

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 28 April 2022

J. Arkko  
Ericsson  
T. Hardie  
Cisco  
T. Pauly  
Apple  
M. Kühlewind  
Ericsson  
25 October 2021

Considerations on Application - Network Collaboration Using Path Signals  
draft-arkko-iab-path-signals-collaboration-01

Abstract

Encryption and other security mechanisms are on the rise on all layers of the stack, protecting user data and making network operations more secured. Further, encryption is also a tool to address ossification that has been observed over time. Separation of functions into layers and enforcement of layer boundaries based on encryption supports selected exposure to those entities that are addressed by a function on a certain layer. A clear separation supports innovation and also enables new opportunities for collaborative functions. RFC 8558 describes path signals as messages to or from on-path elements. This document states principles for designing mechanisms that use or provide path signals and calls for actions on specific valuable cases.

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 April 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Past Guidance . . . . .	4
3. Principles . . . . .	5
3.1. Intentional Distribution . . . . .	6
3.2. Minimum Set of Entities . . . . .	7
3.3. Consent of Parties . . . . .	7
3.4. Minimum Information . . . . .	8
3.5. Carrying Information . . . . .	9
3.6. Protecting Information and Authentication . . . . .	9
4. Further Work . . . . .	10
5. Acknowledgments . . . . .	11
6. Informative References . . . . .	11
Authors' Addresses . . . . .	14

## 1. Introduction

Encryption, besides its important role in security in general, provides a tool to control information access and protects against ossification by avoiding unintended dependencies and requiring active maintenance. The increased deployment of encryption provides an opportunity to reconsider parts of Internet architecture that have rather used implicit derivation of input signals for on-path functions than explicit signaling, as recommended by RFC 8558 [RFC8558].

RFC 8558 defines the term path signals as signals to or from on-path elements. Today path signals are often implicit, e.g. derived from in-clear end-to-end information by e.g. examining transport protocols. For instance, on-path elements use various fields of the TCP header [RFC0793] to derive information about end-to-end latency as well as congestion. These techniques have evolved because the information was simply available and use of this information is

easier and therefore also cheaper than any explicit and potentially complex cooperative approach.

As such, applications and networks have evolved their interaction without comprehensive design for how this interaction should happen or which information would be desired for a certain function. This has lead to a situation where sometimes information is used that maybe incomplete, incorrect, or only indirectly representative of the information that was actually desired. In addition, dependencies on information and mechanisms that were designed for a different function limits the evolvability of the protocols in question.

The unplanned interaction ends up having several negative effects:

- \* Ossifying protocols by introducing unintended parties that may not be updating
- \* Creating systemic incentives against deploying more secure or private versions of protocols
- \* Basing network behaviour on information that may be incomplete or incorrect
- \* Creating a model where network entities expect to be able to use rich information about sessions passing through

For instance, features such as DNS resolution or TLS setup have been used beyond their original intent, such as in name-based filtering. MAC addresses have been used for access control, captive portal implementations that employ taking over cleartext HTTP sessions, and so on.

Increased deployment of encryption can and will change this situation. For instance, QUIC replaces TCP for various application and protects all end-to-end signals to only be accessible by the endpoint, ensuring evolvability [RFC9000]. QUIC does expose information dedicated for on-path elements to consume by design explicit signal for specific use cases, such as the Spin bit for latency measurements or connection ID that can be used by load balancers [I-D.ietf-quic-manageability] but information is limited to only those use cases. Each new use cases requires additional action.

Explicit signals that are specifically designed for the use of on-path function leave all other information is appropriately protected. This enables an architecturally clean approach and evolvability, while allowing an information exchange that is important for improving the quality of experience for users and efficient management of the network infrastructure built for them.

This draft discusses different approaches for explicit collaboration and provides guidance on architectural principles to design new mechanisms. Section 2 discusses past guidance. Section 3 discusses principles that good design can follow. This section also provides some examples and explanation of situations that not following the principles can lead to. Section 4 points to topics that need more to be looked at more carefully before any guidance can be given.

## 2. Past Guidance

Incentives are a well understood problem in general but perhaps not fully internalised for various designs attempting to establish collaboration between applications and path elements. The principle is that both receiver and sender of information must acquire tangible and immediate benefits from the communication, such as improved performance.

A related issue is understanding whether a business model or ecosystem change is needed. For instance, relative prioritization between different flows of a user or an application does not require agreements or payments. But requesting prioritization over other people's traffic may imply that you have to pay for that which may not be easy even for a single provider let alone across many.

But on to more technical aspects.

