Workgroup: Interpeer Project
Published: 30 November 2023
Author: J. Finkhaeuser
Interpeer

Interpeer -- a Human-Centric Networking Architecture

Abstract

This document describes a novel, human centric networking architecture designed to better meet issues arising on the current Internet. The architecture is derived from an examination of existing architectures and their disadvantages in meeting the above issues.

The RFC Editor will remove this note

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://specs.interpeer.org/draft-jfinkhaeuser-interpeer/.

Discussion of this document takes place on the Interpeer mailing list interpeer@lists.interpeer.io, which is archived at https://lists.interpeer.io/pipermail/interpeer/. Subscribe at https://lists.interpeer.io/mailman/listinfo/interpeer.

Source for this draft and an issue tracker can be found at https://codeberg.org/interpeer/specs.

Status of This Memo

Drafts are working documents of the Interpeer Project. The list of current Drafts is at https://specs.interpeer.org/. Drafts may be updated, replaced, or obsoleted by other documents at any time. It is inadvisable to use Drafts as reference material or to cite them other than as "work in progress."

Copyright Notice

Table of Contents

1. Introduction

This document describes a novel, human centric networking architecture; In much the same way that the Internet Protocol suite pivots around IP packets, this architecture pivots around the notion of a resource, shared and collaborted in a user group. This shift in pivot point enables a properties of the architecture that go significantly towards addressing the listed problems.

The document first addresses the problems in Section 2. Section 3 provides use cases for the existing and emerging Internet as a guideline for what an architecture needs to support. A gap analysis framework, as well as an analysis of existing architectures is provided in Section 4. This section also lists the properties one desires of an architecture in order to meet the discussed issues (they can be understood as requirements). Finally, Section 5 describes a novel architecture that induces all of these properties.

2. Problem Statement

Technology can never fix problems of society. At the same time, technological and societal change tend to occur hand-in-hand in history: either the solutions to pressures in society require technological advancement. Or the invention of some technology enables a societal change, the need for which may only be fully understood later.

The currently primary Internet-enabled technology -- the World Wide Web (WWW) -- so contains problems that require solving. The issues discussed here are fundamentally societal in nature, or they are acerbated by current social pressures.

It is easy to dismiss such non-technical issues in a technical forum; one should only focus on the hard facts. One such hard fact is that technology is created by and for humans. Another hard fact is that humans are not infallible. The fields of psychiatry and psychology deal with common failure modes of humans, and assign names for commonly identified behavior patterns.

"Cognitive inertia" [COGNITIVE-INERTIA] describes the difficulty in changing mental direction, analogous to how inertia in physics is the idea that objects keep moving in the same direction and at the same speed until something compels them to change. Similarly, "decision fatigue" [DECISION-FATIGUE] describes the effect that a rested mind finds it easier to assess facts and perform such changes in direction -- making decisions --, while being faced with decision after decision fatigues the mind. Having to come to a decision is akin to a kind of mental crisis that requires resolution for well-being. So when fatigue sets in, the mind may prioritize quick resolution over the most desirable long-term effects.

It is therefore well understood that, as a matter of probability, humans tend to follow the path of least resistance. The implication for engineers is that "a system is what a system does". That is, when acting within the constraints of a technical system humans are prone to making certain decisions, it follows that the system's architectural constraints induce this behavior.

While this section started with stating that technology can never fix problems of society, the above strongly suggests that it can contribute to these problems. It is no stretch to imagine, then, that it can also contribute to the avoidance of the self-same problems by adjusting the system's architectural constraints.

2.1. Issues

The issues in this section may be societal in nature, but they also provide the context for technological issues outlined in Section 4. In particular, they provide a lens through which to view the relationship between architectural constraints of a system, the properties it induces, and a focus for evaluating whether those properties are desirable.

2.1.1. Surveillance Capitalism

The term is popularized in the 2019 book "The Age of Surveillance Capitalism" [AOSC], and is defined on Wikipedia as follows:

> Surveillance capitalism is a concept in political economics which denotes the widespread collection and commodification of personal data by corporations. [SURVEILLANCE-CAPITALISM]

Surveillance capitalism can only thrive in a system where personal data is easily available. Additionally, "data" is relatively worthless in itself until it is linked to other data, such as linking a name to the purchase of some medication. Once the link is established, one can infer further information, such as that the person making the purchase likely suffers from a condition that the medication is commonly prescribed for.

One cannot link data without collecting it, which means that data collection is the activity that fuels the establishment of more links, and thus the opening of more potential revenue streams.

Attempts at curbing surveillance capitalism through policy tend to focus on company size or monopoly position. However, as Tarnoff notes in "Internet for the People" [IFTP], it is competition that drives data collection. Specifically, it is the competitive motivation to discover new or better monetizable links between individual data points.

2.1.2. Information Warfare

According to Zuboff, data collection is not necessarily problematic in itself. The problems arise during usage: when personal profiles are used to target advertising to the recipient, it can also be used to target misinformation, and thus drive decision making processes. Either is monetizable, and the logic of capitalism dictates that such monetization avenues must be exploited.

Surveillance capitalism, in other words, enables power brokerage to become indistinguishable from demagogy. It is no longer necessary to target specific powerful politicians to influence them, when instead one can manipulate their electorate.

The event that showed how this is not just a possibility, but was put into practice is the Cambridge Analytica scandal. Psychographic profiling of Facebook users allowed the Trump campaign team to motivate voters into voting against their own best interest. And yet, the FTC response to this event "has been criticized as failing to adequately address the privacy and other harms emanating from Facebook's release of approximately 87 million Facebook users' data, which was exploited without user authorization." [CAMBRIDGE-ANALYTICA]

Meanwhile, these tactics are further expanded upon in so-called "hybrid warfare" that bridges the battlefield and disinformation campaigns [HYBRID-WARFARE]. The basis, however, remains the same: access to user information that permits to identify demographics vulnerable to disinformation attacks.

2.1.3. Centralization

Where surveillance capitalism describes the market mechanics that lead to data collection en masse, it fails to describe the more technical effect this has: the Web becomes increasingly centralized.

Web centralization has multiple contributing factors. One of the more fundamental ones, however, is that it is significantly easier to collect more data when more data is funneled through a centralized service. Competitive advantages are gained when one provides this service.

It is necessary to distinguish here between different kinds of centralization:

1. A service may be conceptually centralized, but run in multiple locations, both in terms of network topology and geography.
2. A service may run in multiple geographical locations, but run within the same autonomous system.
3. A service may run in a single location.

These different categories of centralization exhibit different levels of vulnerability, but they all carry the same risks:

- It becomes easier to disrupt services when they are centralized.
- It becomes easier to gain access and maliciously harvest data when services are centralized.

The above describe failure modes of centralization. Proponents of centralization argue that mitigating against such failures also becomes easier with centralization. While true to an extent, similar effects can be achieved with processes and tooling, without introducing the risk of these specific failures.

2.1.4. Oppression & Genocide

The Internet is a great enable for communities to come together; this is in particular the case for minority communities that have no other representation in popular media. The Uyghur of China are such a minority, whose usage of the Internet to keep their identity alive has been well documented [UYGHUR-INTERNET].

Equally well documented, unfortunately, is the "cultural genocide" of the Uyghur [UYGHUR-WAR]. The role of the Internet in this is just as central as it was in bringing the community together in the first place. Reports indicate that China is using artificial intelligence operating on personal profile data to identify Uyghur and to target them methodically [UYGHUR-AI].

From the perspective of Internet engineers, it is not necessary to understand the ins and outs of a specific political situation. What is required is the broad understanding that collecting large amounts of personally identifiable information (PII) in a central location enables devastating misuse. The consequence then must be to protect PII and avoid centralization to counteract this.

2.1.5. Machine Learning

The role of artificial intelligence in Section 2.1.4 is one use where access to PII can be abused. The article "Artificial Intelligence, Advertising, and Disinformation" [ML-DISINFORMATION] lays out the overlap between these technologies in more detail.

It is one thing where machine learning is used to impersonate a politician as part of a disinformation campaign. But images and videos of politicians are public goods, as such persons partially give up their right to privacy in being public figures.

Given access to PII, the exact same technology can be used in more personal cyber attack scenarios: for example, access to voice recordings can allow an attacker to use a cloned voice in an attack scenario [ML-VOICE].

The ramification is that with more PII available, ML-based attacks or attacks making use of ML techniques evolve to exploit this access.

2.1.6. State Control over Media & Censorship

Whereas previous examples focused to a large degree on availability of personally identifiable information which may be acerbated by centralization, centralization poses an additional risk all by itself: centralization aids censorship.

The CensoredPlanet project [CENSORED-PLANET] monitors censorship of the internet around the globe, using a variety of techniques. However, they all revolve around measuring access to parts of the internet, such as the parts serving a news site, or an entire country's network.

It's worth highlighting that access is not necessarily binary here. Rather than blocking access, a particular path may merely become slowed down to the point where Internet users prefer to turn to other resources instead.

In either case, censorship relies on identifying targets to censor. The more access to targets is funneled through centralized choke points, the easier it is to apply censorship in those locations.

In many cases, this can be as simple as government buying media outlets outright, a practice that has led to the establishment of the Media Development Investment Fund (MDIF) [MEDIA-INDEPENDENCE]. In this latter form, centralization affects media production in any given outlet than access to the results.

It is necessary to consider the entire pipeline from media creation to consumption, however. Arguably the role of media outlets has historically been to collect potential news, filter this for some notion of "quality" to bring a subset of this source material to production, and then to distribute it again.

In a fully digitized world, collection and distribution find equivalences in ingress and egress traffic, while the news production itself is a data processing task. In other word, the media outlet model is centralized largely because the data processing tasks would historically have been impossible to distribute. This centralization, however, is also the cause for its vulnerability to state control and censorship.

It is imaginable, then, that if data processing tasks are easy to distribute, and the Internet offers the infrastructure to do so, that media censorship would become a significantly harder endeavor.

2.1.7. Human Rights

The issues listed in the previous sections are concrete examples of the impact properties of Internet technology have on the physical world and human beings living within it. They all relate to human rights in some fashion.

A more complete list of the impact of protocol design decisions is maintained by the Human Rights Protocol Considerations within IRTF. In particular, documents such as [I-D.draft-irtf-hrpc-guidelines-20] provide practical considerations for protocol design.

Section 4 will refer to this document in more detail.

2.2. Additional Context

2.2.1. Internet vs. Web

In the above text, and the rest of this document, the term "Internet" and "Web" (referring to the World Wide Web, or WWW) are used more or less interchangeably.

It is clear to the author(s) that these are distinct technologies, and that from the Internet's point of view, the Web is but one of many application protocols.

At the same time, in practice the Web is used almost ubiquitously from the point of view of Internet users. This therefore raises the question why this has come to be?

The arguments for or against this state of things are too numerous to list here. To provide a simpler lens, consider the following statement: "The Internet connects machines. The Web connects people."

In fact, no parts of the Internet protocols are particularly concerned with people. Addressing happens on a per-machine basis -- ignoring for the moment the ability to address more abstract things such as multicast groups or more concrete things such as the link on the machine that is configured to respond to a particular address.

Web technology, on the other hand, is concerned with what it terms "resources", a malleable concept that can represent digital or physical "things" as well as oneself or other people's digital identities. Furthermore, through user based authentication methods, the Web firmly establishes itself as being concerned with bridging between the purely digital and the physical or hybrid worlds.

If this statement is true, then the prevalence of Web-based applications may simply be explained by the fact that humans are trying to solve human problems with technology, and this often involves having a notion of how a human may be represented in the digital realm.

Arguable, then, from a human perspective the distinction between the Internet and the Web is moot (unless you are an engineer). This "end-user perspective" demands that future Internet evolution is free to adopt the concepts of resources and user identification.

Doing so may not only open avenues for evolution that the current stricter split of concerns keeps firmly closed. It also permits the Internet, rather than one of its applications, to become the substrate for transporting people's actual, real world concerns.

