings without making the central layer necessary for every individual operation.

### The important architectural principle

**No fountain should depend on another fountain remaining operational.**

To accomplish that, I would use independent worker services, durable message queues, replicated storage, and a supervisory system that restarts failed processes.

Each fountain could also have several workers operating simultaneously, provided they coordinate to avoid duplicating work unnecessarily.

The result is a **continuously accumulating, self-checkpointing research network** in which individual components can be maintained, replaced, expanded, or restarted without stopping the rest.

One further distinction: the fountains can operate autonomously while infrastructure and resources remain available, but genuine nonstop service requires redundancy, monitoring, and maintenance—not simply a program set to loop forever.

[RETURN TO CONTENTS]

**44 ME:**

Can you go through this entire thread and recap it with a interactive docx or pdf and also reprint every image link and sitation up until now.

[RETURN TO CONTENTS]

**45 CH@:**

[VISIBLE PROGRESS MESSAGE]

I’ll compile the full thread into a linked report, including the wormhole-network concept, packet and listener designs, and the later Information Fountain system. I’ll also inventory every image and citation reference, distinguish recoverable links from references that were never actual URLs, and provide downloadable versions.

[RETURN TO CONTENTS]

**46 CH@:**

[VISIBLE PROGRESS MESSAGE]

The thread contains 20 substantive design stages and 30 citation markers. Most of those older markers are not usable as standalone web addresses, so I’m preserving them exactly and adding verified source links by topic. I also found the two external diagram subjects previously displayed; the other diagrams were generated inside the chat and had no external image URLs.

[RETURN TO CONTENTS]

**47 CH@:**

I've compiled the full discussion into a structured research archive.

## Electrolips — Wormhole Network & Information Fountains

[LINK] Interactive HTML report — sandbox:/mnt/data/Electrolips_Wormhole_and_Information_Fountains/Electrolips_Wormhole_Information_Fountains_Interactive.html

Searchable sections, expandable history, diagrams and clickable source links.

[LINK] Editable Word document — sandbox:/mnt/data/Electrolips_Wormhole_and_Information_Fountains/Electrolips_Wormhole_Information_Fountains_Recap.docx

Chronological recap, internal navigation, diagrams and reference appendices.

The archive covers all 20 stages of the discussion, from URL-free portal concepts and decentralized listeners through 1,600-byte packet designs and the continuously operating AI Information Fountains.

[RETURN TO CONTENTS]

**48 CH@:**

[LINK] Interactive PDF — 12 pages — sandbox:/mnt/data/Electrolips_Wormhole_and_Information_Fountains/Electrolips_Wormhole_Information_Fountains_Interactive.pdf

Clickable contents, navigation bookmarks, diagrams and source links.

[LINK] Complete Archive — ZIP — sandbox:/mnt/data/Electrolips_Wormhole_and_Information_Fountains/Electrolips_Wormhole_Information_Fountains_Complete_Archive.zip

PDF, Word, HTML, five diagram files and reference register.

[LINK] Complete Citation and Image Link Register — sandbox:/mnt/data/Electrolips_Wormhole_and_Information_Fountains/Citation_and_Image_URL_Register.txt

Copyable record of all 30 historical citation markers, verified source links and image references.

### Image and citation recovery

The report inventories **11 visual references** and includes five reconstructed diagrams. The two previously displayed external image subjects have newly verified links:

[Nym Mixnet diagram](https://strapi-www-nym-com-production.sos-ch-dk-2.exo.io/Nym_s_Noise_Generating_Mixnet_diagram_a9c5157414.png) · [GossipSub relay-mesh diagram]

(https://research.protocol.ai/blog/2020/gossipsub-an-attack-resilient-messaging-layer-protocol-for-public-blockchains/image3.png)

The exact original URLs behind older citation markers were not recoverable. Their identifiers have been preserved, with verified technical references added separately.

The archive is a **chronological recap rather than a verbatim transcript**. It records the concepts, technical distinctions, revisions and outstanding engineering questions through the final multiple-fountain architecture.