The main guidance in [RFC8558] is to be aware that implicit signals will be used whether intended or not. Protocol designers should consider either hiding these signals when the information should not be visible, or using explicit signals when it should be.

[RFC9049] discusses many past failure cases, a catalogue of past issues to avoid. It also provides relevant guidelines for new work, from discussion of incentives to more specific observations, such as the need for outperforming end-to-end mechanisms (Section 4.4), considering the need for per-connection state (Section 4.6), taking into account the latency involved in reacting to distant signals, and so on.

There are also more general guidance documents, e.g., [RFC5218] discusses protocol successes and failures, and provides general advice on incremental deployability etc. Internet Technology Adoption and Transition (ITAT) workshop report [RFC7305] is also recommended reading on this same general topic. And [RFC6709] discusses protocol extensibility, and provides general advice on the importance of global interoperability and so on.

### 3. Principles

This section attempts to provide some architecture-level principles that would help future designers and recommend useful models to apply.

A large number of our protocol mechanisms today fall into one of two categories: authenticated and private communication that is only visible to the end-to-end nodes; and unauthenticated public communication that is visible to all nodes on a path. RFC 8558 explores the line between data that is protected and path signals.

There is a danger in taking a position that is too extreme towards either exposing all information to the path, or hiding all information from the path.

Exposed information encourages pervasive monitoring, which is described in RFC 7258 [RFC7258]. Exposed information may also be used for commercial purposes, or form a basis for filtering that the applications or users do not desire.

But a lack of all path signaling, on the other hand, may be harmful to network management, debugging, or the ability for networks to provide the most efficient services. There are many cases where elements on the network path can provide beneficial services, but only if they can coordinate with the endpoints. It also affects the ability of service providers and others observe why problems occur [RFC9075].

This situation is sometimes cast as an adversarial tradeoff between privacy and the ability for the network path to provide intended functions. However, this is perhaps an unnecessarily polarized characterization as a zero-sum situation. Not all information passing implies loss of privacy. For instance, performance information or preferences do not require disclosing user or application identity or what content is being accessed, network congestion status information does not have reveal network topology or the status of other users, and so on.

This points to one way to resolve the adversity: the careful of design of what information is passed.

Another approach is to employ explicit trust and coordination between endpoints and network devices. VPNs are a good example of a case where there is an explicit authentication and negotiation with a network path element that's used to optimize behavior or gain access to specific resources.

The goal of improving privacy and trust on the Internet does not necessarily need to remove the ability for network elements to perform beneficial functions. We should instead improve the way that these functions are achieved. Our goals should be:

- \* To ensure that information is distributed intentionally, not accidentally;
- \* to understand the privacy and other implications of any distributed information;
- \* to ensure that the information distribution targets the intended parties; and
- \* to gate the distribution of information on the consent of the relevant parties

These goals for distribution apply equally to senders, receivers, and path elements.

We can establish some basic questions that any new network path functions should consider:

- \* What is the minimum set of entities that need to be involved?
- \* What is the minimum information each entity in this set needs?
- \* Which entities must consent to the information exchange?

If we look at many of the ways network path functions are achieved today, we find that many if not most of them fall short the standard set up by the questions above. Too often, they send unnecessary information or fail to limit the scope of distribution or providing any negotiation or consent.

Going forward, new standards work in the IETF needs to focus on addressing this gap by providing better alternatives and mechanisms for providing path functions. Note that not all of these functions can be achieved in a way that preserves a high level of user privacy from the network; in such cases, it is incumbent upon us to not ignore the use case, but instead to define the high bar for consent and trust, and thus define a limited applicability for those functions.

### 3.1. Intentional Distribution

This guideline is best expressed in RFC 8558:

"Fundamentally, this document recommends that implicit signals should be avoided and that an implicit signal should be replaced with an explicit signal only when the signal's originator intends that it be used by the network elements on the path. For many flows, this may result in the signal being absent but allows it to be present when needed."

This guideline applies also in the other direction as well. For instance, a network element should not unintentionally leak information that is visible to endpoints. An explicit decision is needed for a specific information to be provided, along with analysis of the security and privacy implications of that information.

### 3.2. Minimum Set of Entities

It is recommended that a design identify the minimum number of entities needed to share a specific signal required for an identified function. In some cases this will be a very limited set, e.g. when the application needs to provide a signal to a specific gateway function. In other cases, such as congestion control, a signal might be shared with every router along the path, since each should be aware of the congestion.