2.2.2. Generative Systems vs. Tethered Appliances

In "The Future of the Internet" [FUTURE-INTERNET], Zittrain describes what he calls "tethered appliances" and "generative systems". In this definition, a tethered appliance, like a kitchen appliance, fulfills a strictly limited set of functions, determined by the manufacturer. It may be "tethered" in the same way that telephone sets used to be distributed by Bell/AT&T, intrinsically linked to the purchase of a phone line.

Zittrain contrasts this to "generative systems" such as the Personal Computer (PC). Here, a semi-finished product with no particular purpose other than to provide compute resources was brought to market -- and flourished, and in so doing changed the world.

He argues that the Internet is such a generative system. When it came to be, few could envision the impact it would have on the world today. He identifies as the reason that, having no specific purpose, the Internet was open to be used for whichever purpose its users desired.

Generative systems not only offer incredible flexibility, and allow for providing solutions to age-old problems. Human ingenuity will also be able to monetize such solutions. Zittrain observes that the businesses that profit most from the generative nature of a system eventually reach a

point where innovation no longer matters; instead, the logic of capitalism dictates that efforts now need to be expended to shut out competition. In effect, it is in the same businesses' interest now to turn the erstwhile generative system into an appliance tethered to their business model.

This view is picked up and largely confirmed in the latter "Internet for the People" by Tarnoff. Where the earlier book predicts the direction the Internet may evolve in, the latter confirms its evolution.

Both authors agree, each in their own terms, that the way to "save" the Internet is to re-focus attention on what made it generative in the first place: it was a substrate for anything.

The Web then has shown us that "anything" tends to refer to human concerns in practice.

2.3. Summary

The Internet and Web are both generative; this means they can and will be abused to create or acerbate societal issues explored in this section.

The section also briefly explores that meeting these abuses with legislation can only be part of the answer. One of the reasons is that legislation is slow compared to technological advancement -- but far more sinister is that fact that legislation can itself work against people's best interests.

By no accident, the examples also focus on three major contributors to abuse:

1. Centralization, either by itself or as an amplifier the following points.
2. Unwarranted access to personally identifiable information (PII).
3. Denial of access to data in general (which may include PII).

We'll explore properties of the Web that contribute to these issues in more detail in Section 4. Before that, Section 3 focuses on how the Web and Internet are currently used, which need to be preserved in any alternate proposal.

3. Use Cases

This section presents use cases to consider, or rather use case classes. While actual use cases are a powerful motivator, we limit ourselves to one or few as a proxy for an entire class.

Since much of Section 2.1 is concerned with the Web, Section 3.1 focuses on Web related use cases which a new system must necessarily capture if it is to be a viable alternative. Section 3.2 contains additional considerations that are not ideally fulfilled by the Web at this time.

3.1. Web Use Cases

The Web started out with a fairly simple idea -- but the generative nature of it then quickly prompted new uses other than the originally envisioned. We identify three relatively distinct classes of use cases for the current, modern Web.

3.1.1. Document Web

The original use of the Web was dissemination of knowledge in the form of publication of documents. This is effectively what the PUT, GET and DELETE methods of HTTP ([RFC7231], Section 4.3) embody: a means to store, retrieve and delete documents from a Web server.

The scope of these operations is an entire resource. HTTP permits optional Range requests [RFC7233] to target sub-resources, but here an interesting dynamic prevents its use across a wide variety of use cases.

On the one hand, the [REST] architectural style that HTTP implements requires that data transferred be "representational". The idea is that it is up to the service implementor how to persist data, which representations to send, and which to accept. However, it is explicitly not implied that the byte sequences that the service sends and receives are identical to the byte sequences it persists.

On the other hand, the Range header operates on byte ranges. Mapping byte ranges of a data representation that differs from the byte ranges of a data storage format onto each is no easy task; it should therefore not be surprising that Range headers are most often used when the representation matches the storage format, i.e. the methods operate on Binary Large Objects (BLOBs).

In summary, the Document Web use case class is best described as one offering simple operations for manipulating entire resources (or their representations). This is likely why the "RESTful" design style (not to be confused with [REST] itself) is characterized by mapping the PUT, GET, POST and DELETE methods directly onto Create, Read, Update and Delete (CRUD) operations. Even though HTTP offers other uses with such extensions as the Range header, it is an uncomplicated and therefore easy to adopt mapping.

3.1.2. API Web

The next use case class treats the Web as a remote procedure call (RPC) protocol, in which resources or resource collections are mostly referred to as application programmer interfaces (APIs) today.

The API Web is not fundamentally distinct from the Document Web in the HTTP methods it employs. But API endpoints (resources) no longer represent a document or document type, but a functionality the client wishes to invoke. As such, the data representation format chosen tends to reflect the needs of APIs, where structured data is transmitted, which may refer to multiple other resources ("connect foo to bars A, B and C").

APIs, as the name suggest, provide interfaces also between distinct engineering teams. As such, common standards have been created and discarded across the existence of the Web, such as [SOAP] and the currently popular [OPENAPI]. These provide interoperability by layering a protocol for specifying RPC invocations onto the HTTP protocol.

In reality, services will usually provide a mixture of API and Document Web functions. The main conceptual distinction between the two use case classes lies in the expectations around the meaning and freshness of a response.

Documents are by nature fixed and self-contained. They can be revised, and refer to other documents to understand them. But each revision is effectively a new document, albeit sharing a history with its predecessors.

An API response, on the other hand, is ephemeral. It describes the result of an operation. The same operation performed at another point in time may yield a different result.

In HTTP, this difference is expressed in caching. The standard provides many headers relating to the longevity or freshness of a response. In Document Web use cases, it can typically be assumed that a response has a fairly long validity -- while in API use cases, this is not a valid assumption.

API Web then differs from Document Web in that the resources one accesses represent functions rather than documents, and the responses it produces are ephemeral and contextual rather than self-contained and long-lived.

3.1.3. Real-time Web/Streaming

Beyond the Document and API Web, there exists also a class of use cases related to data streaming.

Streaming is itself a term with somewhat ambiguous meaning. In our case, let's interpret it as one party in a network transaction consuming some related data (such as a resource), before the other party has finished producing it. Consuming and producing can be understood as receiving and sending data over the network, but may also include processing on either side to create or display the resource in some fashion.

In principle, this can be mapped onto HTTP in arbitrary ways. Range header usage is predestined for this sort of use, but it is equally possible to structure the resource into individual documents that are requested in sequence. Finally, repeated calls to the same or different API endpoints may produce the data incrementally.

The precise mapping onto HTTP mechanics barely matters. The main point of this use case class is that there is a real-time component to it in a processing pipeline. Producers of data can produce data only at a certain pace. Transmission is bounded by the available throughput rate of the network path. Consumers may also only render data at a given pace.

What distinguishes the Real-time Web use case class from the above is that managing the network throughput rate and latency to match the capabilities and expectations of either of consumer, producer or both. This is distinct enough from the others to warrant its own use case class.

3.2. Additional Use Cases

There exist a number of use cases for which the Web is not typically adopted, or where adoption poses additional challenges that are not intrinsically resolved with its architecture or implementation. This section explores these in brief.

3.2.1. Resilience

Preceding the birth of the Internet, Paul Baran describes different communication architectures in [RM3420], which he terms "centralized", "decentralized" and "distributed". He comes to the conclusion that the "distributed" model offers the highest resilience against communications failures, which led to the packet switching paradigm of the Internet.

In the "distributed" model, communications nodes have connectivity with multiple other nodes. In order to communicate with any node, the data packets they send can traverse many intermediary nodes. When one such node fails, a different path can be taken.

By and large, the Internet remains distributed in nature. A number of incentives may push towards more centralization, but this is less to do with the Internet's architecture than the interests of Internet service providers.

The Web, for similar reasons, shows much stronger trends towards centralization. Works such as [I-D.draft-nottingham-avoiding-internet-centralization-14] assess the specific reasons, as well as what standards can do about them in far more detail than this document can.

Centralization introduces real-world risks, as explored in Section 2, some of which directly relate to notions of resilience, such as resilience to censorship. If the Web shows tendencies towards centralization, it follows that heightened resilience is a use case that is not typically captured by Web technology -- and yet may help mitigate issues raised above.

3.2.2. Remote Locations

There is no particular argument for physical location to factor into Internet or Web usage -- but underlying protocols that facilitate connectivity may suffer in some geographical locations. This has follow-on effects for performance and user experience of the Web stack as a whole, which may negatively affect its suitability for a particular use case.

3.2.2.1. (Commercial) Drones

Vehicles (drones) operating in Unmanned Aircraft Systems (UAS) commonly fall into several categories; EASA has standardizes these categories within Europe. On one end of the spectrum lie small drones operated via remote control. On the opposite end lie large drones with high payload capacity, typically military in nature.

Projections predict commercial innovation to occur predominantly in between these extremes; under EASA rules, this would be termed the "specific" category.

In this category, drones are characterized by several factors. On the one hand, they must be large enough to have a useful carrying capacity, which renders them heavy enough to be dangerous when they fail. On the other hand, they must typically operate Beyond Visual Line of Sight (BVLOS), or else their usefulness is questionable compared to sending a person. Finally, they must be reasonably cost effective to purchase and operate, which strongly suggests that Commercially available Off the Shelf (COtS) components should be used in their manufacture.

This proves to be some conundrum to maintaining Command, Control and Communications (C3) links to the vehicles. Suitable technology such as found in mobile devices does not provide the safety requirements mandated for such links by regulators. To mitigate this, failover solutions between such link technologies appear to be the most likely solution [DRONECOMMS].

Connectivity of an individual link can fail for a variety of reasons, such as interference or obstacles -- or simply distance. In remote locations, for example, it is reasonable to assume that 802.11 connectivity is not a given, while satellite based systems may be available.

3.2.2.2. Internet-of-Things (IoT)

In accessing Things on the Internet, [RFC7252] is modelled after [REST]. This is because of a combination of two assumptions, one being that communications with the thing itself underlies constraints that do not occur on the rest of the Internet. The other is that REST is the default method for accessing resources.

As such, it is not surprising that most IoT architectures envision the Things to be accessed via a gateway node that translates from e.g. HTTP to e.g. CoAP and back. A common scenario is that a number of constrained sensor devices communicate over a limited range to some gateway via a protocol such as CoAP. The gateway then is either permanently or intermittently connected to the cloud, where other machines can query it for (aggregated) sensor data.

3.2.2.3. Space Communications

Deep space, as the ultimate remote location, has prompted the development of [RFC4838], the Delay-Tolerant Network (DTN) Architecture and related implementations. Space communications has fundamentally different approaches to latency and intermittency of communications than most earthbound solutions. Whereas in near space such as Low Earth Orbit (LEO), DTN can be used to merely encapsulate regular IP-based traffic to its destination, the same approach may not work when latency exceeds the expectations of the application -- in other words, different approaches for designing applications are needed, which then put into question the use of Internet technology altogether.

3.2.2.4. Generalization

The generalization of the issues with remote locations, as exemplified by the drone example, is that network attachment points may change or vanish at any time. Similarly, it is possible to utilize multiple network attachment points in parallel when available.

3.2.3. Energy Usage

Of growing importance is the energy usage of any networked environment. Using renewable energy for e.g. hosting [GREEN-HOSTING] is commendable, but is hard to build into networking protocols. At the protocol design level, however, it is possible to build less overall energy usage into a system.

The approach, dubbed "green coding", is gaining attention in recent years ([GREEN-CODING-1], [GREEN-CODING-2], [GREEN-CODING-3], etc.). In the realm of networking protocols, the design approach reduces to a relatively simply method: reduce transmissions.

In practice, things are not as simple as that: measurements are needed to determine energy usage in a variety of scenarios. But each packet transmitted induces energy usage not only for the transmission itself, but also for the processors performing packet switching decisions. It is clear that reducing the packet transmission rate by design will have an effect on reducing energy usage.

One approach to this is to treat caching of remote data as a first class problem, such that transfer can be avoided as much as possible.

3.2.4. Data Protection

Increasingly, digital rights are seen as variants of fundamental human rights; in the European Union, the European Parliament and the Council of Europe jointly signed a European Declaration on Digital Rights and Principles, which some researchers consider "transformative" [DIG-RIGHTS].

