Workgroup: Network Working Group
Internet-Draft: draft-hallambaker-everything
Published: 28 June 2023
Intended Status: Informational
Expires: 30 December 2023
Authors: P. M. Hallam-Baker
ThresholdSecrets.com

Mathematical Mesh 3.0 Part X: Everything

Abstract

https://mailarchive.ietf.org/arch/browse/mathmesh/Discussion of this draft should take place on the MathMesh mailing list (mathmesh@ietf.org), which is archived at .

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 December 2023.

Table of Contents

1. Introduction
  1.1. Content Format
  1.2. Everything Notification Message

1.3. Protocols and Services.
1.4. Mesh Account Addresses and Mesh Callsigns
2. Definitions
2.1. Related Specifications
2.2. Defined Terms
2.3. Requirements Language
2.4. Implementation Status
3. Use Scenarios
3.1. Security Model
3.1.1. Confidentiality and Integrity
3.1.2. Abuse Control
3.1.3. Traffic Analysis Resistance
3.2. Asynchronous Messaging and Mail
3.3. Asynchronous Collaboration
3.4. Asynchronous Social Media
3.4.1. Author Role
3.4.2. Reader Role
3.4.3. Publisher Service
3.4.4. Curation Service
3.4.5. Aggregation Service
3.5. Synchronous Messaging
3.6. Synchronous Voice and Video
3.7. Conferencing
4. ETML
4.1. Level 1
4.2. Level 2
4.3. Level 3
4.4. Inline SVG
5. Everything Notification Message
5.1. Shared Classes
5.1.1. Structure: Header
5.1.2. Structure: TextInput
5.1.3. Structure: Footer
5.1.4. Structure: Item
5.1.5. Structure: Feedback
5.1.6. Structure: Comment
5.1.7. Structure: Publication
6. Security Considerations
6.1. Confidentiality
6.1.1. Traffic Analysis
6.2. Integrity
6.2.1. Bullying
6.2.2. Sybil attack (brigading).
6.3. Service
6.3.1. Censorship
6.3.2. Data Loss
7. IANA Considerations
8. Acknowledgements
9. Appendix A: ETML Schema

10. Normative References
11. Informative References

1. Introduction

Everything is a document format, notification structure and protocols that provide a consistent and coherent infrastructure for human interaction through the Internet.

All existing modalities of human Interaction addressed by existing Internet protocols are supported, these include:

*Mail

*News and mailing lists

*Chat

*Voice and Video

*Conferencing

*Collaboration

Despite the wide scope of Everything's capabilities, the adoption of a consistent approach to all forms of messaging with end-to-end security built in allows a considerable reduction in client complexity.

Everything builds on existing approaches. Where possible, existing protocols and infrastructure are adopted without modification (e.g. use of WebRTC for voice and video interaction). In other cases, existing formats are adapted.

Everything is an Open Infrastructure in which there are no gatekeepers determining who is allowed to provide service, any user may use the service provider of their choice and change their choice of service provider without switching costs.

1.1. Content Format

Everything Text Markup Language (ETML) is an XML markup built on a subset of HTML that is purely focused on the expression of content. Presentation capabilities such as choice of fonts, size, color etc. are intentionally omitted.

Specifying a separate markup language for Everything messaging permits HTML features that are undesirable or present security risks to be omitted. ETML does not support presentation capabilities such

as choice of fonts, size, color etc. ETML documents **MUST NOT** include any form of scripting or active code capability.

ETML is divided into three tiers, each of which of represent the content capabilities commonly used in a set of existing interactions:

Paragraph <p> Basic text formatting of single paragraphs of text with emphasis (italic, bold, underline, etc.) with links to remote content and included images and video.

Typical uses include chat messages, comments and annotations on documents.

Post <post> Structured texts with titles, subtitles and section headings.

Typical uses include mail messages and blog posts.

Document <document> Richer structured texts with captioned images etc.

Intended for use in longer articles and papers.

Advanced notations such as markup for mathematics, chemistry, physics, chess, etc. are supported through use of an extension schema (e.g. MathML for mathematics) with alternative presentation in SVG being provided if the recipient does not have the ability to render the markup.

Restricting user content to a limited repertoire according to context simplifies presentation of interactions in a comprehensible format. While texts such as Daniel Bomberg's 1519 edition of the Talmud demonstrate the practicality of presenting a base text, commentaries and commentaries on the commentary on a single page, attempting to present a book length commentary as response to a single paragraph of a base text is clearly absurd.

Restricting user content also simplifies authoring and permits clients to act as gateways to legacy communications infrastructures (SMTP email, proprietary social media platforms).

1.2. Everything Notification Message

Everything Notification Message (ENM) is a message format used to provide notification of content being available, In the case of very short text content, the notification may include the content itself.

ENM messages consist of a DARE envelope containing a JSON body presenting the notification itself. This allows use of existing

protocols used to synchronize DARE Sequences to be used to synchronize exchange of notifications.

The ENM schema is based on the widely used RSS format to facilitate interoperability. The chief difference being that an RSS feed is an XML document which must be replaced every time an item is added while an ENM sequence is an append only log. Thus, while an RSS feed typically contains only the most recently updated items, an ENM sequence can contain the entire publication feed without performance penalty.

1.3. Protocols and Services.

Support for the full range of interactions used in messaging, social media, mail etc. requires a wide range of services. As shown in section [TBS], support for social media interactions alone involves six separate roles and four different types of service. But even though the number of services is large the use of ENS messages encapsulated in DARE Sequences allows most of the interactions between these services to be implemented by a synchronization of DARE sequences.

Support for the following services allows an Everything application to support the full range of functionality in both the authoring and receiving modes:

Notification Publications Clients advise publishers of the availability of new content using the Mesh Service Post Mechanism.

Content Publication Static content that is longer than the maximum payload of a Mesh Message is published using the HTTP POST method.

Sequence Synchronization Deliver notification of the arrival of new content and feedback.

Presence The Mesh Presence service is used to support synchronous messaging. Every device that is accepting inbound communication requests to a Mesh account maintains a constant connection to the corresponding presence service which acts as a broker and access control enforcement point for inbound communication requests.

Voice and Video Streaming of audio and video content is provided by WebRTC under symmetric keys exchanged through the Mesh Presence service.