While it is tempting to consider removing these limitations in the context of closed, private networks, each interaction is still best considered separately, rather than simply allowing all information exchanges within the closed network. Even in a closed network with carefully managed components there may be compromised components, as evidenced in the most extreme way by the Stuxnet worm that operated in an airgapped network. Most "closed" networks have at least some needs and means to access the rest of the Internet, and should not be modeled as if they had an impenetrable security barrier.

### 3.3. Consent of Parties

Consent and trust must determine the distribution of information. The set of entities that need to consent is determined by the scope and specificity of the information being shared.

Three distinct types of consent are recommended for collaboration or information sharing:

- \* A corollary of the intentional distribution is that the sender needs to agree to sending the information. Or that the requester for an action needs to agree to make a request; it should not be an implicit decision by the receiver that information was sent or a request was made, just because a packet happened to be formed in a particular way.

- \* At the same time, the recipient of information or the target of a request should agree to wishing to receive the information. It should not be burdened with extra processing if it does not have willingness or a need to do so. This happens naturally in most protocol designs, but has been a problem for some cases where "slow path" packet processing is required or implied, and the recipient or router did not have willingness for this.
- \* Internet communications are not made for the applications, they are ultimately made on behalf of users. Information relating to the users is something that both networks and applications should be careful with, and not be shared without the user's consent. This is not always easy, as the interests of the user and (for instance) application developer may not always coincide; some applications may wish to collect more information about the user than the user would like.

As a result, typically an application's consent is not the same as the user's consent.

#### 3.4. Minimum Information

Parties should provide only the information that is needed for the other party to perform the collaboration task that is desired by this party, and not more. This applies to information sent by an application about itself, information sent about users, or information sent by the network.

An architecture can follow the guideline from RFC 8558 in using explicit signals, but still fail to differentiate properly between information that should be kept private and information that should be shared.

In looking at what information can or cannot easily be passed, we can look at both information from the network to the application, and from the application to the network.

For the application to the network direction, user-identifying information can be problematic for privacy and tracking reasons. Similarly, application identity can be problematic, if it might form the basis for prioritization or discrimination that the application provider may not wish to happen. It may also have undesirable economic consequences, such as extra charges for the consumer from a priority service where a regular service would have worked.

On the other hand, as noted above, information about general classes of applications may be desirable to be given by application



providers, if it enables prioritization that would improve service, e.g., differentiation between interactive and non-interactive services.

For the network to application direction there is similarly sensitive information, such as the precise location of the user. On the other hand, various generic network conditions, predictive bandwidth and latency capabilities, and so on might be attractive information that applications can use to determine, for instance, optimal strategies for changing codecs. However, information given by the network about load conditions and so on should not form a mechanism to provide a side-channel into what other users are doing.

While information needs to be specific and provided on a per-need basis, it is often beneficial to provide declarative information that, for instance, expresses application needs than makes specific requests for action.

### 3.5. Carrying Information

There is a distinction between what information is passed and how it is carried. The actually sent information may be limited, while the mechanisms for sending or requesting information can be capable of sending much more.

There is a tradeoff here between flexibility and ensuring the minimality of information in the future. The concern is that a fully generic data sharing approach between different layers and parties could potentially be misused, e.g., by making the availability of some information a requirement for passing through a network.

This is undesirable, and our recommendation is to employ very targeted, minimal information carriage mechanisms.

### 3.6. Protecting Information and Authentication

Some simple forms of information often exist in cleartext form, e.g., ECN bits from routers are generally not authenticated or integrity protected. This is possible when the information exchanges are advisory in their nature, and do not carry any significantly sensitive information from the parties.

In other cases it may be necessary to establish a secure channel for communication with a specific other party, e.g., between a network element and an application. This channel may need to be authenticated, integrity protected and encrypted. This is necessary, for instance, if the particular information or request needs to be shared in confidentiality only with a particular, trusted node, or there's

a danger of an attack where someone else may forge messages that could endanger the communication.

However, it is important to note that authentication does not equal trust. Whether a communication is with an application server or network element that can be shown to be associated with a particular domain name, it does not follow that information about the user can be safely sent to it.

In some cases, the ability of a party to show that it is on the path can be beneficial. For instance, an ICMP error that refers to a valid flow may be more trustworthy than any ICMP error claiming to come from an address.

Other cases may require more substantial assurances. For instance, a specific trust arrangement may be established between a particular network and application. Or technologies such as confidential computing can be applied to provide an assurance that information processed by a party is handled in an appropriate manner.

#### 4. Further Work

This is a developing field, and it is expected that our understanding continues to grow. The recent changes with regards to much higher use of encryption at different protocol layers, the consolidation or more and more traffic to the same destinations, and so on have also greatly impacted the field.