This relatively local legislation reflects a larger trend in tending to digital rights of citizens worldwide, which inevitably influences data protection practices.

3.2.4.1. PII and GDPR

In the European Union, the General Data Protection Regulation [GDPR] has set the standard for data protection laws worldwide, with several legislations adopting comparable frameworks. It is focused on protecting Personally Identifiable Information (PII) and requires, amongst other things, that such data may only be collected with "informed consent".

A future architecture must be structured such that data sharing occurs with informed consent, or else risks running afoul of such requirements.

3.2.4.2. Whistleblower Protection

Similar to the GDPR, in some legislations there exist whistleblower protection laws, such as the German [HinSchG].

Using this example, the law requires confidentialy of whistleblower identities, and additionally requires that reports are handled anonymously -- the distinction implies that the report must be devoid of references to a reporter's real world identity, while such an identity may be confidentially managed in a separate data store.

3.2.4.3. Journalism & Source Protection

Aside from the protection of whistleblowers, the protection of sources of journalists is generally considered to be a fundamental component for press freedom. In Europe, courts have regularly held that reavealing sources would constitute a violation of Article 10 of the European Convention on Human Rights [ECHR], which is concerned with freedom of expression.

From a technical point of view, source and whistleblower protection has strong similarities. But it is worth highlighting that the legal frameworks they may refer to can be different, and so impose different requirements.

4. Gap Analysis & Requirements

In order to justify a novel architecture, it is prudent to examine which existing architectures may or may meet the issues outlined above. This requires the use of some evaluation framework.

This document uses [REST] as the evaluation framework, and expands on it. The REST document does not merely provide an architecture description; significant amounts of it are dedicated to describing desired architectural properties.

As the previous sections outline issues with the Web, it stands to reason to start with these properties when examining other architectures, including the one put forth later in this document. Additional desired properties are derived from the problem statements above.

Section 4.1 summarises this analysis framework. The following section Section 4.2 lists desired properties derived from REST and the additions above; these are effective requirements for a future architecture. Finally, Section 4.3 provides an analysis of several existing architectures in the light of these properties.

4.1. Gap Analysis Framework

We are analysing existing architectures, and so it's worth considering a common framework for doing so; unfortunately, these are hard to come by. One of the best -- and easily accessible -- can be found in [REST] Chapter 1, in which Fielding provides the following definitions. For a full explanation, please see that document.

Architecture: A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture.

Element: A software architecture is defined by a configuration of architectural elements -- components, connectors, and data -- constrained in their relationships in order to achieve a desired set of architectural properties.

Component: A component is an abstract unit of software instructions and internal state that provides a transformation of data via its interface.

Connector: A connector is an abstract mechanism that mediates communication, coordination, or cooperation among components.

Data: A datum is an element of information that is transferred from a component, or received by a component, via a connector.

Configuration: A configuration is the structure of architectural relationships among components, connectors, and data during a period of system run-time.

Style: An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.

This document does not decompose the architectures it refers to strictly according to this schema. But relating this scheme does help to highlight that it is the *constraints* placed upon elements that induces certain *properties* in an architecture. We can derive desired properties the problem statement (Section 2.1) and use cases (Section 3), and determine if and how those architectures have such properties.

In Section 5 we will then describe constraints to address any gaps.

4.2. Properties

Based on the framework outlined in Section 4.1 above, this section lists properties considered desirable.

4.2.1. REST

In lieu of citing the entire [REST] document, below is the list of properties it induces.

Performance: The performance (((!performance)) property can be further subdivided into:

- Network Performance , which refers to high throughput and low overhead characteristics.
- User-Perceived Performance , which refers to low latency, and low time to completion of a request.
- Network Efficiency , which makes the point that the most efficient thing is not to use the network at all, i.e. rely on cached data.

Scalability: The scalability property refers to the ability of the architecture to support large numbers of components and/or interactions.

Simplicity: The simplicity property may seem self-explanatory, but often people treat simplicity as a lack of complication. In systems theory, complicated systems can be solved, even if the solution itself is complicated. Complex systems, on the other hand, involve too many variables to solve completely.

For the purposes of this document, the goal should be to avoid complexity, even if the result is somewhat complicated.

Modifiability: The property of again refers to several more specific properties:

- Evolvability refers to the ability to change e.g. the implementation of a component without affecting the overall system.
- Extensibility refers to the ability to add new functionality safely.
- Customizability refers to the ability of the client to initiate server behaviour.
- Reusability refers to the ability to re-use components in the system.

Visibility: The property of requests that middle boxes should be able to monitor and mediate connections.

Portability: Portability in this document refers to the ability to move code along with data, so that the code can run locally on the data. On the Web, this means JavaScript code in current practice.

Reliability (REST): Here, reliability refers to resilience to (partial) failure of components.

As these properties are induced by REST, and REST -- via the Web -- is the current most prevalent application of the Internet, it stands to reason that a future architecture should likely also induce these properties.

4.2.2. Access

Access to Internet resources can be difficult due to a variety of societal reasons; it could be that network attachment points are unavailable or not affordable, that power supplies for networking equipment are not reliable, and so forth. For an exploration of societal factors, see e.g. [BRIDGES].

Where the above issues focus on earth bound issues, [RFC4838] considers cases where physics interfere with access. The most notable use case here is in space communications. Where round-trip times are measured in minutes or hours, and radio transmitters are turned of for power conservation reasons, access is simply not a given.

Intermittency: The above document focuses predominantly on the issue of intermittent access, whereby the (logical) link layer is frequently unable to establish connection.

We can state that the ability to deal with high intermittency is a desired property of a future architecture. But the ability to do so must not break existing usages where intermittency is low. We shall call these desired properties high intermittency tolerance and low intermittency utilization, respectively.

Latency: Related to intermittency is the problem of latency. In fact, it is possible to consider high latency as a result of intermittent link layer connections, or vice versa consider intermittency as situations where latency trends far outside of expected ranges.

That second view is useful because it speaks to the boundaries of the system design parameters -- when latency is *within* the expected ranges of the design, it is also possible to find solutions within the system. When latency exceeds those ranges, solutions need to be found outside of the system.

For this reason, it is best to distinguish between latency and intermittency, with the understanding that and how they are related.

Similar to the above, the ability to handle high latency should not imply that low latency applications cannot make use of the architecture. We so desire high latency tolerance and low latency utilization as properties.

Reliability (Access): A related metric in the access problem space is reliability: this refers to the probability that a given communications unit (packet, etc.) passes from one node to another successfully.

For the purposes of this document, it is not relevant how precisely this metric is defined, e.g. by a packet loss percentage or otherwise. What is necessary is to establish that reliability varies from deployment to deployment, and the effects it can have on access.

It is also worth noting that path reliability generally is bounded by the reliability of path segments -- that is, the least reliable segment defines how reliable the path is.

Finally, different applications have different needs for reliability. While it makes little sense for an application to require low reliability, it can certainly be tolerant of it. The properties we desire for a system then should be high reliability and inconsequential reliablity.

Throughput: Similar to the reliability metric, throughput can vary situationally. And like reliability, throughput is effectively bounded by the lowest throughput path segment.

The range of throughput difference is vast, and it is best understood as the maximum transmission unit (MTU) times the throughput rate . This definition is deliberately separated from reliability, though in practice, the two can have a strong interaction. Effectively, this describes the theoretical throughput of an unobstructed link.

Both components of throughput are bounded by physics and signaling decisions provided by the link layer.

Just as with the reliability metric, in terms of desired properties, a system should offer high throughput as well as inconsequential throughput.

The above list may not be exhaustive -- the purpose is to list characteristics of communications links, the access to which can be derived from either physical or societal factors.

They are here expressed as pairs of opposing desired properties, because depending on the use case, the system should behave closer to one end of the performance spectrum than in other cases. It would be easy to argue that only the more demanding of the two properties is of

interest, but it is optional. Unfortunately this can imply that the opposing end of the spectrum is of little interest -- by providing pairs of opposing properties, we can instead observe whether an architecture caters to both (albeit not simultaneously).

4.2.3. HRPC

In the IETF's sister organization IRTF, the Human Rights Policy Considerations group is drafting [I-D.draft-irtf-hrpc-guidelines-20] as a guideline document for considering aspects of protocol design that may impact human rights negatively. All considerations that are listed in [I-D.draft-irtf-hrpc-guidelines-20], Section 4 can reasonably be translated into desirable properties, in that a novel system should e.g. have the property that a user's anonymity can be guaranteed, etc.

That document does a better job at weighing these considerations against each other, and many are already reflected in the properties listed above. That said, it's worth highlighting in particular the following considerations as desrirable properties (though not to the exclusion to any of the others).

Integrity: The integrity property refers to the system mantaining, assuring and verifying the integrity of the payload data.

Authenticity: The property of authenticity means that the system ensures the data comes from the source it claims to come from.

Confidentiality: This property does not merely refer to cryptographic confidentiality , but extends the meaning to include mechanisms and controls that prevent data from being shared without consent.

Security: The draft defines the security property to refer to the collection of considerations of [BCP72].

Privacy: The privacy property refers to having considered carefully the items listed in [RFC6973], Section 7.

Anonymity: The draft defines anonymity has providing no indication of the user's identity, not even through statistical analysis.

Pseudonymity: This property refers to the use of identifiers in such a way that they cannot be related to a user's real-life identity.

Unlinkability: The unlinkability property relates to re-use of pseudonymous identifiers, i.e. that they should not be re-used in a way that permits inferring data about a user's identity.

Censorship resistance: This property means that a protocol should not include choke points at which censorship can be enacted .

Accessibility: The draft refers to accessibility as ensuring the protocol provides an enabling environment for all.

It is well worth highlighting that the issues described above (Section 2.1) all relate to concerns listed in the HRPC document and versa. We focus here on the above subset of criteria mostly because they can be more easily discussed in terms of architecture; it should not be assumed that the remainder are irrelevant.

4.2.4. Miscellaneous

In addition to the well defined properties above, today's world is one with many more connected devices than existed when the Internet or Web were conceived. Additionally, a growing number of those devies are no longer stationary, but quite mobile.

Numerous attempts have been made to provide connectivity for mobile devices at distinct layers; this is sufficient effor that we should define additional desired properties.

Mobility: Describes the ability to react to changes in network attachment, and the desire to maintain connectivity throughout this change.

Multi-Homing: Describes the ability of devices to maintain multiple simultaneous network attachment points, and the desire to connect to resources on the device indepedent of how many or which such attachments exist at any given time.

It may be worth highlighting that in the context of the Internet as a network of networks, it is easy to treat network attachment as attachment to the entirety of the Internet -- but that view stands in conflict with the intermittency related properties. Rather, one should visualize attachment to *a* network which may or may not provide connectivity to another endpoint, based on the current intermittency conditions.

Examining the issues further, it must also be noted that there is an issue of centralization (Section 2.1.3) -- which is strictly speaking not an issue in and of itself, but rather enables other problems. To borrow wording from this gap analysis framework, centralization induces, or contributes to induce, undesirable effects.

The way it contributes is by tightly coupling functionality of the system to a single conceptual location -- this does not have to be a single machine, nor a single system, but could also be a number of systems under control of a single entity.

Perhaps it is best to borrow a political term as the property that best describes an opposing principle:

Self-Determination: This property describes the ability of any element of the system to freely select which other element(s) it cooperates with and for what purpose.

It is clear that a single, isolated element does not make a distributed system, or at least a very poor one. Being tied by design to some other elements means that the system is vulnerable to censorship, to intermittency issues, and to reliability concerns.

4.3. Gap Analysis

This part of the document provides an analysis of several existing architectures, starting with the Internet and the Web, which are the primary focus of the problem analysis in Section 2.1.

Additionally, ICN and DTN are examined as examples of more experimental and/or special purpose architectures.

4.3.1. Internet

The first candidate to examine is the Internet itself. It's a building block for the Web (Section 4.3.2), so cannot be expected to have all the desired properties -- but it is well worth noting which it fulfils. When we're discussing the Internet in this context, we'll consider this to mean the Internet Protocol (IP, [RFC791], [STD86]), with the two most common and oldest protocols, the Transmission Control Protocol (TCP, [RFC9293]) and the User Datagram Protocol (UDP, [RFC768]).