The protocols supporting these services are either existing standards (e.g. WebRTC) or represent codification of existing

practices that have evolved over decades within the consistent approach provided by the Mesh Architecture.

1.4. Mesh Account Addresses and Mesh Callsigns

The internal identifier used to uniquely identify Mesh accounts is the UDF fingerprint of the root signature keys(s) of their Mesh profile. Since a string of 20 or more characters of Base-32 characters offers poor usability, provision is made to allow use of DNS account addresses and/or Mesh callsigns instead.

A DNS Account Address is of the form defined in [RFC822] (*account@domain*) where *domain* is the DNS address of the Mesh Service Provider servicing the account and *account* is a text label uniquely identifies a particular Mesh account serviced by a provider. Mesh Clients **MUST** support use of internationalized domain names [RFC5890] and UTF8 Unicode addresses.

A Mesh callsign is of the form @account as specified in [draft-hallambaker-mesh-callsign]. In normal circumstances, a callsign is a lifelong address that a once issued, a user can hold without payment of any renewal fee or other form of rent.

The separation of the internal identifier from the human readable form allows users to make use of multiple account addresses and callsigns for the same account. The primary purpose of the human readable form is to facilitate contact exchange rather than to support addressing.

For example, Alice initially registers her account with example.com which assigns her the address *alice@example.com*. She then registers the callsign @alice and @alice-lastname. Finally, Alice changes her service provider example.net which assigns her the account address alice68@example.net as the account 'alice' is already taken there.

If Bob exchanges Mesh contact details with Alice, it is the contact entry in his address book used to send the message rather than alice@example.com, alice68@example.net, @alice, etc. Thus, he can enter any of these forms to send a message to Alice because the contact entry he has for Alice is bound to her permanent account fingerprint and is automatically updated when Alice changes her service provider.

If Carol attempts to use the address alice@example.com after Alice has moved to example.net, the contact exchange will only succeed if example.com has not assigned Alice's account identifier to a new user and provides forwarding service to Alice's new account.

2. Definitions

This section presents the related specifications and standards on which Everything is built, the terms that are used as terms of art within the Mesh protocols and applications and the terms used as requirements language.

2.1. Related Specifications

Everything is built on the Mathematical Mesh platform and makes use of the data formats and service formats defined by the Mesh specification suite. The following specifications have particular relevance to Everything:

Mesh Architecture [draft-hallambaker-mesh-architecture] Provides an overview of the Mesh architecture.

Uniform Data Fingerprint [draft-hallambaker-mesh-udf] Describes the UDF format used to represent cryptographic nonces, keys and content digests in the Mesh and the use of Encrypted Authenticated Resource Locators (EARLs) and Strong Internet Names (SINs) that build on the UDF platform;

Data at Rest Encryption [draft-hallambaker-mesh-dare]. Describes the cryptographic message and append-only sequence formats used in Mesh applications and the Mesh Service protocol.

Reliable User Datagram [draft-hallambaker-mesh-rud]. Describes the Mesh presentation and transport layer.

Mesh Callsign [draft-hallambaker-mesh-callsign] Describes issue and use of Mesh callsigns.

Mesh Presence Service [draft-hallambaker-mesh-presence] Specifies the protocol used to establish synchronous interaction sessions.

JSON-BCD Encoding [draft-hallambaker-jsonbcd]. Describes extensions to the JSON serialization format to allow direct encoding of binary data (JSON-B), compressed encoding (JSON-C) and extended binary data encoding (JSON-D). Each of these encodings is a superset of the previous one so that JSON-B is a superset of JSON, JSON-C is a superset of JSON-B and JSON-D is a superset of JSON-C.

2.2. Defined Terms

This document makes use of the terms defined in [draft-hallambaker-mesh-architecture].

2.3. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [draft-hallambaker-mesh-developer].

3. Use Scenarios

As its name suggests, Everything is designed to support the widest possible range of human interactions. But unlike the World Wide Web in which Web content is understood to be content that can be accessed using the vast majority of current Web browsers, it is not expected that every Everything client will support every form of interaction supported by Everything. The communication modalities supported by an application necessarily depends on the affordances offered by the device it is running on. While a user is very likely to perform collaborative editing of a document on a desktop or laptop, they are much less to be doing so on their smartwatch.

3.1. Security Model

One of the chief advantages of deploying a new interaction infrastructure is the ability to apply a consistent security approach to every form of interaction.

3.1.1. Confidentiality and Integrity

All user generated content is authenticated. All user generated content is end-to-end encrypted unless explicitly designated as public by the author.

In cases where the set of authorized recipients is not known at the time content is generated or is subject to change, the content is encrypted under the key of a threshold encryption group and decryption of the content controlled by a Mesh key service.

3.1.2. Abuse Control

Every interaction is gated by access control.

Everything interactions are always brokered by one or more services. In the case of an asynchronous interaction, the broker is either the Mesh Service Provider of the recipient or a collaboration service host supporting a subset of the Mesh Service Protocol operations. In the case of a synchronous interaction, a Mesh Presence Service is

used to establish a session. In each case, the inbound connection broker is responsible to perform access control on the communication request. Thus knowing the address or callsign of an Everything user does not in itself grant the ability to wake them at night and attempt to sell them double glazing.

3.1.3. Traffic Analysis Resistance

Establishing a direct communication channel between the initiator and responder in a synchronous messaging or streaming interaction is most efficient but may compromise the privacy of the participants. IP addresses provide a powerful tracking tool that may expose the participants location or other identities.

This concern may be efficiently addressed by introducing proxy forwarders. Since the RUD transport protocol ignores the packet source address, separate proxy forwarders may be used on the inbound and outbound connections.

If a forwarder is trusted by both participants and third party traffic analysis is not being considered as a threat, a single stage of forwarding is sufficient. This approach is shown in Figure 1.

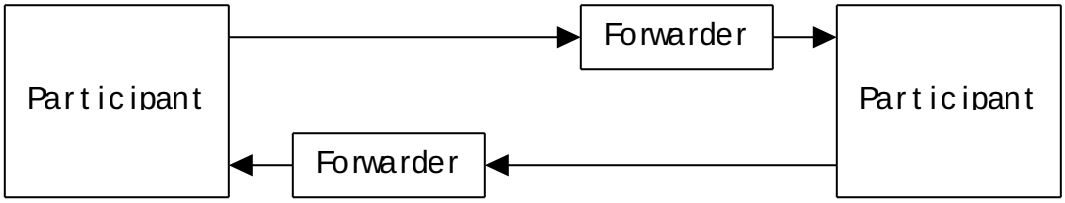

Figure 1: Concealing participant IP address.