While there are some examples of modern, well-designed collaboration mechanisms, clearly more work is needed. Many complex cases would require significant developments in order to become feasible.

Some of the most difficult areas are listed below. Research on these topics would be welcome.

- \* Business arrangements. Many designs - for instance those related to quality-of-service - involve an expectation of paying for a service. This is possible and has been successful within individual domains, e.g., users can pay for higher data rates or data caps in their ISP networks. However, it is a business-wise much harder proposition for end-to-end connections across multiple administrative domains [Claffy2015] [RFC9049].
- \* Secure communications with path elements. This has been a difficult topic, both from the mechanics and scalability point view, but also because there is no easy way to find out which parties to trust or what trust roots would be appropriate. Some application-network element interaction designs have focused on

information (such as ECN bits) that is distributed openly within a path, but there are limited examples of designs with secure information exchange with specific nodes.

- \* The use of path signals for reducing the effects of denial-of-service attacks, e.g., in the form of modern "source quench" designs.
- \* Ways of protecting information when held by network elements or servers, beyond communications security. For instance, host applications commonly share sensitive information about the user's actions with other nodes, starting from basic data such as domain names learned by DNS infrastructure or source and destination addresses and protocol header information learned by all routers on the path, to detailed end user identity and other information learned by the servers. Some solutions are starting to exist for this but are not widely deployed, at least not today [Oblivious] [PDoT] [I-D.arkko-dns-confidential] [I-D.thomson-http-oblivious]. These solutions address also very specific parts of the issue, and more work remains.
- \* Sharing information from networks to applications. Some proposals have been made in this space (see, e.g., [I-D.flinck-mobile-throughput-guidance]) but there are no successful or deployed mechanisms today.

## 5. Acknowledgments

The authors would like to thank everyone at the IETF, the IAB, and our day jobs for interesting thoughts and proposals in this space. Fragments of this document were also in [I-D.per-app-networking-considerations] and [I-D.arkko-path-signals-information] that were published earlier. We would also like to acknowledge [I-D.trammell-stackevo-explicit-coop] for presenting similar thoughts. Finally, the authors would like to thank Adrian Farrell, Toerless Eckert, and Jeffrey Haas for useful feedback in the IABOPEN session at IETF-111.

## 6. Informative References

[Claffy2015]

kc Claffy, . and D. Clark, "Adding Enhanced Services to the Internet: Lessons from History", TPRC 43: The 43rd Research Conference on Communication, Information and Internet Policy Paper , April 2015.

[I-D.arkko-dns-confidential]

Arkko, J. and J. Novotny, "Privacy Improvements for DNS

Resolution with Confidential Computing", Work in Progress, Internet-Draft, draft-arkko-dns-confidential-02, 2 July 2021, <<https://www.ietf.org/archive/id/draft-arkko-dns-confidential-02.txt>>.

[I-D.arkko-path-signals-information]

Arkko, J., "Considerations on Information Passed between Networks and Applications", Work in Progress, Internet-Draft, draft-arkko-path-signals-information-00, 22 February 2021, <<https://www.ietf.org/archive/id/draft-arkko-path-signals-information-00.txt>>.

[I-D.flinck-mobile-throughput-guidance]

Jain, A., Terzis, A., Flinck, H., Sprecher, N., Arunachalam, S., Smith, K., Devarapalli, V., and R. B. Yanai, "Mobile Throughput Guidance Inband Signaling Protocol", Work in Progress, Internet-Draft, draft-flinck-mobile-throughput-guidance-04, 13 March 2017, <<https://www.ietf.org/archive/id/draft-flinck-mobile-throughput-guidance-04.txt>>.

[I-D.ietf-quic-manageability]

Kuehlewind, M. and B. Trammell, "Manageability of the QUIC Transport Protocol", Work in Progress, Internet-Draft, draft-ietf-quic-manageability-13, 2 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-quic-manageability-13.txt>>.

[I-D.per-app-networking-considerations]

Colitti, L. and T. Pauly, "Per-Application Networking Considerations", Work in Progress, Internet-Draft, draft-per-app-networking-considerations-00, 15 November 2020, <<https://www.ietf.org/archive/id/draft-per-app-networking-considerations-00.txt>>.

[I-D.thomson-http-oblivious]

Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-thomson-http-oblivious-02, 24 August 2021, <<https://www.ietf.org/archive/id/draft-thomson-http-oblivious-02.txt>>.