The Internet has most of the properties defined for REST, with some meriting closer examination.

Extensibility: It's worth noting that IP version 4 is not particularly extensible, but that version 6 contains provisions for so-called extension headesr, by which implementations can add additional functionality to the system.

Moreover, also version 4 has evolved significantly since its inception. But the protocol was not designed for a lot of flexibility -- rather, the evolution was more focused on implementations than specification. TCP, for example, has seen a number of different congestion control algorithms which are entirely implementation specific.

Customizability: The design of the Internet protocol suite is such that clients do not dictate how servers behave. Rather, its fundamental principle is the end-to-end principle, meaning that both endpoints in a communication negotiate behaviour. In order to remain compatible with other endpoints, such negotiation is largely optional, and endpoints are encouraged to continue functioning in the absence of a sucessful negotiation.

Visibility: All protocol information is visible in Internet packets; this has led to the creation of middleboxes that may monitor and control the flow of data. Critics argue that this violates the end-to-end principle, and has led to more problems than benefits.

It is not the purpose of this document to settle the debate. However, where there exists a debate about fundamental principles, it can be argued that the architectural properties are either not well defined, chosen or enforced.

Portability: Portability in the REST sense does not apply.

Reliability: Reliability in the REST sense, as in resilience to failure of components is a fundamental property of routing in packet switching networks.

User-Perceived Performance: Considering the lack of reliability in IP and UDP, the notion that there is a user perceived time to completion of a request at all does not exist on the Internet.

Let's examine the access criteria.

Intermittency: The Internet is *only* designed for situations of low intermittency. Its ability to route around failures (see reliability above) can mitigate some intermittency situations, but not all.

Latency: Likewise, the Internet is designed for low latency only. In principle, latency can vary in IP and UDP. But the fundamental design is one where endpoints communicate with each other as fast as they can. The design of TCP reflects this, which treats a lack of acknowledgement of receipt of a packet within a short time window as failure.

Reliability: TCP's notion of reliability is that when several attempts to resend a packet have failed to produce acknowledgements of receipt, the entire connection is to be terminated.

This speaks less of a design *for* reliability, than it is an expression of an assumption *of* reliability of the underlying data links -- which is arguably the opposite notion.

This also means that this notion of reliability will always be at odds with the desired access property of high reliability.

Throughput: The Internet does not particularly care about throughput, and functions both in high and low throughput situations. That said, TCP again introduces its own limits in that it produces keep-alive packets when no user data is being sent for a while, so adding a notion of a minimum throughput rate.

In terms of HRPC's criteria, the Internet protocol suite meets very few of the desired properties. Integrity of data packets is produced via checksums, but without extensions such as IPsec ([RFC4301], [RFC4309]), none of the other properties are provided.

It's worth discussing the use of IP addresses as identifiers in this context. IP addresses provide little in the way of linkability to a specific user, and so are pseudonymous. That being said, the hierarchical structure of IP addresses and its mapping to physical networks permits deduction where in the world an endpoint resides.

Additionally, every application re-uses the same identifier. That means there is strong linkability between different concerns, which again implies the ability to analyze traffic and gather more information about potential person identifiers than immediately apparent.

This is so bad in practice, that IP addresses are considered PII under GDPR (Section 3.2.4.1).

Note also that the hierarchical structure of IP addressing and the related routing over physical infrastructure provide easy means for broad censorship of entire network segments (Section 2.1.6).

IP is also not well prepared for the mobility and multi-homing properties described above, though it is worth noting that Multipath TCP ([rfc8684]) serves to address both in principle.

The preceding paragraphs should show that while the Internet has many of the desired properties, it falls short in areas of human rights protections as well as access.

It may also be worth drawing attention to QUIC ([RFC9000]) briefly, because of the amount of work that has been put into making it a generic transport. QUIC does cater to some of the HRPC criteria. But in it's wortst case behaviour relating to reliability, it actively imitiates TCP's resend mechanisms. As such, it cannot be seen as providing signficiantly more of the desired properties than for example IPSec and TCP might.

Nonetheless, it also goes to demonstrate that the underlying IP protocol is flexible enough and provides enough primary properties that layering additional protocols over it may then yield all of the desired properties.

4.3.2. Web

Given that REST and the Web enjoyed concurrent development, it is unsurprising that the modern Web still has the properties described in the REST paper, so there will be no need to further discuss those here.

Access properties are also based on what the underlying Internet provides, and so make much the same assumptions on the reliability, latency, intermittency and throughput of lower layer links.

Regarding mobility and multi-homing, the Web fares a bit better than the Internet, simply because it deals in names rather than addresses. By requiring a resolution from Domain Name System (DNS, [RFC1034] and its many extensions) names to IP addresses, Web servers can in principle be mobile and multi-homed. As long as the DNS entries are kept up-to-date, and the client queries DNS often enough, it is possible to transparently move servers around.

On the client side, things are much the same -- the server (usually) does not have to care about the client's network attachment. But note that these effects last only for the duration of a request-response pair. Should network attachment change between the time a request is initiated by the client, and a response is received by it completely, the transmission will fail.

REST additionally intends for server implementations to be *stateless*, which means that every client request needs to contain every piece of information required by the server to process the request. Unfortunately in practice, the introduction of user sessions is commonplace, while the transfer of session state between servers is not uniformly well implemented. As a result, the Web is far less able to address mobility issues as REST would require in principle.

We can also update our understanding of the Web and assume that the modern Web will always utilize QUIC as its transport -- this is, at least, the current state of the art, whether or not that has caught on everywhere yet.

QUIC mandates the use of DTLS ([RFC9147]), the Datagram version of TLS. This provides transport encryption, and so integrity and some confidentiality. Other properties in the HRPC section above are met in much the same way as the Internet meets them, namely not very much.

If the situation is as bad as described above, it begs the question why the Web has enjoyed the success it has. The answer here is that plenty of solutions exist to extend the Web to provide more of the desired properties. For example, the use of OAuth ([RFC6749], [RFC7650]) can help mitigate issues in server mobility, by providing a means by which authorization information can be (partially) forwarded by clients to servers -- which means systems can be built that better conform to the REST principles.

However, the benefits this technology provides in practice is highly dependent of the overall design of the Web-based system. Additionally, this is not part of Web technology per se, but an optional addition. So while it can be said that it's possible to build a system using the Web that has more than the desired REST properties, the Web itself does not -- simply because such properties are not ubiquitously deployed.

This neatly brings us to the desired HRPC properties. Technologies such as OAuth not only functionally extend the Web, they also address an overlapping problem domain to the Web proper.

At the architectural level, REST is largely concerned with connecting components such as clients and servers. It introduces a notion of a *resource* as a major element in its design, which IP does not know much about. And then it glibly describes REST in terms of users accessing resources (using user agents connected to servers), without much consideration about what "resource access" may entail.

Indeed, also Web protocols define largely how a user may authenticate with a server, but the "resoure access" problem must include notions of authorization as well. By neglecting to address these at all, it is impossible for the Web proper to be considered complete without the addition of OAuth or similar tech.

This point can be made more clearly when one considers that the Web effectively models user-to-resource interaction (while the Internet is focused on endpoint-to-endpoint interaction). User-to-user interaction is not addressed on the Web at all, yet that is what we often use it for.

While implementations differ, the most basic approach to this is to have multiple users interacting with a common resource, which provides functionality for relaying between one user and another, and so implement user-to-user interaction.

In order to address the HRPC properties in such a system, authorization is a fundamental component, and the Web's lack of ubiquitous provision for it provides for many pain points.

More importantly, however, requiring authorization makes it very difficult to provide anonymity, and it becomes harder to manage this whilst providing unlinkability. In addition, REST's visibility property applies to user-to-resource interaction -- and when layering user-to-user interaction on top of this, it cannot provide confidentiality. Lastly, layering user-to-user interaction over user-to-resource interaction provides an anchor point for censorship in resource.

In relation to this, the property of self-determination is particularly undermined by the compbination of the REST properties of portability and visibility, albeit somewhat indirectly. One of the issues with the Web is that it describes quite clearly how to retrieve, create and delete resources (via the GET, PUT and DELETE methods respectively). In each case, it is assumed that the resource is manipulated in its entirety.

There is, however, no generic update mechanism. Rather, the POST method's request and response bodies are explicitly left undefined, in order to provide the most appropriate application-defined means by which to modify resources. The HTTP specifications provide one means to frame a POST request, as multipart/form-data. But they then permit the application to send code to the client (portability), which can create an appropriate POST body. This must necessarily imply that the application has visibility into that payload. But by making the client dependent on a particular piece of code, its functioning is not self-determined -- but rather entirely coupled to what code the server sends.

As a side effect, this enforced visibility also means that REST is architecturally incapable of meeting all of the the HRPC properties, adding to the previous incompabilities. Again, as above, this does not imply that applications cannot be built on the Web that also provide those properties. But the Web does not and cannot provide them by itself.

4.3.3. Information-Centric Networking

Information-Centric Networking (ICN) describes a class of solutions derived from peer-to-peer (P2P) networking, with the most prominent current examples being Named Data Networking ([NDN]) and Content Centric Networking ([CCNx]).

P2P networks have been created for many purposes, but the most infamous one is for sharing files. ICN imagines the internet as being primarily built around this usage, and presents answers to how to make content directly addressable, and most importantly, how to make content *routable*.

Superficially, this is not significantly different from a file sharing P2P network. The main differences lie less in what functionality is presented to the user, as in how fundamentally different the implementation details are.

For this, ICN re-imagines the layer model of the Internet as defined by [RFC791]. On the Internet protocol stack, there is a central funnel of IP packets. Any layers below the IP layer are freely exchangeable, and IP continues to function. As an implication of this, any layer above the IP layer can be chosen to the application's best intersts, and this similarly has no impact on IP.

In ICN, this central pivot point becomes content chunks, each of which have their own address. ICN protocols are then concerned with addressing and retrieving these chunks with so-called "interest" packets, and retrive the chunks in "data" packets.

As a result of this change, the current location -- or locations, in fact -- of a content chunk is no longer particularly relevant. Location is a concern of lower layers in this adjusted model, while higher layers only deal in content identifiers.

This leads to a relatively significant shift in properties for ICN. Most notably, it easily provides for mobility and multi-homing properties, can help with intermittency, and provide for some censorship resistance. It also enables self-determination, because the content layer simply no longer cares about the specifics of how the system is connected.

By itself, ICN does not provide any of the other HRPC properties, however, and induces overhead that stands in contrast to the desired properties of network performance, latency, reliability (in the access sense), and possibly throughput.

The reason for this is fairly simple: in order to safely connect a content chunk with a content identifier and vice versa, the content identifier is chosen to be a cryptographic hash of the content it identifies. The properties of such functions imply that there is no reasonably way to determine from the hash identifier whether the content chunk they represent is similar or related to any other content chunk.

This has two immediate effects: one is that hierarchical routing such as used in the IP address spaces is impossible to achieve; there simply is no structure to the identifiers that can be exploited for this. More precisely, any similarity in e.g. shared identifier prefixes is entirely unrelated to the content's purpose.

As a result of this, it is not possible for routers to predict or optimize for related content. Content chunks belonging to the same resoure may be routed along wholly different paths with distinct performance characteristics, which means that any notion of end-to-end quality of service can be discarded from the outset.

The other issue relates to how routing is implemented in ICN. Because there is no structural information to addresses, each router is left with little choice but to treat every content address as a distinct route. The design is for routers to store every interest packet and forward it to the next hop, and discard it only when the responding data packet is returned (ignoring any optimizations here).

Current discussions on Internet routing tend to focus around how much optimization of common prefixes is good for reducing the overall volume of routing information required, vs. its negative effects in also reducing routing accuracy. The discussion occurs because without such route bundling, the current volume of routing information is no longer tractable. In ICN, routing information is orders of magnitude larger over the scale of the entire Internet, so that a similarly sized deployment is infeasible with current router capacities.