If third party traffic analysis is a concern or the forwarders are not trusted, a multi-level onion routing network may be established with different packets taking different routes through a network whose usage patterns are concealed by traffic from other parties and the use of flood fill approaches. This approach is shown in Figure 2.

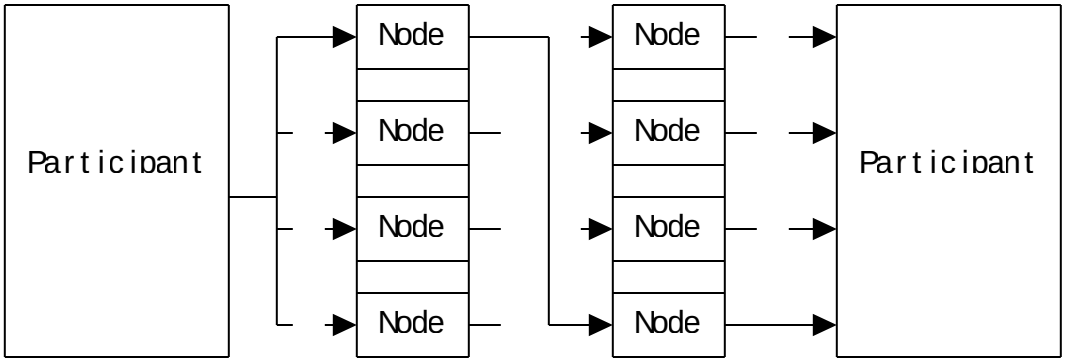

Figure 2: Traffic analysis resistance via onion routing.

3.2. Asynchronous Messaging and Mail

From a protocol point of view, asynchronous messaging services such as SMS only differ from email in the message sizes supported. Since SMTP is a push protocol in which the receiving mail server must accept and store the entire mail message on receipt, the message size is necessarily limited according to site policy. The maximum message size accepted is rarely more than a few MB and so larger data transfers are typically achieved by sending a link from which the receiver may retrieve the data from a clous service. Thus users are required to make use of three separate applications for very short messages, longer messages and very large data sets.

In the Everything Mail model (Figure 3), a sender passes messages to a receiver through a four-corner model. If the message is small enough to fit inside a notification, the sender first forwards the message to their own service provider who forwards it to the service provider acting for the receiver from which the receiver may collect the message to read on their various devices.

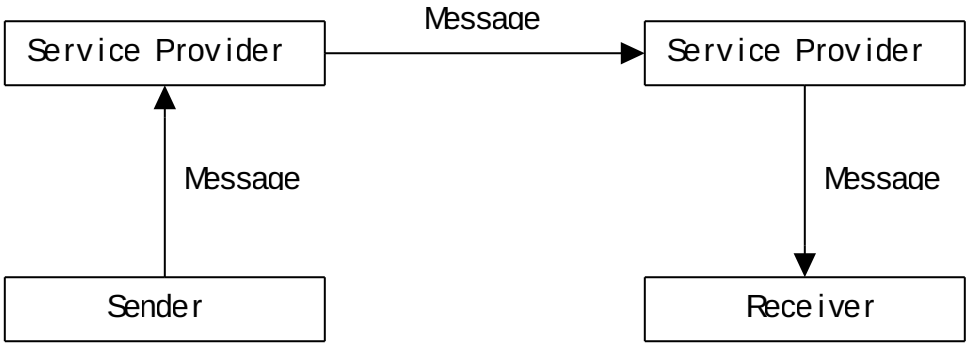

Figure 3: Four Corner Mail.

The outbound and inbound service providers perform access control on the notification request.

The maximum notification message size is intentionally limited to a small value, currently 32KB. This has performance and implementation advantages. There is little point in synchronizing a streaming video file of several GB in length to the user's watch. Limiting notification message size allows an approach in which every Everything device downloads the complete notification spool.

Transfer of longer messages is supported through a pull interaction in which the notification message contains only the message title and either an EARL providing the location of the encrypted resource and a decryption key in the case of a private message or a link to the resource in the case of a public message (Figure 4).

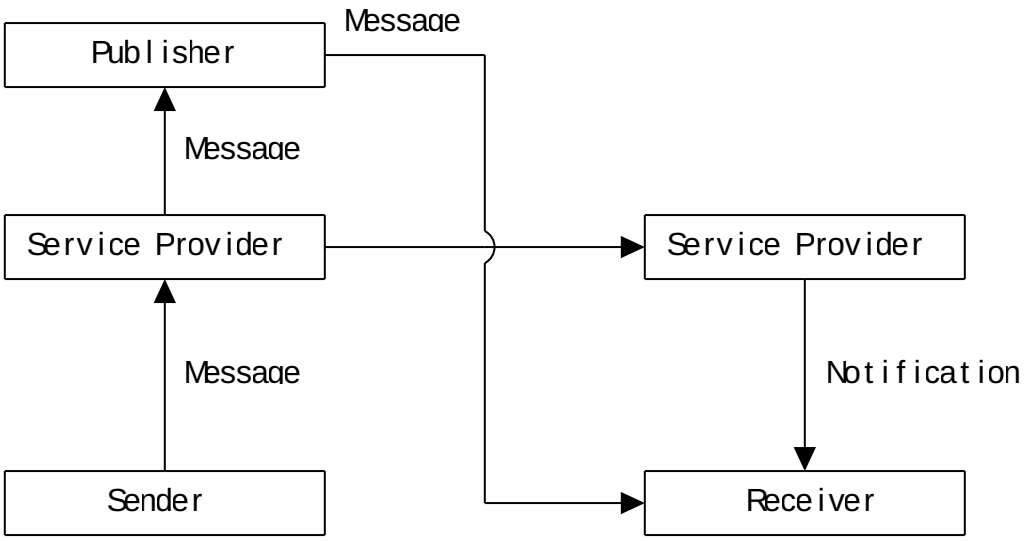

Figure 4: Mail With Large Payload.

