Workgroup: The Interpeer Project
Published: 19 January 2026
Author: J. Finkhaeuser
Interpeer

Gap Analysis & Requirements for Human-Centric Networking

Abstract

Whereas [INTERPEER-PROBLEM-STATEMENT] describes issues with the current Internet's design, this companion document describes an evaluation framework for networked architectures derived from those problems. It then performs a gap analysis on common and some uncommon network architectures.

The second companion document is [INTERPEER-ARCHITECTURE].

About This Document

This document is a draft PIE and so adheres to the publishing process and naming described in [PIE.f92f09.00].

- This version is published at PIE.f92f09.gap-analysis
- The latest version can be found at PIE.f92f09.gap-analysis-00

Contributing

Responsibility for this document lies with The Interpeer Project.

Source code for it can be found at https://codeberg.org/interpeer/draft-jfinkhaeuser-interpeer.

Additional coordination and discussion occurs on a mailing list:

- Address: interpeer@lists.interpeer.org
- Archive
- Membership management

Table of Contents

1. Introduction

This document contains an evaluation framework for networked architectures derived from the issues outlined in [INTERPEER-PROBLEM-STATEMENT]. The framework is based on a well-respected earlier example, namely the framework described in [REST], Roy Fielding's dissertation that describes the principles that the World Wide Web is built upon. This document expands on this earlier example with specific properties addressing human-centric needs.

The second part of this document then contains a gap analysis of common network architectures, as well as some more experimental architectures that have been proposed either as alternatives to the current Internet, or evolutionary designs.

The other companion document to this one is [INTERPEER-ARCHITECTURE]. If there is a document proposing a novel architecture, it should not be surprising to conclude that none of the current architectures meet human-centric needs adequately. However, the details of the gap analysis are nonetheless worthwhile to consider, as they also form the foundation on which to evaluate such a novel architecture.

2. Gap Analysis & Requirements

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

As the many of the aforementioned issues relate to the Web, it stands to reason to start with these properties when examining other architectures. Additional desired properties are derived from the problem statements and use cases in prior sections of this document.

Section 2.1 summarises this analysis framework. The following section Section 2.2 lists desired properties derived from REST and the additions above; these are effectively requirements for a future architecture. Finally, Section 2.3 provides an analysis of several existing architectures in the light of these properties.

2.1. 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 also from the problem statement ([INTERPEER-PROBLEM-STATEMENT]), and determine if and how those architectures have such properties.

In [INTERPEER-ARCHITECTURE] we will then describe constraints to address any gaps.

2.2. Properties

Based on the framework outlined in Section 2.1 above, this section lists properties considered desirable.

2.2.1. 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.

Resilience: REST refers to this property as "Reliability", but we rename it to better distinguish it from other properties. Resilience refers to the ability to deal with (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.

2.2.2. 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).

2.2.3. HRPC

In the IETF's sister organization IRTF, the Human Rights Policy Considerations group has published [RFC9620] as a guideline document for considering aspects of protocol design that may impact human rights negatively. All considerations that are listed in [RFC9620], 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 ([INTERPEER-PROBLEM-STATEMENT]) 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.

2.2.4. 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 ([INTERPEER-PROBLEM-STATEMENT], Section 2.1.3) -- which is not only an issue in and of itself, but rather enables other problems. To borrow wording from this gap analysis framework, centralization induces, or contributes to inducing, 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.

A final property emerges in its clearest form from [INTERPEER-PROBLEM-STATEMENT] Section 3.2.2.4 on robotics.

Interest Grouping: Nodes in the networked system should be able to discover nodes based on interest topics, and communicate with likewise interested nodes in groups.

But it also has to be stressed that the same problem re-emerges at the so-called layers 8 and higher, i.e. the networking that human users of a computer network do. Here as well, the default mode of communication is to communicate around a shared interest -- which may be as simple as being born into or assigned to a group, and can be as narrowly scoped as an extremely short-lived collaboration, such as setting a date to meet next.

2.3. 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 [INTERPEER-PROBLEM-STATEMENT].

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

2.3.1. Internet

The first candidate to examine is the Internet itself. It's a building block for the Web (Section 2.3.2), 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 ([INTERPEER-PROBLEM-STATEMENT], 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 ([INTERPEER-PROBLEM-STATEMENT], 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.

2.3.2. 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.

2.3.3. 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.

2.3.4. 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 ([INTERPEER-PROBLEM-STATEMENT], 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 particularly 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.

2.3.5. Robotics

We lightly touched on robotics as an application in [INTERPEER-PROBLEM-STATEMENT], as exemplified by the Robot Operating System ([ROS2]). It barely meets any of the desired properties, and we can safely skip a more in-depth gap analysis.

That being said, it is worth mentioning as it represents a different form of interaction than the other architectures listed here.

We see both a Publish/Subscribe interaction, as well as a Request/Response interaction, as in other architectures.

But there is an additional interaction: Actions can be sent similar to requests, but are intended for situations where real-time replies to a request cannot be guaranteed. Instead, remote nodes send intermittent Status messages informing the client of the current state, and end this implied subscription with a final Result message.