This discussion requires two caveats to be made. One is that the usage of content hashes as identifiers is largely a CCNx choice, while NDN uses application chosen "names" instead. This does alleviate both issues to the extent that prefix-based routing is feasible again, and so is relating content by a shared prefix. However, the names are intended to be opaque to the network, and have meaning only to the application -- which makes such optimizations difficult to do perform accross all applications.

This aside, the retrieval model of ICN is similar to that of REST, in that it relies on clients requesting content (i.e. a largely unidirectional PULL model). By contrast, the Internet's IP model is inherently a bidirectional model, in which there is no clear distinction between requests/

interests and response/data. The only point where endpoints' roles are distinct in whether an endpoint listens to incoming connections or initiates an outgoing one (and that assumes the use of TCP and the notion of "connections" in the first place, which is not the only use of IP).

As a consequence, ICN is not well set up for many of IP's use cases, where bidirectional communication is required. Recent developments such as reflexive forwarding ([I-D.draft-oran-icnrg-reflexive-forwarding-06]) seek to address this, but do so by mirroring routes, and so the already existing pressure on the routing information volume increases. (The actual details are more complex and include some path aware routing elements, which are not necessary to discuss here.)

In summary, ICN introduces a meaningful shift away from packets as the thin waist of an hourglass stack towards data -- which helps with some desired properties, but is less conducive to others. However, HRPC properties remain largely as little addressed as in previously examined archiectures, with the notable exception that addressing by content hash at least provides for some integrity and/or authenticity.

4.3.4. Delay Tolerant Networking

Delay Tolerant Networking is an effort to directly address some of the access properties listed above, though the problem domain is space communications (Section 3.2.2.3) rather than more earth bound access issues. The Bundle Protocol (BP, [RFC9171]) provides a specification for a DTN protocol, though the generalized architecture is provided in [RFC4838].

The approach chosen here is again a conceptual shift away from using IP packets as the focal point of the architecture, towards so-called data bundles. Much like content blocks in ICN, they can contain arbitrary application data. Also similar to ICN, BP is mostly concerned with handling bundles, and does not care about which specific lower layer "convergence protocol" transports bundles.

However, BP (and by extension, bundles) do differ strongly from ICN content in that they are conceptually messages addressed to a particular endpoint in the system. As a consequence, bundle metadata is mostly concerned with specifying the destination endpoint (in a similar way as e.g. email does), as well as processing flags.

The fundamental function of BP is to ensure custody transfer. At some level, this is comparable to TCP's acknowledgement of receipt of a packet. But unlike in TCP, transfer of custody means that it is now no longer the concern of the node in which the bundle originated to ensure the application data arrives at its destination. Rather when a BP node takes custody, the originating endpoint effectively shifts away from that node to the node taking custody.

In this way, custody transfer enables high reliability and tolerance for high intermittency. Bundles can also be arbitrarily large, because the problem in space communications revolve around latency and intermittency -- bandwidth is not (much of) a problem. In that sense, BP also induces high throughput.

Given the discussion of ICN as primarily a unidirectional, PULL-based approach to networking, it is well worth highlighting that DTN/BP is fundamentally bi-directional in nature, in much the same way as TCP/IP is.

Its properties are thus very much comparable to that of TCP/IP, albeit with the above additional access properties. Unfortunately a side effect of BP's design space, the relative lack of bandwidth constraints, is that is not particuarly optimized for situations in which access is bounded by the throughput of the underlying convergence protocol and lower layers.

The approach of transferring custody also implies that there is little in the way of fallback mechanisms when an intermediate node that has taken custody fails. Unlike TCP, which remains fully end-to-end oriented, shifting the endpoint away from the originating node also means that such failures are difficult to detect and recover from in the BP protocol layer.

This is partially addressed by efforts such as [I-D.draft-ietf-dtn-bibect-03], which adds flexible forwarding mechanisms for bundles that include duplicating bundle transfer along multiple indepdendent paths, and de-duplicating them at some node further along (which may or may not be the designated endpoint).

Similarly, HRPC properties are not immediately addressed in this approach. Additions such as Bundle Protocol Security (BPSec, [RFC9172]) do provide these means, however, and modifications to BP v7 over the prior v6 version include provisions for supporting BPSec.

4.3.5. Summary

None of the architectures examined in the previous sections provide all of the properties derived from the problem section. In each case, additional layers can be found to address HRPC considerations better, and in some cases these do exist, ready for deployment.

Each architecture additionally makes choices according to their use cases which in some cases hinder providing HRPC or access properties, which makes them less than ideally suited for addressing the issues outlined in the problem section.

The architectures are combined in one specific flaw: a curious absence of the end user in their design. Anything relating to user identification or or authorization is largely left undefined (with various nods to such problems scattered through the respective specifications).

This apparently derives from a perspective that these systems are either concerned with machine-to-machine interactions, and may have evolved that towards machine-to-resource interactions. Clearly, as designers of computer networks, these must be prominent concerns.

But should this not equally as clearly occur in the service of user-to-user, truly human centric networking?

5. Architecture

This part is currently missing or needs revision.

- drop visibility, it has too many issues
- discuss portability in the sense that data may be executable, but we'll still define how to access data.

5.1. Elements

In order for a novel architecture that shows all of the above properties, it must be designed in a human centric fashion. But it still needs to be acknowledged that computers' primary function is to provide resources to humans, which can be anything from data files via services to representations of other humans.

We can thus model a new architecture around the expected human-to-resource interactions. REST, ICN and DTN all contain the notion of resources, albeit not necessarily in that form. IP's abstraction is that of an interface to which an IP address is bound -- but looking a little higher in the stack, we can find that both TCP and UDP define services as the main abstraction via the assignment of services to ports.

Architecture	Name	Access method
Internet	Service/Port	Bi-directional packet exchange
REST	Resource	Request/response
ICN	Content, Data	Interest/data (similar to REST)
DTN	Endpoint	Bi-directional message exchange

Table 1: Resource Equivalents in the examined architectures

We note that these methods fall into two categories: the bi-directional exchange category is very general purpose; it does not define in which direction there is more data flow between endpoints. It is similarly not too well defined which endpoint initiates contact.

By contrast, the more request/response oriented methods clearly, if perhaps implicitly, define the role of a resource consumer who initiates a transmission, an a resource holder, who responds. In fact, resource creation is left largely undefined in ICN, and left incomplete in REST. The main advantage of this method is that it permits, at least in principle, removing the resource location from the equation, which induces increased network efficiency due to caching, etc.

Noteworthy is also that the request/response method maps fairly neatly to the document web use case (Section 3.1), while the bi-directional exchange maps well to the real-time use case. The API use case can be fulfilled by either mode; many APIs follow a request/response pattern, but callback or notification patterns require bi-directionality.

The clear implication is that a new abstraction must provide both categories of access method. This abstraction must also be the central pivot point for the architecture, tightly coupled to the user interactions it affords.

5.1.1. Users/Humans

Users must be represented in the system somehow, which means they need to be identified. Identification can rub up against HRPC properties, so it should be highlighted that identifiers under this architecture must be pseudonymous.

It must additionally be possible to create many ephemeral identifiers, in order to provide for unlinkability. If ephemeral identifiers are indistinguishable from other identifiers, this enables a quasi anonymous mode of interaction, as a random identifier reveals as little identifiable information as an absence of an identifier, provided that they remain unlinkable.

The HRPC guidelines acknowledge that some rights are difficult to meet when others are fully embraced. For example, the right to remedy can be difficult to enforce when full anonymity is given to users in a system.

The solution we choose to this dilemma is to leave it to the specific application how much a user identifier can be traced to a person's real world identity. Using a lot of different ephemeral identifiers will provide more in the way of anonymity, though at the expense of remedy -- and conversely, permanent identifiers would provide zero anonymity and unlinkability, but help with full remedy.

In order to permit both, however, user identifiers must NOT contain, and so leak any information to an observer that would indicate the level of permanence the identifier enjoys. Instead the system must treat all identifiers equally, and must presume they are fully ephemeral, and exist solely for the duration of (a part of) some session.

Furthermore, identifiers must be related to some asymmetric cryptographic key pair. A typical such relationship would be for the identifier to either be a public key (such as e.g. in elliptic curves of [RFC8410], [RFC8032]), or be a key fingerprint, i.e. a hash over some public key for RSA or DSA keys ([NIST.FIPS.186-4]). Establishing this relationship means that it can be verified that e.g. a cryptographic signature is issued by the entity identified via a matching identifier, which in turn induces or helps incude the remainder of the HRPC properties.

5.1.1.1. Creators

Creators of a resource (Section 5.1.2) are those identities that generate a resource identifier, and store it along with other data in an initial resource chunk (origin).

Creators also own the resource. While many other participants can contribute (Section 5.1.1.3), the creator determines whether such contributions are authorized by adding other contributors to the resource.

5.1.1.2. Consumers

Consumers of a resource request and process a resource.

5.1.1.3. Contributors

Contributors to a resource are verifiable authors of a chunk of a resource, i.e. they have provided some signature or other means of verification that this resource chunk is authored by them.

The creator of a resource is a contributor, at least of the origin. Other contributors are not creators.

All contributors to a resource are also consumers of that resource.

5.1.2. Resources

We define a resource as an element in the network architecture that provides some utility to humans interacting with the system. Just as in REST, "any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on."

A resource is distinct from an ICN content chunk, because it is not bounded to any particular size (as e.g. chunks may be). If the resource is data, this implies that the data can grow and mutate over the lifetime of the resource. This induces REST's modifiability properties.

A resource is distinct from an IP service and from a REST resource, because it is location independent. A resource is distinct from an IP service also because it may persist beyond the lifetime of a service port. This induces the mobility and multi-homing properties, and both of the reliability properties. It also helps with some performance properties.

A resource is distinct from a DTN endpoint because it does not only name a destination for messaging, but may have other meanings (see above). This induces other modifiability properties, but may also include also portability.

A resource, in other words, is a data stream that may be transmitted efficiently, and has a purpose -- and this purpose must be embedded into the resource, or not all of the above properties can be fulfilled.

In order to retain these properties, a resource MUST also be self-contained. The moment additional metadata needs to be managed for a resource, such as a manifest, or ownership information, etc. the architecture introduces a dependency on a different element which can fail or be unreachable -- and so undermine the reliability and intermittency related properties in the worst case.

This also implies that a resource needs to be verifiable in itself, and so requires a method such as cryptographic signatures to be added.

For public resources, this is sufficient. But for modelling user-to-user interactions, not all resources can reasonably be public. Therefore, end-to-end content encryption also must be supported.

5.1.3. Nodes & Convergence

In order to provide mobility, multi-homing, improve reliability and deal with intermittency to varying degrees, we consider communications to occur between nodes, not between any particular network interfaces. We borrow a page from the DTN approach and indeed treat the architecture as able to function with arbitrary convergence layer protocols that transport the protocol messages we will discuss below.

In fact, it is likely that each node in the network should be addressable via multiple convergence protocols simultaneously. If the convergence protocol is e.g. IP based, this directly enables mobility, as node addressing can remain static even as attachment points to the IP network change.

One consequence of this approach is that nodes require addresses independent of the convergence layer. There exist a many-to-many relationship between nodes and users -- a system may cater to multiple concurrent users, but at the same time, users may simultaneously use multiple devices, each with their own network attachment points.

For this to work, each convergence layer must provide more than a transport means, but rather also provide a means for managing the mapping of a node address to the convergence layer's own addressing scheme. Unlike in DTN, this architecture assumes that such mapping is highly dynamic (DTN is relatively agnostic here). The specification of such mechanisms is outside of this document's scope; however, in principle any mechanism such as the Domain Name System ([RFC1035] and its many extensions and updates) or more novel approaches such as Routing On Service Addresses (ROSA, [I-D.draft-trossen-rtgwg-rosa-arch]) can be used.

This architecture, however, defines more requirements on the convergence layer below.

5.1.3.1. Custodians vs. Caches

Any node that has custody of (parts of) a resource is a custodian of that part.

Note that this definition excludes nodes that merely cache a resource. In contrast to a cache, a custodian is charged with continued storage of the (parts of the) resource they manage. Caches merely provide best effort storage.