The notification-pull interaction removes the need to limit the size of messages transferred in the protocol, the only limit being the storage capacity available at the sender and receiver.

This interaction is not currently fully specified. The following capabilities should be considered:

*Specify the size and content type of the payload message

*Provision for integration of anti-malware scanning services on content

*Linking to 'live' documents that remain at the specified link

*Caching of message content

*Message recall

Note that since a client **MAY** retrieve the message payload automatically without user interaction, message retrieval **MUST NOT** be considered proof of the message being read by a recipient. If a read receipt capability is required, this should be provided through a separate cryptographic enforcement mechanism such as Mikali fair contracts protocol.

3.3. Asynchronous Collaboration

Asynchronous collaboration in closed communities presents a subset of the requirements of a global scale social network.

For example, consider the case of a group of authors collaboratively editing a document through an annotation service. Authors read the document and associated annotations published by the Annotation Host and post their comments to the Annotation Host in response (Figure 5).

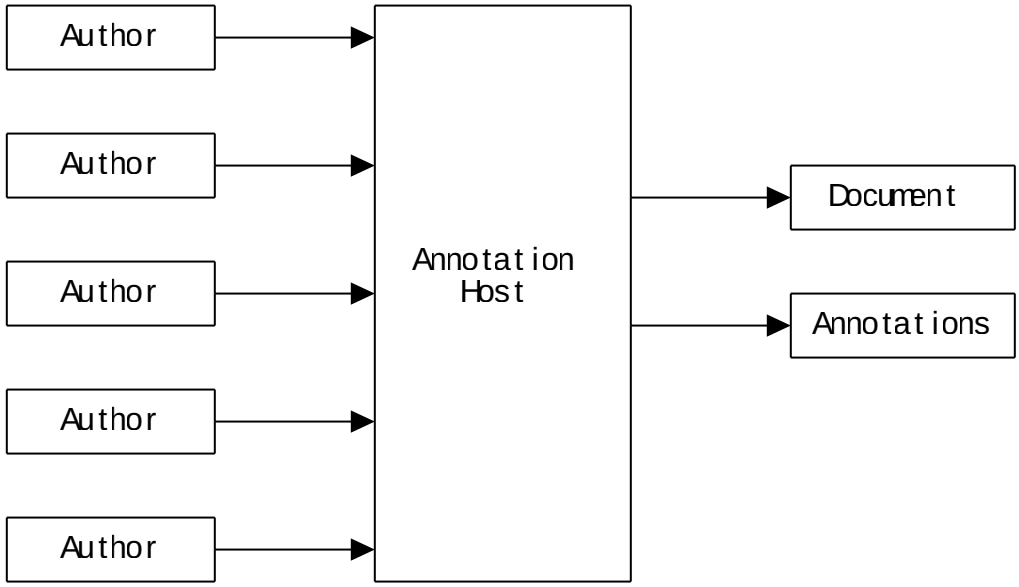

Figure 5: Document Annotation Service.

Comments made on a document **MAY** be tagged with a lightweight semantic as introduced by the [Open Meeting]. These tags provide context to tools designed to support document editors, highlighting parts of the document that need correction, clarification or further discussion.

3.4. Asynchronous Social Media

Asynchronous Social Media has rapidly established itself as one of the world's most popular forms of entertainment. But existing social

media based on proprietary walled garden forums have received increasing criticism.

In particular, the ability of the owner of the forum to control the agenda by selecting content users view and to suppress certain content entirely is not subject to any form of accountability towards the community established. Such circumstances are unlikely to endure.

Enabling social media interaction at global scale is a challenging proposition even in the proprietary walled garden model. In an Open Infrastructure such as Everything, great care must be taken to ensure that the system scales both technically and socially.

To achieve this scaling, it is useful to distinguish between the user in their content consuming role (Reader) and the user in their content creation role (Author). This in turn allows the functions of the social media platform to be broken down into a set of three distinct services (Publisher, Aggregator, Curator).

This architecture is shown in Figure 6 in which the content (original posts and comments) flows downwards and reader feedback flows up.

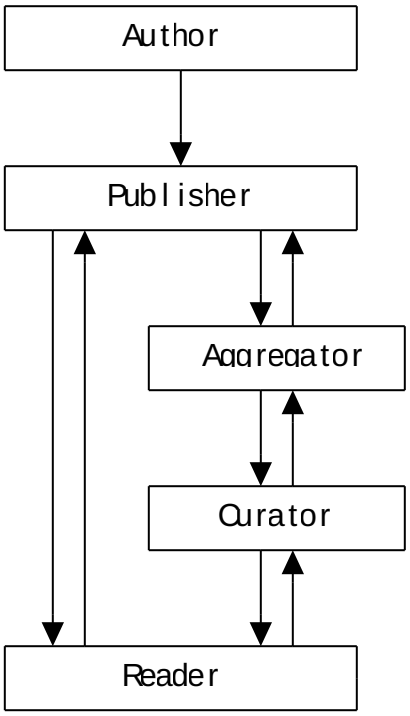

Figure 6: Social Media Roles.

3.4.1. Author Role

An Author is a participant that generates content either in the form of original content or comments. Each author chooses their content publisher and is free to change their decision at any time without switching costs. Authors can even set up their own Publication service. Thus, no author can be censored but neither is anyone forced to read the content they produce.

Authors may publish content through more than one publisher. They may also publish content to legacy walled garden social media through a gateway provider (Figure 7).