This approach is listed mostly for completeness' sake here, but is picked up again in [INTERPEER-ARCHITECTURE].

2.3.6. Recursive Architectures

There are two architectures that bear special mention, and this section will treat them as largely identical, even though there are significant differences.

The first of these is [RINA], the "Recursive INternetworking Architecture" is a proposed architecture that recursively uses the same protocols in multiple nested layers. The main argument of its proponents is that the same functions are provided between different "layers", such as the layers in the OSI or the IP stack, and re-implementing those functions with completely different protocols at each layer has more drawbacks than upsides.

By contrast, the [REAL-INTERNET-ARCHITECTURE] is a *descriptive* book on the current state of the Internet, which nonetheless comes to the a similar conclusion. Most notable is that the 7 layers of the OSI model are a narrow subset of the entire set of layers of your typical "Internet" use. Depending on whether one speaks to engineers closer to the end-user or the hardware side of said stack, their view of what the 7 layers actually are will wildly diverge. A recursive view is therefore reflected also here, but it is descriptive rather than trying to provide unified solutions.

It's worth noting that the same realization is reflected less formally elsewhere in the form of so-called *overlay* and *underlay* networks. At least informally, it is well established that "networks" re-use "networks" for forwarding messages.

Where this second book provides insight in the context of this document is that it provides its own set of properties of network components in the abstract sense, and how they are composed.

The good news is that they do not contradict any of the above properties we already outlined. The downside on the other hand is that by being descriptive, they take on a somewhat limiting view when it comes to the specific requirement of *Interest Grouping*.

Specifically, they correctly identify that a session in one layer is a link for another layer. But due to the point-to-point nature of network links typical in current use, they treat group communications as functions of a router to duplicate messages from one sender to multiple participants.

On a more conceptual level, whether one considers the use case in robotics from which the property is derived, or the generalized human communications patterns, group communications are sent along a *shared medium*, and do not require this duplication.

Precisely because the book offers an analysis of many different technologies, this limiting view is both to be expected, and at the same time worth highlighting: while *overlay* networks may provide *Interest Grouping* in some way, *underlay* will eventually simulate this functionality via duplication of messags to multiple recipients.

2.3.7. Summary of Architecture Analysis

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

Furthermore, group communications (*Interest Grouping*) is fairly rarely adressed in current or potential future architectures, with the notable exception of Information-Centric Networking approaches. However, none of the examined approaches succeed at inducing all the desired properties.

3. Summary

This document has outlined desired properties of a future Internet architecture, and provided an analysis of various architectures whether they do or can meet these properties as requirements. The conclusion from this theoretical analysis is that they are incapable of fully doing so.

The driving force for this appears to be mixture of historical decision making combined with treating networking as being primarily concerned with machine-to-machine communications.

Human-to-human communications follow different paths, and have different requirements. As such, they can currently only be layered haphazardly on top of existing machine networks. Meanwhile some machine-to-machine networks exhibit similar requirements, and could also benefit from better underlay support for their needs.

The composition of this haphazard topmost layer *is* possible, but being left to the topmost layer is also seldomly implemented. This is because multiple layers of a networking stack would need to collaborate for better support, which is often beyond the practical means of any engineer solving topmost layer concerns.

In practice, this leads to a highly fragmented solution space, which allows limited interoperability only along the most basic of use cases. At the same time, each solution will exhibit only a subset of the desired properties.

This demonstrates a clear need for

1. standardizing (in terms of documents and as de-facto standards) a solution, and
2. ensuring that this solution exhibits all desired properties.

An architecture that would meet this need is proposed in the companion document, [INTERPEER-ARCHITECTURE].

3.1. Properties Summary

Below is a summary table containing the properties listed in this part of the document, limited to those deemed desirable for the stated goals.

Property	Description	Section
User-Perceived Performance	Low time to completion of queries, or "time to first Byte".	Section 2.2.1
Network Efficiency	The most efficient thing is to not use the network and use cached data.	Section 2.2.1
Scalability	The architecture supports a large number of components.	Section 2.2.1
Simplicity	From the systems theory definition, avoid complexity.	Section 2.2.1
Evolvability	Be able to change a system component without affecting the rest.	Section 2.2.1
Extensibility	Be able to add new functions safely.	Section 2.2.1
Customizability	Allow clients to influence system behaviour to their needs.	Section 2.2.1
Reusability	Be able to re-use components in the system.	Section 2.2.1