5.2. Interactions

Defining the elements of this architecture is the easier task. There is a strong distinction made between the human as represented by a user identity, and the network node. Another distinction is made between the node and its convergence layer protocols. Finally, a relationship -- in the abstract -- between users and resources in the form of cryptographic signatures is established.

We can now elaborate the interactions users have with resources, and examine how this may related to network nodes.

We have already explored how ICN and REST vs. the Internet and DTN model different access patterns. In particular, one group is more heavily focused on resource consumption, while the other on bi-directional messaging.

We have to observe that neither fully describes the user-to-user interactions we see either on the Internet, or in fact in the physical world. While it is common enough for people to have personal conversations with other folk, a lot of our time we spend speaking to and collaborating in groups.

Rather than treat groups as a special case of interaction, one should consider that a pair of people are still a group, albeit a very small one. In fact, one-to-one communications should be considered the special case, while group communcations must be the general case, if we are to model human needs well in the digital realm.

Within group communications, we can identify the speaker role and the listener role. It is by no means given that there is only one speaker, and only one listener at any given time. In fact, the roles constantly shift back and forth (some people claim they can speak and listen simultaneously, but the data on that is apocryphal at best).

Neither are groups static; members constantly get added or leave. Finally, individuals can be in multiple groups simultaneously.

With so much in flux, this makes it hard to pinpoint exactly how to define a group. The response, in most any digital system, is that groups are "things" that can be created, and that provide affordance for the management and self- management of its members, and relay messages between members.

Now that we have this group "thing" described, should we add a new element to the architecture?

It turns out, that is not necessary. We can simply define a group as the set of users currently concerned with a particular resource.

This has two major implications.

The first, and simpler one is that the convergence layer protocols really must provide group communications, and their respective method for mapping node identifiers to convergence layer addresses is really an exercise in group membership management.

One obvious way in which this can be provided is via IP multicast, e.g. by mapping a resource identifier to a multicast address in some way. But as we will see later, this approach could be a little too naive.

The second implication is because resources must be self-contained: this means also that group membership -- at the level of user identifiers, not at the level of node identifiers -- must be contained within the resource, and by extension such related information as permissions.

It must be, because if it were not, then we would again introduce a dependency of the resource on other elements (nodes, extra metadata) which provides challenges for fulfilling the desired architectural properties.

5.2.1. Resource Creation

Resources, like communication groups, must be created. In the process of creation, a creator user provides an identifier for the resource, as well as information on the intended usage of the resource.

If we wish to maintain the property of customizability, then the end-to-end principle must be redefined. We no longer treat this as relating to network endpoints (nodes), but rather to the principal elements of users and resources.

Specifically, we establish two separate processes: the first is a user-to-resource process, in which the creator of a resource documents their intent. This is not fundamentally different from e.g. choosing a TCP address and port. The choice of TCP as a protocol documents a streaming intent, and the port typically correlates to a service protocol which defines how higher layer interactions are to occur.

The key difference is that in this architecture, this intent is not an ephemeral state of a single machine, but rather a permanent feature of the resource itself.

This creation cannot be advertised to the group, because prior to the existence of the resource, no resource specific group can exist. We will get back to this later.

5.2.2. Resource Consumption

Resource consumption works quite similar to how it does in ICN: a consumer user posts an interest in a resource to its neighbouring nodes.

The interest specifies not only the resource identifier, but also the user identifier that expresses an interest. It may furthermore provide one or more node identifiers as routing information, and could even contain current convergence layer addresses for these nodes.

Recipients of an interest have have no obligation to store this interest as in ICN. They can respond in one of several ways.

1. If they have custodianship of the resource, they can decide whether to add the interested user to the resource group or not.
2. If they know or suspect nodes that may have custodianship, they can forward the interest to that node. If so, they should add the originating node identifier to the interest (if none is present), as well as the convergence layer address by which the interest was received.
3. If they are neither custodians nor can locate a custodian of the resource, they should return an error response to the interested consumer.

Adding a user to the resource group is a multi-step process.

1. First, custodians needs to check whether the user can join the group as a consumer.
2. If that is permitted, the resource data itself is updated to record that the user is now part of the resource group (this step may be omitted for public resources). This may mean distributing additional resource data within the resource group.

3. The custodian now signs the interest, and repeats it to the resource group. Depending on the convergence layer, this can result in several communication packets being sent along different channels.

Whether or not the custodian does permit this joining of a user to a resource or resource group can depend on many different factors. The custodian may be the node that currently hosts the creator of the resource; in this case, the creator can be explicitly asked to consent to this request.

Or the creator can have already added the user identifier to the resource; in that case, only the convergence layer operations for joining the resource group must be performed.

It's also worth highlighting that in principle it is possible to deterministically map a resource identifier to an IP multicast address, e.g. via an ORCHID v2 ([RFC7343]) or similar process. In that case, the interest can be posted directly to the resource group, and no routing of the interest to a more likely destination has to occur.

The key point here is not that all of the above steps have to be performed in precisely this order -- but rather that at the end of a successful initiation of consumption, the user identifier is recorded in the resource, and a matching node identifier has joined the convergence layer(s) groups.

In order to satisfy all of the use cases outlined above, an additional thing needs to happen: just like the resource creator needs to record intended usage into the resource, a consumption interest needs to specify desired usage. This can provide the second process, the resource-to-user negotation whether this desired usage matches the creator's intent.

This provides more information to the custodian node to decide whether joining the resource group is feasible. For example, when the consumer desires some bi-directional communications, and the creator just names some static data, that effectively represents a failed user-to-user negotation of the communications parameters.

Interests must also contain one or more of the following pieces of information:

- On the one hand, it is likely that a resource *is* in some way chunked up for better transport, even if it must be self-contained. An interest could be expressed for a (set of) particular content chunk(s) for the resource.
- Alternatively, the interest could be in the entire resource, which implies a subscription to updates to the resource -- either from the origin chunk, or from some chunk position specified in the interest.

At any rate, and unlike ICN, either of the above means that an interest can yield more than a single response. For this reason, nodes sending an interest must choose an identifier -- a cookie -- for the interest, which responses must contain. In this way, a single interest can be responded to multiple times.

For this to be manageable, interests must also contain a time stamp until which the interest is valid. Custodians must not respond to a timed out interest. Note that timeout of an interest, does not automatically imply timing out of group membership.

5.2.3. Data

Caches of a resource must ignore unsigned interests. Interests signed by a custodian of the resource must be responded to by sending data according to the interest, if the cache contains such data. Data responses must contain the interest cookie.

Care must be taken how to send data responses. While it is safe to assume that all members of a resource group share some interest in a resource, it is not a given that all the interests are equivalent.

Consider a video broadcast -- it is quite likely that when a user intends to join a broadcast, they wish to do so at the current time point. But perhaps they also wish to catch up with what has already been sent. Both are subscription interests, but they specify different starting content chunks (or perhaps none, in the case of the current time point).

To stay with the IP multicast example, it would not make sense to flood the multicast group with data some of the members would discard. For this convergence layer, it may be best to maintain multiple resource related groups -- one for sending interests to caches, perhaps, and one for each group of nodes that wish to consume the resource from (roughly) the same offset.

In other words, convergence layers require significant knowledge of the interest that data is sent in response to. Conversely, it is infeasible to suggest that data is sent to the entire resource group all the time.

Instead, data is sent to the convergence layer group(s) that this layer determines is best suited for the interest(s) at hand.

A few comments should be made on the distinction of custodians, caches, creators and contributors at this point.

Any node that hosts a creator or contributor acts as a cache, at least of the resource chunks that this user has created. Such nodes may also be full custodians. The key characteristic for the purposes of this section, however, is that they store some data, and can therefore send it in response to an interest, i.e. act as a cache.

5.2.4. Custodianship Provision

Where the previous sections have somewhat glibly assumed that the consumer and creator of a resource share some means to find each other, be they part of an IP multicast convergence layer group that can be derived from a resource identifier, or by some other means.

In order to provide the high intermittency tolerance property first and foremost, custodianship cannot merely lie in the node that happens to host the resource creator. At the same time, the mobility property requires that the network is not statically designed, but that nodes can be flexible in providing custodianship -- a static network design of custodians may not suffice here.

In order to find custodians, creators must send a custodianship request to neighbouring nodes. Again, what this notion of "neighbouring" entails is dependent on the convergence layer.

Nodes can respond in one or all of the following ways:

1. Generating a custodianship offer response to indicate that they wish to become custodians of the resource.
2. Forward a previously received offer of possible custodianship from another node.
3. Forward the custodianship request to other nodes it knows.

The specifics of the custodianship request and response are outside of the scope of this architecture.

A physically highly reglemented network may provide custodianship only from nodes operating on a specific converge layer protocol address. A logically highly reglemented network may provide custodianship only from nodes who can prove they are designated custodians by providing a signature of that fact from some mutually recognized authority. Other networks may provide more flexible custodianship.

A custodian node has two tasks:

1. First, it must permanently store any parts of the resource it receives. It may decide that it can best serve its purpose by also becoming a consumer of the resource in general, so that it accumulates all of the resource eventually. Note that a request for custodianship may request this behaviour explicitly.
2. Second, it must act on behalf of the resource owner to the best of its ability. As such, it may decide to permit consumers or contributors to join the resource.

In order to perform this job, authorization information that can ultimately be traced back to the creator must be embedded into the resource. For example, a resource may contain a section that explicitly marks a user identifier as a contributor or consumer. Or it may record other custodian nodes in the resource. It may delegate the ability to name other custodians to a particular (set of) custodians.

When a creator or authorized custodian accepts an offer, a custodianship acceptance response is sent to the newly inaugurated custodian. The same or equivalent may be sent to the resource group, and/or stored in the resource itself.

Custodianship, unlike group membership, is not technically a property of the resource itself. However, as the resource is shared amongst all group members in some way, recording primary custodians in the resource may be a convenient choice.

5.2.5. Custodianship Offers

Custodianship offers are generally the response to a custodianship request. Since custodianship requests pertain to an identified resource, the offer should typically also contain the same resource identifier.

It is also possible for nodes to spontaneously send custodianship offers for unspecified resources, to indicate capacity. Receiving nodes must store these offers, and respond with them as described in Section 5.2.4.

Offers are not permanent; they must be equipped with a lifetime. After an offer expires, nodes should discard them.

Note that there is no particular requirement for nodes to keep offers for a specific duration, or to keep all offers it receives; nodes can apply local policies here, including flood and DDoS protection policies, etc.

The main purpose of spontaneous offers is to pre-populate offer tables in nodes so that finding a suitable custodian can be accelerated. An implementation which forwards custodianship requests is equally viable.

Note that the above formulation of custodianship requests and responses makes it compatible with a large variety of convergence layer mechanisms.

In a local area network, for example, nodes may periodically broadcast spontaneous custodianship offers on the data link layer. Conversely, the criteria for custodianship selection are just wide enough to also e.g. permit mapping this mechanism onto a distributed hash table such as described in [KADEMLIA].

5.2.6. Custodianship Removal

Removal of custodianship effectively demotes a custodian to a mere cache. It is primarily information that needs to be communicated to the resource group, and so could be embedded into the resource.

In order to prevent situations in which custodianship is repeatedly accepted and removed by competing parties, we define that custodianship can only be removed by the party that accepted it, or by any party higher in the authority chain.

To illustrate this, assume that creator A delegated selection of custodianship to contributors B and C. B selects a custodian node CN.

Now contributer C cannot remove CN as a custodian. Contributor B could, or creator A could, because A initially delegated custodianship selection.

5.2.7. Data Removal

Data removal in a distributed system is difficult to guarantee. The preferred mechanism e.g. in ICN is to provide end-to-end encrypted data only, and then lose the encryption key, making data unrecoverable, which is similar in effect.

This mechanism is sound enough, but suffers from a data race. If a decryption key has leaked before legitimate nodes forgot it, the data remains accessible. Worse, it only remains accessible to illegitimate nodes.

Implementations are strongly encouraged to find complementary means to ensure data deletion. Some are discussed below.

1. In ICN, where resource chunks are addressed via a hash of their content, they are effectively immutable as any mutation creates a new content hash identifier, and so a distinct chunk.