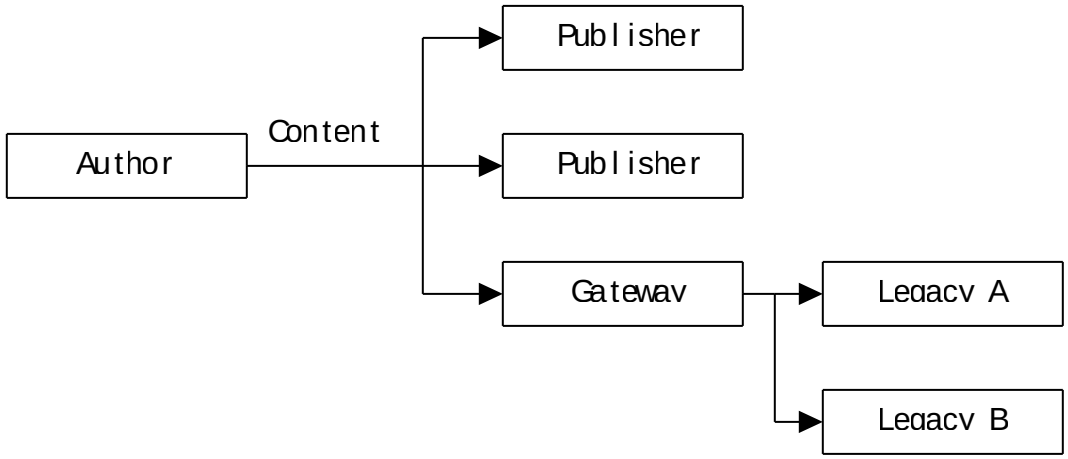

Figure 7: Author Role.

3.4.2. Reader Role

Readers select the content they consume, either directly from the publisher of individual author feeds or indirectly through a curation service which monitors multiple feeds selecting content likely to be of interest to their audience based on feedback notifications created by the reader (Figure 8).

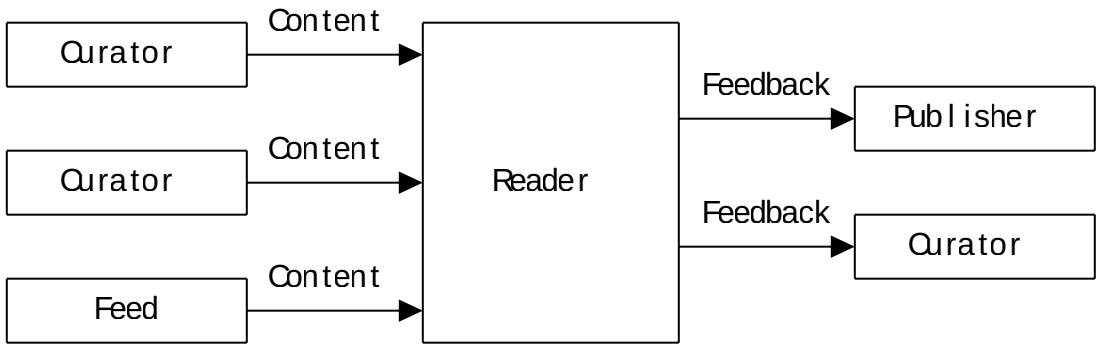

Figure 8: Reader Role.

Feedback consists of lightweight semantic labels drawn from a constrained vocabulary defined by the recipient of the feedback notifications. To ensure interoperability, a core set of feedback semantics {Agree, Disagree, Like, Dislike} is defined. The inclusion of positive and negative feedback is based on the observation that systems that only allow positive feedback are inherently unstable.

One objection commonly made to the 'reader choice' model is that the reader may not be the best judge of what content they consume. In particular, it is theorized that unless people are exposed to contrary viewpoints, they may adopt insular and uninformed viewpoints. The experience of a decade of social media based on such patronizing conceits is that the proprietors of such forums are most interested in promoting their own uninformed and parochial views. An environment in which the bully is able to force others to receive their ignorant views is quickly dominated by disinformation designed to suppress rational discourse.

3.4.3. Publisher Service

A publisher receives content from one or more authors, publishes it to the corresponding feed and forwards the corresponding update notification messages to the readers, curators and aggregators subscribed to the feed (Figure 9).

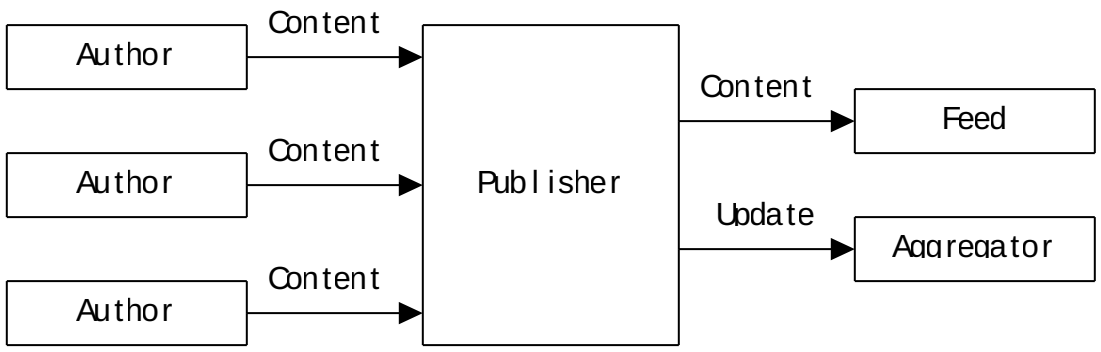

Figure 9: Publisher Service.

The only requirement to offer a publication service is to have a suitable host that is always available and has a static public IP address.

3.4.4. Curation Service

One of the chief challenges in global scale social media is enabling readers to find content they find enjoyable or is useful to them. In the proprietary walled garden model, curators face a conflict of

interest between presenting the content that will prove most profitable to them and the content their audience prefers. Experience has proved that it is na?ve to expect such conflicts to be resolved in favor of the user.

In the Everything model, readers select the curation services they wish to use from a competitive marketplace in which curators compete to best serve the reader's interests. Curators are free to adopt the business models and the technologies of their choice (Figure 10).

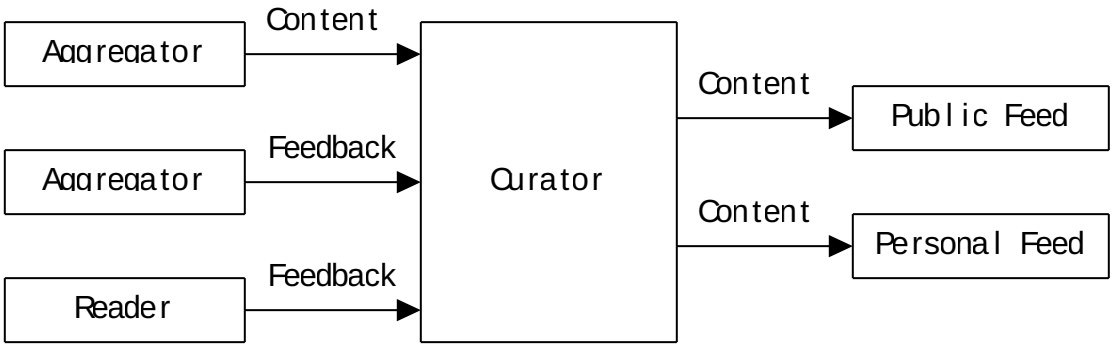

Figure 10: Curator Role.