Property	Description	Section
Resilience	Be able to deal with (partial) failure of components.	Section 2.2.1
High Intermittency Tolerance	Be able to support applications in dealing with high intermittency.	Section 2.2.2
Low Intermittency Utiliziation	Be able to utilize low intermittency underlays.	
High Latency Tolerance	Be able to support applications in dealing with high latency.	Section 2.2.2
Low Latency Support	Be able to support applications that require low latency.	Section 2.2.2
Low Reliability Tolerance	Be able to support applications in dealing with low reliability.	Section 2.2.2
High Reliability Support	Be able to support applications that require high reliability.	Section 2.2.2
High Throughput Support	Applications that need to move a lot of data should be able to do so.	Section 2.2.2
Inconsequential Throughput Tolerance	It should be possible to prioritize the needs of high throughput applications against the needs of applications for which throughput is less of a concern.	Section 2.2.2
Integrity	Verify and assure integirty of the payload.	Section 2.2.3
Authenticity	Ensure that data comes from the source it claims.	Section 2.2.3
Confidentiality	Prevent data from being shared without consent.	Section 2.2.3
Security	Follow security considerations of [BCP72].	Section 2.2.3
Privacy	Protect privacy (see also [RFC6973], Section 7).	Section 2.2.3
Anonymity	Allow anonymous use.	Section 2.2.3

Property	Description	Section
Pseudonymity	Where anonymous use is impossible, allow pseudonymous use.	Section 2.2.3
Unlinkability	Prevent linkability of multiple pseudonymous activities.	Section 2.2.3
Consorship Resistance	Avoid choke points at which censorship can be enacted.	Section 2.2.3
Accessibility	Provide an enabling environment for all.	Section 2.2.3
Mobility	Tolerate changes in connectivity (network attachment).	Section 2.2.4
Multi-Homing	Tolerate and be able to utilize multiple points of connectivity/network attachment.	Section 2.2.4
Self-Determination	Permit components to make local decisions.	Section 2.2.4
Interest Grouping	Support components in communicating with other components with related interests.	Section 2.2.4

Table 1: Properties of a Human-Centric Architecture

Note that "time to first Byte" is common Web performance measurement meant to roughly indicate how long a Web user has to wait until a sent request returns results that the Browser may be able to render.

Below are examined properties that have deliberately been removed from the table above, listed for the sake of completeness.

Property	Reason for Removal	Section
Visibility	Conflicts with more critical properties such as privacy, confidentiality, etc.	Section 2.2.1
Network Performance	Better examined in Section 2.2.2, and so replaced by other properties.	Section 2.2.1
Portability	Any code is also data. It should not be a property of a Network architecture to distinguish between the two, but rather the property of a system built upon a network architecture.	Section 2.2.1

Table 2: Examined properties not included

4. References

4.1. Normative References

[INTERPEER-ARCHITECTURE] Finkhaeuser, J., "Architecture for Human-Centric Networking", PIE PIE.f92f09.architecture-00, 19 January 2026, <https://specs.interpeer.org/PIE.f92f09.architecture/PIE.f92f09.architecture-00>.

[INTERPEER-PROBLEM-STATEMENT] Finkhaeuser, J., "Problem Statement & Gap Analysis for Human-Centric Networking", PIE PIE.f92f09.problem-statement-00, 22 July 2025, <https://specs.interpeer.org/PIE.f92f09.problem-statement/PIE.f92f09.problem-statement-00>.

[INTERPEER-REQUIREMENTS] Finkhaeuser, J., "Gap Analysis & Requirements for Human-Centric Networking", PIE PIE.f92f09.gap-analysis-00, 19 January 2026, <https://specs.interpeer.org/PIE.f92f09.gap-analysis/PIE.f92f09.gap-analysis-00>.

[PIE.f92f09.00] Finkhaeuser, J., "PIEs - Proposals for Interpeer Enhancement", PIE PIE.f92f09.00-00, 14 March 2025, <https://specs.interpeer.org/PIE.f92f09.00/PIE.f92f09.00-00>.

4.2. Informative References

[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>.

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

[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-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>.

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

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

[NGI-Assure] PNO Digital Srl, "NGI Assure", DOI 10.3030/957073, Grant Agreement ID 957073, 31 August 2024, <https://doi.org/10.3030/957073>.

[NGI0-Discovery] Stichting NLNet, "NGI Zero Discovery", DOI 10.3030/825322, Grant Agreement ID 825322, 1 November 2018, <https://doi.org/10.3030/825322>.

[REAL-INTERNET-ARCHITECTURE] Zave, P. and J. Rexford, "The Real Internet Architecture", ISBN 9780691255804, n.d..

[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>.

[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>.

[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>.

[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>.

[RFC9620] Grover, G. and N. ten Oever, "Guidelines for Human Rights Protocol and Architecture Considerations", RFC 9620, DOI 10.17487/RFC9620, September 2024, <https://www.rfc-editor.org/rfc/rfc9620>.

[RINA] Day, J., "Patterns in Network Architecture: A Return to Fundamentals", ISBN 9780132252423, 2008.

[ROS2] The ROS Community, "Robot Operating System", n.d., <https://ros.org/>.

[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>.

Acknowledgments

Development of this document started as work undertaken under a grant agreement with the Internet Society Foundation [ISOC-FOUNDATION], but has since seen a number of revisions. Some revisions are inspired by work undertaken under grant agreements from Horizon Europe, specifically [NGI0-Discovery] and [NGI-Assure].

Copyright Notice

Index

Author's Address

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