If resource chunks are mutable, on the other hand, zeroing out data is as valid a mutation as writing any other data, and so can help protecting plain text data (either in public resources, or the plain text prior to encryption).

2. Creators may send explicit deletion requests to caches and/or custodians.

A conforming implementation SHOULD provide as many of these complementary methods as feasibly to best provide data protection, but MUST provide at least the above three -- unless some future revision of this document obsoletes them.

5.3. Notes

A number of notes apply to this architecture which are not easily expressed either as elements or interactions.

5.3.1. Performance

Several of the desired properties can only be achieved by selecting an appropriate convergence layer protocol for the combination of the creator and consumer intents.

Assuming that the two parties wish to engange in a video call; the resource may then represent the call session. This is a high throughput, low latency scenario with bi-directional messaging. A choice of BP as a convergence layer protocol may not yield the desired results here, due to it's design of dealing primarily with high intermittency.

Conversely, a creator may create a live video stream, but a consumer may not care at all to watch it as it is being created. They may merely wish to record it for later consumption. Here, even though the creator's intent is similar to the above scenario, the consumer's intent relaxes the requirements and can make BP a more viable choice.

The key thing to stress is that it is the combination of the creator's intent (as embedded in the resource origin) and the consumer's intent that makes for the best choice of convergence layer protocol, and implementations MUST take this into consideration when choosing from available protocols.

Additionally, implementations MUST provide such convergence layer protocols as necessary to induce all of the desired properties in order to be considered a full implementation of this architecture.

5.3.2. Intermittency

The explicit custodianship mechanism described above is different from the one in DTN, in that it applies to storing and making available of a resource rather than to taking care of forwarding a message.

One implication is that the end-to-end principle is not violated by the contributor discarding a resource chunk after a custodian has received it.

The main distinction to DTN here is that in DTN, the sender of a message is still part of the communications flow. Moving the effective endpoint to a custodian which can then fail leaves little means for notifying the sender, and allowing it to find a contingency solution.

In this architecture, because resources are self-contained, once a contributor has transferred custody of a resource chunk, it is -- conceptually -- no longer involved in how the resource is being accessed; the end-to-end scenario is ended. When a consumer accesses a resource, a new end-to-end scenario is established.

That said, in order to achieve intermittency mitigation akin to DTN, custodians MUST explicitly acknowledge the receipt of all data. Such recepts should be made at the granularity of resource chunks, however, not at the data packet or message granularity of TCP vs. DTN.

5.3.3. Resources and Chunks

The above deliberately avoids being too detailed about how resources or their respective chunks may be identified. Interests can be in an entire resource, however, or in an individual chunk. It follows that these identifiers may occupy a shared namespace -- but no such requirement is imposed here.

It is worth emphasizing that the notion that a resource is subdivided into chunks is not necessarily given. A resource may consist soley of a single mutable chunk -- if so, then why distinguish between the chunk and resource?

Rather, the distinction is explicitly made so that implementations consider the ramifications of how resources should be represented.

One point to stress again is that resources MUST be self-contained. That is, information on which chunk(s) appear in the resource in which order must be embedded into the resource itself. Only then can we guarantee that no dependencies on additional architecture elements are introduced.

5.3.4. Multiple Convergence Layer Protocols

Each node may not only provide multiple convergence layer protocols, but may also use them simultaneously for a single resource. This implies the existence of a messaging abstraction in implementations whereby a node sends a message into a resource group. Each convergence layer can then forward the message according to its means.

If a receiving node receives the same message via multiple convergence layer protocols, it must discard duplicates of the message and process them only once.

If nodes A and B communicate via one convergence layer protocol, and nodes B and C via another, incompatible one, this does not pose a problem. What counts are that messages -- intents, data, etc. -- are forwarded, not that all nodes communicate in the same way.

5.3.5. Pivot Point

Due to the exchangable convergence layer protocols, we have a narrow waist in the architecture that is different from e.g. the Internet architecture, where the narrow waist is IP packets.

Downwards, towards the convergence layer, the narrow waist consists of the messages in a compatible protocol. Upwards, towards the user, the narrow waist consists of a self-contained, shared resource. The architecture does not place any constraints on the data embedded in the resource (with the exception of meta information discussed in this document).

It is likely necessary to acknowledge this dual layer narrow waist. That said, the leading abstraction is the self-contained resource. It is feasible that several competing messaging protocols exist that conform to this architecture.

5.3.6. Multiple Contributors

The notion that multiple parties can contribute to a single resource is unusual, and derives from group communications as the primary mode of communications. But it is also what makes this architecture effective at modelling real life user-to-user interactions.

This has some implications on how to model a self-contained resource, but this architecture should not be prescriptive of the means, only the effect.

In particular, it implies that updates to the resource should likely be structured in such a way that updates by multiple contributors do not conflict with each other. One set of methods for this are conflict-free replicated data types (an overview can e.g. be found in [CRDT]).

But just because the existence of multiple contributors is explicitly acknowledged and considered, this does not imply that every application of this architecture must in fact provide for multiple contributors. A single contributor/creator is equally supported. In such a scenario, a resource payload may also consist of a simple file of well-established type, etc.

5.3.7. Self-Contained Resources

This document stresses that resources must be self-contained, but it should hopefully be apparent at this point that this relates to not introducing dependencies on other users or nodes. It is perfectly fine for a resource to *refer* to other resources, e.g. in the same way that a hyperlink in an HTML document does.

On the other hand, it is equally possible for a resource to contain several multiplexed data streams, as is e.g. the case for most video file formats.

5.4. Analysis

With the elements, interactions and notes elaborated, we can now analyse whether this architecture induces all of the desired properties, and we must conclude that it does not.

This is, however, by design.

In particular, we must note that REST's visibility property is somewhat incompatible with the HRPC properties. In particular, while the architecture does provide for some theoretical visibility into aspects custodianship management, end-to-end encrypted resource payloads mean that the type of visibility that REST provides is not possible. But we must end-to-end encrypt in order to guarantee the HRPC properties.

We must, therefore, discard visibility as a desired property.

At this point, we can examine if and how the proposed architecture induces the remaining properties.

Network Performance: The architecture requires that creators and consumers advertise their intents, and that implementations choose convergence layers to satisfy these. That can must include high throughput, low overhead scenarios.

User-Perceived Performance: Similar to the above, the choice of convergence layer may satisfy this property. That said, the architecture explicitly encourages caching of data, as well as custodians. In this manner, nodes are free to select the lowest latency data cache available to them, which may be local.

Network Efficiency: See the use of caches above.

Scalability: Scalability is induced in multiple ways: - In much the same way as REST provides scalability by separating interactions around specific services, this architecture separates interactions around specific resources. - The use of custodianship management is chosen to make it optional (though necessary for inducing other properties). Additionally, it is designed so that not every node needs to be aware of every custodian and vice versa.

Simplicity: The amount of elements and interactions is deliberately kept low. Complication is introduced by the interdepencence of the messaging layer with the convergence layers, but this is necessary complication in order to induce other properties. All things considered, the architecture is simple enough.

Evolvability: The architecture establishes interaction patterns and some requirements on individual elements, but does not perscribe how these requirements are fulfilled. It thus provides evolvability of the system.

Extensibility: The architecture defines a minimum set of different roles and interactions, but does not limit extensions. Once group messaging is implemented, additional messages can be added to permit extensions, different groups from the one(s) described can be created, etc.

Customizability: The client does not so much initiate server behaviour as it negotiates its intent with the intent recorded in the resource. The server should chose to respond such that both intents are satisified.

Reusability: Components are re-usable in the same sense as REST. The caches provide a uniform interface, and could so also be implemented as proxies. This implies the inclusion of a user agent-like component, which may then communiacte with downstream caches.

Portability: Portability is not explicitly addressed in this architecture in the same way as in REST, but the property is nonetheless fulfilled. Consider that 1. A resource may also contain code, and 1. resources being self-contained can be moved anywhere to be processed, and 1. resources being shared in a communication group, can be procssed at each of the participating nodes.

Reliability (REST): As in REST, the architecture provides resilience against failure of components not involved in a particular resource's group. Unlike REST, the custodianship transfer mechanism provides additional resilience against failures of all but one custodian and all caches.

High Intermittency Tolerance: The custodianship transfer mechanism as well as the requirement that resources are self-contained provides for high intermittency tolerance.

Low Intermittency Utilization: The selection of convergence layer protocols based on consumer and creator intents permits for effective utilization of low intermittent connections.

High Latency Tolerance: Treating resources as self-contained and mutable effectively introduces a high tolerance for latency, in that a resource is always "complete". Whether the current node has all the parts that other nodes have does not affect this notion of completeness.

Low Latency Utilization: Where low latency convergence layer protocols are available, they can be utilized.

High Reliability (Access): The high access reliability property is provided by explicit custodianship management.

Inconsequential Reliability (Access): Custodianship management is an optional component of the system, which means it is also possible to treat reliability of access as inconsequential.

High Throughput: Much as with latency, high throughput convergence layers can be chosen according to the needs of the consumer and the intent of the creator.

Inconsequential Throughput: (((inconsequential throughput)) Similar to reliability, high throughput is not a necessary choice.

Integrity: The requirement of resources to be signed by contributors provides integrity.

Authenticity: The signature above also provides authenticity.

Confidentiality: Cryptographic confidentiality is provided by end-to-end encryption of resources. Other meanings of confidentiality are supported, in that resource creation and contribution are, at the abstraction of the architecture and protocols implementing it, conscious user choices. Applications MUST NOT implicitly share resources for this property to be maintained.

Security: Where appropriate for an architecture document, [BCP72] is followed. Implementations MUST ensure further compliance themselves.

Privacy: The considerations of [RFC6973], Section 7 are reflected to the best an architecture document can.

Anonymity: Anonymity is provided by making user identifiers for consumption relatively irrelevant; contributors are identified. However, each identifier can be ephemeral and limited to (a subset of) a resource.

Pseudonymity: User identifiers are fully pseudonymous.

Unlinkability: Ephemeral user identifiers provide unlinkability.

Censorship resistance: The architecture utilizes caches and custodians in order to ensure that a resource can exist in multiple places, making censorship resistance difficult here. Furthermore, ephemeral user identifiers make it difficult to censor individual people.

Accessibility: Many accessibility concerns are outside of the scope of an architecture. However, as the architecture places no constraints on resources or identifiers that make them particularly inaccessible, we can consider this property fulfilled.

One note should be made about identifiers that are hashes of something, such as e.g. user identifiers. A hash is not a particularly accessible datum. However, this architecture does not require that such data are visible to the user at all -- a hash may be an identifier in the protocols, but may be represented by a human readable and screen reader friendly string in the user interface, for example.

Mobility: The distinction between node identifiers and convergence layer addresses provides for mobility.

Multi-Homing: The same distinction permits for multi-homed nodes.

Self-Determination: Nodes are free to select how they communicate; they can reject data or interests from other nodes as dictated by local policy.

Self-determination is also guaranteed at the user layer, with some caveats. In particular, the model of resource ownership by the creator implies that contributors cannot force their way into contributing to a resource; in that sense, their self-determination is limited. But they are not equally limited when it comes to resources they create themselves.

As the above list demonstrates, this architecture incudes all of the properties defined as desirable based on the problem section, with the exception of the visbility property from REST.

The key points that make this architecture distinct from previous attempts and contributes to these results are:

1. The architecture is focused on user-to-user interactions, which are group efforts. Endpoint-to-endpoint interactions are described, but in the service of the above.
2. The architecture pivots around the notion of a group resource, which members can contribute to and/or consume.
3. The inclusion of user-to-user interaction implies the existence of user identifiers, which provide the hooks for making such resources end-to-end encrypted by default.

6. IANA Considerations

This document has no IANA actions.

7. Informative References

[AOSC] Zuboff, S., "The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power", ISBN 9781781256855, 2019.

[BCP72] Best Current Practice 72, <https://www.rfc-editor.org/info/bcp72>.
At the time of writing, this BCP comprises the following:

Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

Gont, F. and I. Arce, "Security Considerations for Transient Numeric Identifiers Employed in Network Protocols", BCP 72, RFC 9416, DOI 10.17487/RFC9416, July 2023, <https://www.rfc-editor.org/info/rfc9416>.