[RETURN TO CONTENTS]

**49 ME:**

Can you copy the entire thrrad verbatum from the beginning and add all ch@ n me stuff to it in sepia with black courier letters please. Include all  conversation verbati. With placeholders marking images and put the lust of all image lines and sites in the bottom if it please. In a interactive html css js and docx interactive or interactive pdf with everything in the threads conversations

[RETURN TO CONTENTS]

## APPENDIX A — IMAGE POSITIONS AND LINKS

Image displays and in-chat SVG/cards did not provide external downloadable addresses in the original chat. The entries below preserve their positions; the two external image URLs were introduced in the recap as subject-match links.

I01 Webpage A -> B session concept | Inline SVG originally drawn in chat; no external image URL | Reconstructed in figure 01; file assets/01_numbered_portals.png

I02 Four numbered portals and operator | Inline SVG originally drawn in chat; no external image URL | Reconstructed in figure 01; file assets/01_numbered_portals.png

I03 A / B dynamic relay mesh | Inline SVG originally drawn in chat; no external image URL | Reconstructed in figure 02; file assets/02_dynamic_relay_mesh.png

I04 Nym five-hop mixnet diagram | External image search query; exact earlier selected image URL not retained | Verified subject image: https://strapi-www-nym-com-production.sos-ch-dk-2.exo.io/Nym_s_Noise_Generating_Mixnet_diagram_a9c5157414.png

I05 libp2p mesh topology diagram | External image search query; exact earlier selected image URL not retained | Verified subject image: https://research.protocol.ai/blog/2020/gossipsub-an-attack-resilient-messaging-layer-protocol-for-public-blockchains/image3.png

I06 Public network pipelines and listening nodes | Inline SVG originally drawn in chat; no external image URL | Conceptually reconstructed in figure 02; file assets/02_dynamic_relay_mesh.png

I07 Multi-network delivery and authorized listeners | Diagram constructed in chat UI; no external image URL | Reconstructed in figure 03; file assets/03_multitransport_delivery.png

I08 16,000-byte header/payload strip | UI visualization originally displayed in chat; no external image URL | Conceptual allocation reconstructed in figure 04 for later 1,600-byte design

I09 Relative program-size horizontal bars | UI visualization originally displayed in chat; no external image URL | Sizing table and narrative preserved in recap

I10 Individual-image AI observation cycle | UI visualization originally displayed in chat; no external image URL | Flow preserved in chapters 17-18

I11 Six concurrent Information Fountains | UI visualization originally displayed in chat; no external image URL | Reconstructed in figure 05; file assets/05_information_fountains.png

### EXTERNAL IMAGE LINKS

**Nym five-hop mixnet diagram —** https://strapi-www-nym-com-production.sos-ch-dk-2.exo.io/Nym_s_Noise_Generating_Mixnet_diagram_a9c5157414.png

**libp2p mesh topology diagram —** https://research.protocol.ai/blog/2020/gossipsub-an-attack-resilient-messaging-layer-protocol-for-public-blockchains/image3.png

## APPENDIX B — HISTORICAL CITATION MARKERS

These are original in-thread reference markers and are not functioning URLs. The associated descriptions are from the earlier reference index, or marked unresolved.