For most participants, social media is primarily an entertainment medium. We must accept that fact and give them the right to choose what content they consume.

3.4.5. Aggregation Service

Aggregation Services act as brokers for notifications. Forwarding a notification to an aggregation service allows a publisher to avoid the need to track curators and vice versa (Figure 11).

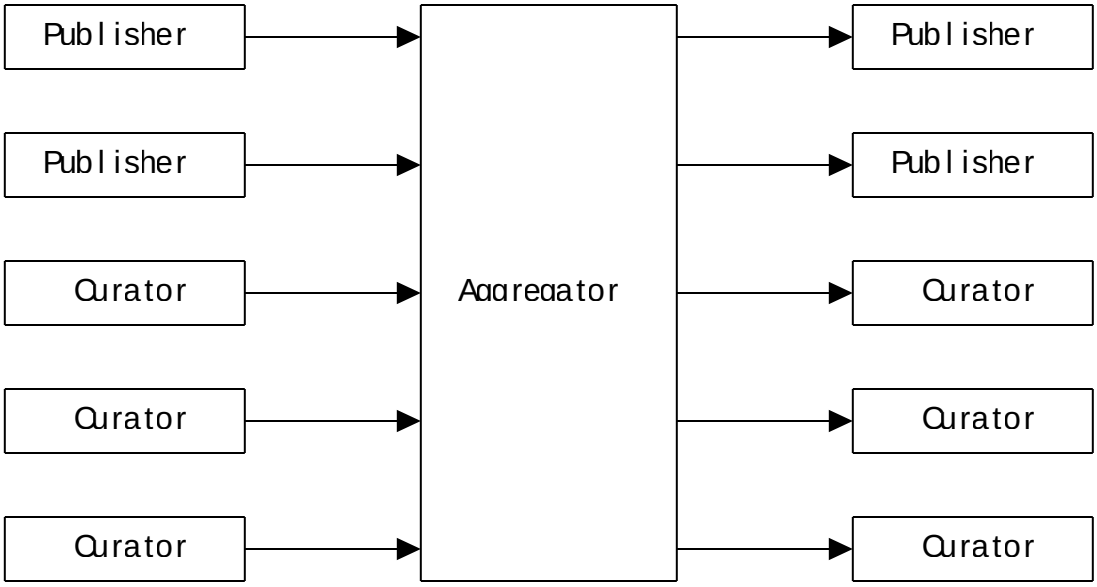

Figure 11: Aggregation Service.

3.5. Synchronous Messaging

Establishing a synchronous text messaging interaction between two users presents a number of challenges:

- The intended responder may not be available.
- The intended responder may not wish to accept the interaction request
- The intender responder may have multiple devices
- The intender responder may behind a NAT device that does support inbound connection requests.

These requirements are addressed through the use of a presence service. Whenever a device that is connected to a Mesh account is willing to accept inbound interaction requests, it maintains a constant connection to a presence service. To establish an interaction the initiator sends the request to the presence service servicing the account which performs an access control check on the request and if accepted, notifies the responder's connected devices. If the responder accepts the request, a bilateral communication channel is established between the initiator and the responder (Figure 12).

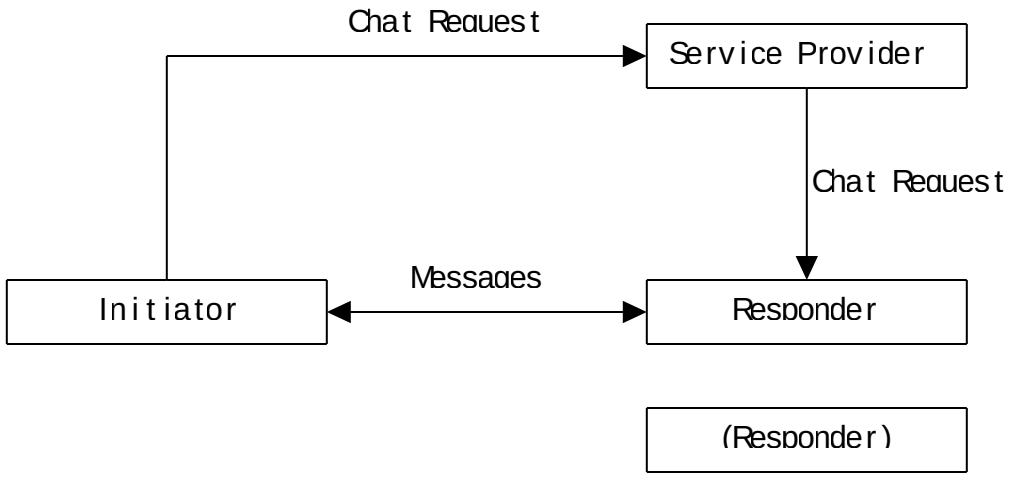

Figure 12: Synchronous Chat.

3.6. Synchronous Voice and Video

Establishing a synchronous voice or video interaction between two users presents essentially the same set of requirements as for text messaging but with the additional requirement of supporting streaming audio or video. Since the WebRTC protocols already provide this capability, it suffices to use the Mesh Presence service to establish the connection and perform the necessary key exchange (Figure 13).