[BRIDGES] bridges.org, "The Real Access / Real Impact framework for improving the way that ICT is used in development", 26 December 2005, <https://archive.org/details/bridgesorg_real_access_real_impact1>.

[CAMBRIDGE-ANALYTICA] Hu, M., "Cambridge Analytica's black box", SAGE Publications, Big Data & Society vol. 7, no. 2, DOI 10.1177/2053951720938091, July 2020, <https://doi.org/10.1177/2053951720938091>.

[CCNx] "Content Centric Networking", n.d., <https://ccnx.org>.

[CENSORED-PLANET] Sundara Raman, R., Shenoy, P., Kohls, K., and R. Ensafi, "Censored Planet: An Internet-wide, Longitudinal Censorship Observatory", ACM, Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security pp. 49-66, DOI 10.1145/3372297.3417883, October 2020, <https://doi.org/10.1145/3372297.3417883>.

[COGNITIVE-INERTIA] McGuire, W., "Cognitive consistency and attitude change.", American Psychological Association (APA), The Journal of Abnormal and Social Psychology vol. 60, no. 3, pp. 345-353, DOI 10.1037/h0048563, May 1960, <https://doi.org/10.1037/h0048563>.

[CRDT] Preguiça, N., "Conflict-free Replicated Data Types: An Overview", arXiv, DOI 10.48550/ARXIV.1806.10254, 2018, <https://doi.org/10.48550/ARXIV.1806.10254>.

[DECISION-FATIGUE] Pignatiello, G., Martin, R., and R. Hickman, "Decision fatigue: A conceptual analysis", SAGE Publications, Journal of Health Psychology vol. 25, no. 1, pp. 123-135, DOI 10.1177/1359105318763510, March 2018, <https://doi.org/10.1177/1359105318763510>.

[DIG-RIGHTS] Cocito, C. and P. de Hert, "The Transformative Nature of the EU Declaration on Digital Rights and Principles: Replacing the Old Paradigm (Normative Equivalency of Rights)", Elsevier BV, DOI 10.2139/ssrn.4341816, 2023, <https://doi.org/10.2139/ssrn.4341816>.

[DRONECOMMS] Finkhäuser, J. and M. Larsen, "Reliable Command, Control and Communication Links for Unmanned Aircraft Systems: Towards compliance of commercial drones", ACM, Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings pp. 22-28, DOI 10.1145/3444950.3444954, January 2021, <https://doi.org/10.1145/3444950.3444954>.

[ECHR] European Court of Human Rights; Council of Europe, "European Convention on Human Rights", 4 November 1950, <https://www.echr.coe.int/documents/d/echr/convention_ENG>.

[FUTURE-INTERNET] Zittrain, J., "The Future of the Internet -- And How To Stop It", ISBN 9780300151244, 2008, <https://dash.harvard.edu/handle/1/4455262>.

[GDPR] Council of the European Union, "General Data Protection Regulation (GDPR)", EU Regulation 2016/679, 27 April 2016, <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32016R0679&qid=1661512309950>.

[GREEN-CODING-1] Rua, R., Fraga, T., Couto, M., and J. Saraiva, "Greenspecting Android virtual keyboards", ACM, Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems pp. 98-108, DOI 10.1145/3387905.3388600, July 2020, <https://doi.org/10.1145/3387905.3388600>.

[GREEN-CODING-2] Pereira, R., Matalonga, H., Couto, M., Castor, F., Cabral, B., Carvalho, P., de Sousa, S., and J. Fernandes, "GreenHub: a large-scale collaborative dataset to battery consumption analysis of android devices", Springer Science and Business Media LLC, Empirical Software Engineering vol. 26, no. 3, DOI 10.1007/s10664-020-09925-5, March 2021, <https://doi.org/10.1007/s10664-020-09925-5>.

[GREEN-CODING-3] Ribeiro, F., Abreu, R., and J. Saraiva, "On Understanding Contextual Changes of Failures", IEEE, 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) pp. 1036-1047, DOI 10.1109/qrs54544.2021.00112, December 2021, <https://doi.org/10.1109/qrs54544.2021.00112>.

[GREEN-HOSTING] Karyotakis, M. and N. Antonopoulos, "Web Communication: A Content Analysis of Green Hosting Companies", MDPI AG, Sustainability vol. 13, no. 2, pp. 495, DOI 10.3390/su13020495, January 2021, <https://doi.org/10.3390/su13020495>.

[HinSchG] "Hinweisgeberschutzgesetz vom 31. Mai 2023 (BGBl. 2023 I Nr. 140)", DE Hinweisgeberschutzgesetz, HinSchG, 31 May 2023, <https://www.gesetze-im-internet.de/hinschg/>.

[HYBRID-WARFARE] Dov Bachmann, S., Putter, D., and G. Duczynski, "Hybrid warfare and disinformation: A Ukraine war perspective", Wiley, Global Policy vol. 14, no. 5, pp. 858-869, DOI 10.1111/1758-5899.13257, August 2023, <https://doi.org/10.1111/1758-5899.13257>.

[I-D.draft-ietf-dtn-bibect-03] Burleigh, S. C., "Bundle-in-Bundle Encapsulation", Work in Progress, Internet-Draft, draft-ietf-dtn-bibect-03, 18 February 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bibect-03>.

[I-D.draft-irtf-hrpc-guidelines-20] Grover, G. and N. ten Oever, "Guidelines for Human Rights Protocol and Architecture Considerations", Work in Progress, Internet-Draft, draft-irtf-hrpc-guidelines-20, 4 October 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-hrpc-guidelines-20>.

[I-D.draft-nottingham-avoiding-internet-centralization-14] Nottingham, M., "Centralization, Decentralization, and Internet Standards", Work in Progress, Internet-Draft, draft-nottingham-avoiding-internet-centralization-14, 14 September 2023, <https://datatracker.ietf.org/doc/html/draft-nottingham-avoiding-internet-centralization-14>.

[I-D.draft-oran-icnrg-reflexive-forwarding-06] Oran, D. R. and D. KUTSCHER, "Reflexive Forwarding for CCNx and NDN Protocols", Work in Progress, Internet-Draft, draft-oran-icnrg-reflexive-forwarding-06, 26 September 2023, <https://datatracker.ietf.org/doc/html/draft-oran-icnrg-reflexive-forwarding-06>.

[I-D.draft-trossen-rtgwg-rosa-arch] Trossen, D., Contreras, L. M., Finkhäuser, J., and P. Mendes, "Architecture for Routing on Service Addresses", Work in Progress, Internet-Draft, draft-trossen-rtgwg-rosa-arch-01, 9 July 2023, <https://datatracker.ietf.org/doc/html/draft-trossen-rtgwg-rosa-arch-01>.

[IFTP] Tarnoff, B., "Internet for the People", ISBN 9781839762024, 2022.

[ISOC-FOUNDATION] Internet Society Foundation, "Internet Society Foundation", n.d., <https://www.isocfoundation.org/>.

[KADEMLIA] Maymounkov, P. and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric", Springer Berlin Heidelberg, Lecture Notes in Computer Science pp. 53-65, DOI 10.1007/3-540-45748-8_5, ISBN ["9783540441793", "9783540457480"], 2002, <https://doi.org/10.1007/3-540-45748-8_5>.

[MEDIA-INDEPENDENCE] Steele, J., "What Can We Learn From the Short History of Independent Media in Serbia? Radio B92, George Soros, and New Models of Media Development", SAGE Publications, The International Journal of Press/Politics vol. 29, no. 3, pp. 646-666, DOI 10.1177/19401612231170092, May 2023, <https://doi.org/10.1177/19401612231170092>.

[ML-DISINFORMATION] Katyal, S., "Artificial Intelligence, Advertising, and Disinformation", Project MUSE, Advertising & Society Quarterly vol. 20, no. 4, DOI 10.1353/asr.2019.0026, 2019, <https://doi.org/10.1353/asr.2019.0026>.

[ML-VOICE] Galyashina, E. and V. Nikishin, "AI Generated Fake Audio as a New Threat to Information Security: Legal and Forensic Aspects", SCITEPRESS - Science and Technology Publications, Proceedings of the International Scientific and Practical Conference on Computer and Information Security pp. 17-21, DOI 10.5220/0010616700003170, 2021, <https://doi.org/10.5220/0010616700003170>.

[NDN] "Named Data Networking", n.d., <https://named-data.net>.

[NIST.FIPS.186-4] "Digital signature standard (DSS)", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.186-4, 2013, <https://doi.org/10.6028/nist.fips.186-4>.

[OPENAPI] Miller, D., Ed., Whitlock, J., Ed., Gardiner, M., Ed., Ralphson, M., Ed., Ratovsky, R., Ed., and U. Sarid, Ed., "OpenAPI Specification v3.1.0", 15 February 2021, <https://spec.openapis.org/oas/v3.1.0>.

[REST] Fielding, R. T., "Architectural Styles and the Design of Network-based Software Architectures", doctoral dissertation, 2000.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/rfc/rfc4301>.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <https://www.rfc-editor.org/rfc/rfc4309>.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, DOI 10.17487/RFC4838, April 2007, <https://www.rfc-editor.org/rfc/rfc4838>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/rfc/rfc7231>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/rfc/rfc7233>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC7343] Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers Version 2 (ORCHIDv2)", RFC 7343, DOI 10.17487/RFC7343, September 2014, <https://www.rfc-editor.org/rfc/rfc7343>.

[RFC7650] Jimenez, J., Lopez-Vega, J., Maenpaa, J., and G. Camarillo, "A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD)", RFC 7650, DOI 10.17487/RFC7650, September 2015, <https://www.rfc-editor.org/rfc/rfc7650>.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/rfc/rfc768>.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/rfc/rfc791>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/rfc/rfc8410>.

[rfc8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/rfc/rfc8684>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/rfc/rfc9171>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/rfc/rfc9172>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/rfc/rfc9293>.

[RM3420] Baran, P., "On Distributed Communications: I. Introduction to Distributed Communications Networks", RAND Corporation, DOI 10.7249/rm3420, 1964, <https://doi.org/10.7249/rm3420>.

[SOAP] World Wide Web Consortium (W3C), "SOAP Version 1.2", 27 April 2007, <https://www.w3.org/TR/soap12/>.

[STD86] Internet Standard 86, <https://www.rfc-editor.org/info/std86>.
At the time of writing, this STD comprises the following:

Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[SURVEILLANCE-CAPITALISM] Wikipedia contributors, "Surveillance Capitalism", Wikipedia -- The Free Encyclopedia, 8 September 2023, <https://en.wikipedia.org/w/index.php?title=Surveillance_capitalism&oldid=1166575963>.

[UYGHUR-AI] Roberts, H., Cowls, J., Morley, J., Taddeo, M., Wang, V., and L. Floridi, "The Chinese approach to artificial intelligence: an analysis of policy, ethics, and regulation", Springer Science and Business Media LLC, AI & SOCIETY vol. 36, no. 1, pp. 59-77, DOI 10.1007/s00146-020-00992-2, June 2020, <https://doi.org/10.1007/s00146-020-00992-2>.

[UYGHUR-INTERNET] Clothey, R. and A. Meloche, "Don’t lose your moustache: community and cultural identity on the Uyghur internet in China", Informa UK Limited, Identities vol. 29, no. 3, pp. 375-394, DOI 10.1080/1070289x.2021.1964783, August 2021, <https://doi.org/10.1080/1070289x.2021.1964783>.

[UYGHUR-WAR] ROBERTS, S., "The War on the Uyghurs: China's Internal Campaign against a Muslim Minority", Princeton University Press, DOI 10.2307/j.ctvsf1qdq, ISBN ["9780691202211", "0691202214", "9780691202181"], September 2020, <https://doi.org/10.2307/j.ctvsf1qdq>.

Acknowledgments

Development of this work was performed as part of work undertaken under a grant agreement with the Internet Society Foundation [ISOC-FOUNDATION].

Index

Author's Address

Jens Finkhäuser
Interpeer gUG (haftungsbeschraenkt)
Email: ietf@interpeer.org
URI: https://interpeer.org/