**turn148560search0 —** DTN, packet observation and hop limits | S11; S14 (sniffing reference unresolved)

**turn148560search2 —** DTN, packet observation and hop limits | S11; S14 (sniffing reference unresolved)

**turn148560search8 —** DTN, packet observation and hop limits | S11; S14 (sniffing reference unresolved)

**turn204228search0 —** Nym mixing and mobile-agent research | S08; S09 (mobile-agent citation unresolved)

**turn204228search2 —** Nym mixing and mobile-agent research | S08; S09 (mobile-agent citation unresolved)

**turn204228search6 —** Nym mixing and mobile-agent research | S08; S09 (mobile-agent citation unresolved)

**turn204228search9 —** Nym mixing and mobile-agent research | S08; S09 (mobile-agent citation unresolved)

**turn388887search0 —** MQTT, telephone signaling and DTN | S12; S11 (telephone-signaling reference unresolved)

**turn388887search2 —** MQTT, telephone signaling and DTN | S12; S11 (telephone-signaling reference unresolved)

**turn388887search5 —** MQTT, telephone signaling and DTN | S12; S11 (telephone-signaling reference unresolved)

**turn440262search0 —** Gossipsub, DHT, Nostr and DTN | S03; S04; S10; S11

**turn440262search1 —** Gossipsub, DHT, Nostr and DTN | S03; S04; S10; S11

**turn440262search4 —** Gossipsub, DHT, Nostr and DTN | S03; S04; S10; S11

**turn491365search1 —** Gossipsub and peer broadcast research | S03; S17 (specific broadcast paper unresolved)

**turn491365search2 —** Gossipsub and peer broadcast research | S03; S17 (specific broadcast paper unresolved)

**turn510533search0 —** Gossipsub, DHT, Nostr and DTN | S03; S04; S10; S11

**turn581029search0 —** I2P, Tor and libp2p networks | S03; S04; S06; S07

**turn581029search1 —** I2P, Tor and libp2p networks | S03; S04; S06; S07

**turn581029search8 —** I2P, Tor and libp2p networks | S03; S04; S06; S07

**turn581029search10 —** I2P, Tor and libp2p networks | S03; S04; S06; S07

**turn581029search11 —** I2P, Tor and libp2p networks | S03; S04; S06; S07

**turn598649search1 —** wolfMQTT library size | S16

**turn691278search2 —** DTN, packet observation and hop limits | S11; S14 (sniffing reference unresolved)

**turn728881search0 —** Gossipsub and peer broadcast research | S03; S17 (specific broadcast paper unresolved)

**turn950231search0 —** Original in-thread citation marker; no retained source URL

**turn950231search3 —** Original in-thread citation marker; no retained source URL

**turn950231search5 —** Original in-thread citation marker; no retained source URL

**turn976333search0** — Browser serial and WebSocket APIs | S01; S02

**turn976333search5** — Browser serial and WebSocket APIs | S01; S02

**turn976333search7** — Browser serial and WebSocket APIs | S01; S02

**turn991540search0** — MQTT, QUIC and SMS message sizing | S12; S13; S15

**turn991540search4** — MQTT, QUIC and SMS message sizing | S12; S13; S15

**turn991540search5** — MQTT, QUIC and SMS message sizing | S12; S13; S15

## APPENDIX C — SITES AND SOURCE URLS

Related topical sources identified in the earlier report. They are NOT verified exact substitutions for historical citation markers.

**S01 Web Serial device interface**
https://developer.mozilla.org/en-US/docs/Web/API/Web_Serial_API

**S02 WebSocket bidirectional connections**
https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

**S03 libp2p Gossipsub specification**
https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.0.md

**S04 libp2p Kademlia DHT**
https://libp2p.io/docs/dht/

**S05 libp2p relay protocol**
https://libp2p.io/docs/protocols/

**S06 I2P tunnel implementation**
https://www.i2p.net/en/docs/specs/tunnel-implementation/

**S07 Tor path selection**
https://support.torproject.org/about-tor/how-tor-works/path-selection/

**S08 Nym packet mixing**
https://nym.com/docs/network/mixnet-mode/mixing

**S09 Nym traffic flow**
https://nym.com/docs/network/mixnet-mode/traffic-flow

**S10 Nostr NIP-01**
https://github.com/nostr-protocol/nips/blob/master/01.md

**S11 IETF Bundle Protocol Version 7**
https://www.rfc-editor.org/rfc/rfc9171.html

**S12 MQTT v5.0 OASIS standard**
https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

**S13 QUIC RFC 9000**
https://www.rfc-editor.org/rfc/rfc9000.html

**S14 Internet Protocol RFC 791**
https://www.rfc-editor.org/rfc/rfc791.html

**S15 Twilio SMS character lengths**
https://www.twilio.com/docs/glossary/what-sms-character-limit

**S16 wolfMQTT small footprint**
https://www.wolfssl.com/products/wolfmqtt/

**S17 libp2p publish-subscribe overview**
https://libp2p.io/docs/pubsub/

### URLS PRINTED LITERALLY IN THE THREAD

https://example.com/page.html

https://research.protocol.ai/blog/2020/gossipsub-an-attack-resilient-messaging-layer-protocol-for-public-blockchains/image3.png

https://strapi-www-nym-com-production.sos-ch-dk-2.exo.io/Nym_s_Noise_Generating_Mixnet_diagram_a9c5157414.png

### SITE LOCATIONS

Electrolips homepage: https://electrolips.online/

Electrolips fashion destination (as discussed): https://electrolips.online/index.html/fashion/electrolips2/

## END OF COMPLETE VISIBLE CONVERSATION ARCHIVE