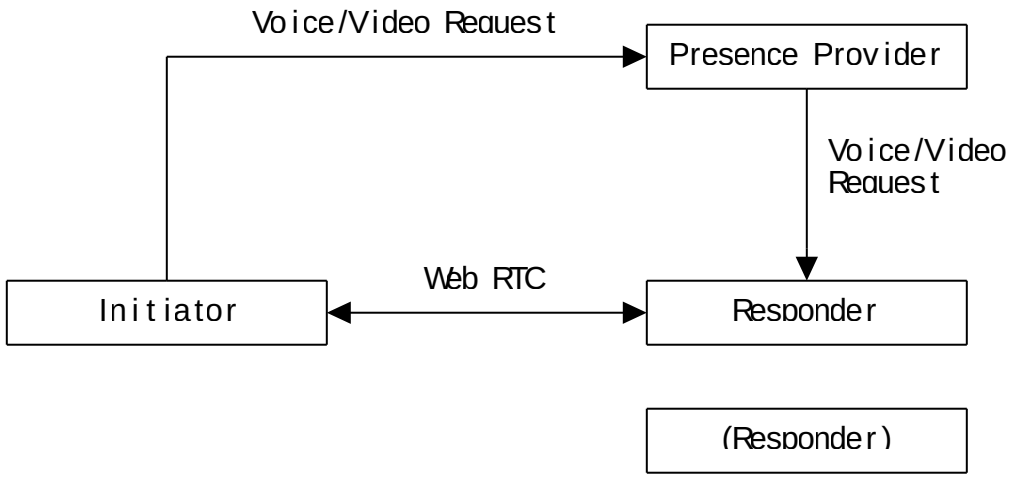

Figure 13: Synchronous Voice and Video using WebRTC.

One important limitation of this approach is that WebRTC is not designed to provide the same degree of traffic analysis resistance as the Mesh RUD transport. Another consequence of this design is that support for WebRTC capabilities on most platforms currently comes in the form of a captive Web Browser widget that is invoked by

a parent application. While this approach offers all the power and rich functionality of Web technologies such as CSS and JavaScript, it also presents the accompanying attack surface.

3.7. Conferencing

Conferences and meetings offer streaming content but are structured as an interaction in which participants interact with a central host (Figure 14).

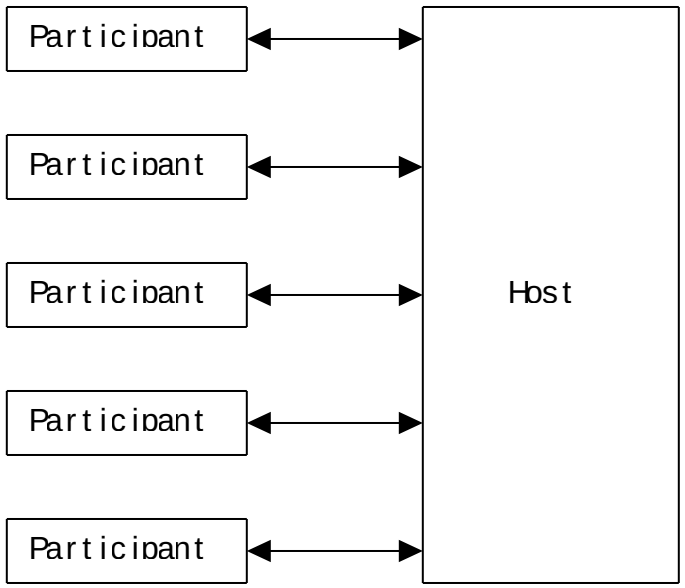

Figure 14: Synchronous conference.

While Everything is in principle capable of peer-to-peer meeting model, the social scaling challenges for structuring conversations of more than four people make use of such a scheme implausible.

4. ETML

Everything Text Markup Language is a lightweight document markup designed for use in social media interactions. Unlike HTML 4.0 and later version which are primarily intended to provide presentation capabilities, ETML is focused on representing content and structure.

One consequence of this more limited focus is that the ETML schema makes minimal use of attributes.

id Element identifier, may occur in paragraph type elements

Class Class identifier, may occur in paragraph type elements.

The schema for ETML is shown in Section 9.

4.1. Level 1

ETML Level 1 is intended for representation of chat messages, comments and other short messages. There are two top level elements:

*<p> Paragraph containing text data.

*<ps> A sequence of paragraphs

The <p> element may contain any of the following elements:

Element	Name	Description
i	Italic	Italic text
b	Bold	Bold text
u	Underline	Underlined text
del	Deleted	Deleted text
ins	Inserted	Inserted text
tt	Fixed font	Fixed font
a	Anchor	Link to external resource
sub	Subscript	Subscript
sup	Superscript	Superscript
dfn	Definition	Mark surrounding paragraph as the definition of the term contained
img	Image	Image data
svg	SVG	Inline Scalable Vector Graphics markup (see TBS)
video	Video	Video data
bdi	Bidirectional	Bidirectional Isolation.
alt	Alternative	Contains a list of content blocks containing alternative descriptions of the same content.

Table 1

[]

4.2. Level 2

ETML level 2 contains the elements of level 1 with the addition of structural elements to identify titles, headings, lists, etc. typically used in email messages, blog posts and other short articles. There is one top level element:

Post Post containing structured text data.

The additional tags specified in the level 2 repertoire mostly define specialized forms of paragraph style.

Element	Name	Description
post	Post	Post containing structured text data.
title	Title	

Element	Name	Description
		Post title, a post may only have one title. On a mail message, the title is the subject.
subtitle	Subtitle	Subtitle, a post may have multiple subtitles
h1	Heading 1	First level heading
h2	Heading 2	Second level heading
h3	Heading 3	Third level heading
h4	Heading 4	Fourth level heading
dt	Tag	Defined tag
dd	Data	Defined data
li	Bullet	Bulleted list item
ni	Numbered	Numbered item
pre	Preformatted	Preformatted text used for examples, etc.
quote	Quotation	Incorporate element from another post
cite	Citation	Source citation.

Table 2

Tables are omitted from the Level 2 model on account of table editing capabilities presenting a considerable degree of complexity.

4.3. Level 3

ETML Level 3 contains additional structural elements required in the production of articles, books and reports. There are two top level elements:

document Contains a complete document

fragment Contains a complete or partial document.

The Level 3 markup adds features such as the ability to nest lists, add footnotes, endnotes, and captions and to mark document sections.

Element	Name	Description
document	Document	Container for document elements.
fragment	Fragment	A document fragment which contains documents elements but is not necessarily a complete document.
cap	Caption	Captioned paragraph, image or video item
meta	Meta	Metadata describing the document.
tp	Tagged	Tagged paragraph belonging to a particular collection.
list	List	Container for list elements
foot	Footnote	Footnote presented in body of the text
end	Endnote	Endnote presented at the end of a document
section	Section	Section heading, used to divide longer document into sections
author	Author	Identifies an author

Element	Name	Description
abstract	Abstract	Marks a set of paragraphs as an abstract
preamble	Preamble	Document preamble
main	Main	Marks the beginning of the main section of a document that begins with a foreword or abstract
postamble	Postamble	Marks the end of the main section of the document. Headings within the postamble will be marked as appendices.
table	Table	Table, as in HTML model

Table 3

The ETML level 3 markup borrows heavily from the document markup specified in [RFC7991]. ETML documents may be converted to RFC7991 format and back again with little loss of information.

The <tp> element may be used to specify a tagged paragraph. Tagged paragraphs may be used to mark a paragraph as a lemma or theorem.

4.4. Inline SVG

ETML paragraphs and paragraph sequences **MAY** include an SVG element containing SVG content with the following restrictions:

* The <script> element **MUST NOT** be present.

* The ability to transfer active content through a messaging system is neither necessary nor desirable and is inherently unsafe. Clients **MUST** reject ETML markup containing a <script> element.

5. Everything Notification Message

5.1. Shared Classes

The following classes are used as common elements in Mesh profile specifications.

5.1.1. Structure: Header

Describes the feed

Title: String (Optional)
The name of the channel.

Link: String (Optional) The URL to the HTML website corresponding to the channel.

Description: String (Optional) Phrase or sentence describing the channel

Language: String (Optional) The language the channel is written in. As specified in the HTM: lang attribute.

Copyright: String (Optional) Copyright notice for content in the channel.

ManagingEditor: String (Optional) Email address for person responsible for editorial content.

WebMaster: String (Optional) Email address for person responsible for technical issues relating to channel.

Category: String [0..Many] Specifies one or more categories that the channel belongs to.

Generator: String (Optional) A string indicating the program used to generate the channel.

Docs: String (Optional) A URL that points to the documentation for the format used in the RSS file.

cloud: String (Optional) Allows processes to register with a cloud to be notified of updates to the channel, implementing a lightweight publish-subscribe protocol for RSS feeds.

Image: String (Optional) Specifies a GIF, JPEG, PNG or SVG image that can be displayed with the channel

Rating: String (Optional) A PICS rating for the channel.

TextInput: TextInput (Optional) Specifies a text input box that can be displayed with the channel for HTML or RSS feedreader interaction.

skipHours: String (Optional) A hint for aggregators telling them which hours they can skip. This element contains up to 24 hour sub-elements whose value is a number between 0 and 23, representing a time in UTC, when aggregators, if they support the feature, may not read the channel on hours listed in the skipHours element. The hour beginning at midnight is hour zero.

skipDays: String (Optional)
A hint for aggregators telling them which days they can skip. This element contains up to seven day sub-elements whose value is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or Sunday. Aggregators may not read the channel during days listed in the skipDays element.

5.1.2. Structure: TextInput

Provided for compatibility with legacy RSS readers.

Title: String (Optional) The label of the Submit button in the text input area.

Description: String (Optional) Explains the text input area

Name: String (Optional) The name of the text object in the text input area.

Link: String (Optional) The URL of the CGI script that processes text input requests.

5.1.3. Structure: Footer

Inherits: Header The publication date for the content in the
PubDate: DateTime (Optional) channel.

LastBuildDate: DateTime (Optional) The last time the content of the channel changed.

5.1.4. Structure: Item

Base class for feed items.

Author: String (Optional) Identifies the author of the item. Defaults to the value specified in the feed descriptor.

PubDate: DateTime (Optional) Indicates when the item was published.

5.1.5. Structure: Feedback

Inherits: Item
A response item consisting of only a subject specified and semantic, no content is included.

Subject: String (Optional) The URI of the subject matter being commented on.

Semantic: String (Optional) The feedback semantic

5.1.6. Structure: Comment

Inherits: Item

A response item consisting of a short form response to a single subject.

ContentType: String (Optional) The content type.

Text: Binary (Optional) The comment content.

5.1.7. Structure: Publication

Inherits: Feedback

Publication notice for a document that is not a comment on another publication or comment or is of extended length.

Title: String (Optional) The title of the item.

Link: String (Optional) A URI from which the content may be retreived. This is used for longer content such as a document.

Description: String (Optional) The item synopsis.

Category: String [0..Many] Includes the item in one or more categories.

Comments: String (Optional) URL of a page for comments relating to the item.

Enclosure: String (Optional) Describes a media object that is attached to the item.

Guid: String (Optional) A string that uniquely identifies the item.

Source: String (Optional) The RSS channel that the item came from.

6. Security Considerations

6.1. Confidentiality

6.1.1. Traffic Analysis

6.2. Integrity

6.2.1. Bullying

6.2.2. Sybil attack (brigading).

6.3. Service

6.3.1. Censorship

6.3.2. Data Loss

7. IANA Considerations

This document requires no IANA actions.

8. Acknowledgements

9. Appendix A: ETML Schema

TBS

*

10. Normative References

[draft-hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", Work in Progress, Internet-Draft, draft-hallambaker-jsonbcd-23, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-jsonbcd-23>.

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", Work in Progress, Internet-Draft, draft-hallambaker-mesh-architecture-21, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-architecture-21>.

[draft-hallambaker-mesh-callsign]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Mesh Callsign Service", Work in Progress, Internet-Draft, draft-hallambaker-mesh-callsign-02, 23 October 2022,

<https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-callsign-02>.

[draft-hallambaker-mesh-dare]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part III : Data At Rest Encryption (DARE)", Work in Progress, Internet-Draft, draft-hallambaker-mesh-dare-16, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-dare-16>.

[draft-hallambaker-mesh-presence]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part XI: Mesh Presence Service", Work in Progress, Internet-Draft, draft-hallambaker-mesh-presence-01, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-presence-01>.

[draft-hallambaker-mesh-rud]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VI: Reliable User Datagram", Work in Progress, Internet-Draft, draft-hallambaker-mesh-rud-02, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-rud-02>.

[draft-hallambaker-mesh-udf]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-17, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-udf-17>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[RFC822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <https://www.rfc-editor.org/rfc/rfc822>.

11. Informative References

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-

hallambaker-mesh-developer-10, 27 July 2020, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-developer-10>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/rfc/rfc7991>.