[I-D.trammell-stackevo-explicit-coop]

Trammell, B., "Architectural Considerations for Transport Evolution with Explicit Path Cooperation", Work in Progress, Internet-Draft, draft-trammell-stackevo-explicit-coop-00, 23 September 2015, <<https://www.ietf.org/archive/id/draft-trammell-stackevo-explicit-coop-00.txt>>.

- [Oblivious] Schmitt, P., "Oblivious DNS: Practical privacy for DNS queries", Proceedings on Privacy Enhancing Technologies 2019.2: 228-244 , 2019.
- [PDoT] Nakatsuka, Y., Paverd, A., and G. Tsudik, "PDoT: Private DNS-over-TLS with TEE Support", Digit. Threat.: Res. Pract., Vol. 2, No. 1, Article 3, <https://dl.acm.org/doi/fullHtml/10.1145/3431171> , February 2021.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC5218] Thaler, D. and B. Aboba, "What Makes for a Successful Protocol?", RFC 5218, DOI 10.17487/RFC5218, July 2008, <<https://www.rfc-editor.org/info/rfc5218>>.
- [RFC6709] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, DOI 10.17487/RFC6709, September 2012, <<https://www.rfc-editor.org/info/rfc6709>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7305] Lear, E., Ed., "Report from the IAB Workshop on Internet Technology Adoption and Transition (ITAT)", RFC 7305, DOI 10.17487/RFC7305, July 2014, <<https://www.rfc-editor.org/info/rfc7305>>.
- [RFC8558] Hardie, T., Ed., "Transport Protocol Path Signals", RFC 8558, DOI 10.17487/RFC8558, April 2019, <<https://www.rfc-editor.org/info/rfc8558>>.
- [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/info/rfc9000>>.
- [RFC9049] Dawkins, S., Ed., "Path Aware Networking: Obstacles to Deployment (A Bestiary of Roads Not Taken)", RFC 9049, DOI 10.17487/RFC9049, June 2021, <<https://www.rfc-editor.org/info/rfc9049>>.

[RFC9075] Arkko, J., Farrell, S., Kühlewind, M., and C. Perkins,  
"Report from the IAB COVID-19 Network Impacts Workshop  
2020", RFC 9075, DOI 10.17487/RFC9075, July 2021,  
<<https://www.rfc-editor.org/info/rfc9075>>.

Authors' Addresses

Jari Arkko  
Ericsson

Email: [jari.arkko@ericsson.com](mailto:jari.arkko@ericsson.com)

Ted Hardie  
Cisco

Email: [ted.ietf@gmail.com](mailto:ted.ietf@gmail.com)

Tommy Pauly  
Apple

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

Mirja Kühlewind  
Ericsson

Email: [mirja.kuehlewind@ericsson.com](mailto:mirja.kuehlewind@ericsson.com)

Network Working Group  
Internet-Draft  
Updates: 3172 (if approved)  
Intended status: Informational  
Expires: January 13, 2022

K. Davies  
IANA  
J. Arkko  
Ericsson  
July 12, 2021

Nameservers for the Address and Routing Parameter Area ("arpa") Domain  
draft-iab-arpa-authoritative-servers-01

Abstract

This document describes revisions to operational practices to separate function of the "arpa" top-level domain in the DNS from its historical operation alongside the DNS root zone.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 13, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Requirements for the "arpa" zone . . . . .	3
3. Transition Process . . . . .	3
3.1. Dedicated nameserver hostnames . . . . .	3
3.2. Separation of infrastructure . . . . .	4
3.3. Zone administration . . . . .	4
3.4. Conclusion of process . . . . .	4
4. IANA Considerations . . . . .	5
5. Security Considerations . . . . .	5
6. References . . . . .	5
6.1. Normative References . . . . .	5
6.2. Informative References . . . . .	5
Acknowledgments . . . . .	6
Authors' Addresses . . . . .	6

## 1. Introduction

The "arpa" top-level domain [RFC3172] is designated as an "infrastructure domain" to support techniques defined by Internet standards. Zones under the "arpa" domain provide various mappings, such as IP addresses to domain names and E.164 numbers to URIs. It also contains special use names such as "home", which is a non-unique name used in residential networks.

Historically, the "arpa" zone has been hosted on almost all of the root nameservers, and [RFC3172] envisages the "arpa" domain to be "sufficiently critical that the operational requirements for the root nameservers apply to the operational requirements of the "arpa" servers". To date, this has been implemented by serving the "arpa" domain directly on a subset of the root server infrastructure.

This bundling of root nameserver and "arpa" nameserver operations has entwined management of the zones' contents and their infrastructure. As a result, some proposals under consideration by the IETF involving the "arpa" zone have been discarded due to the risk of conflict with operations associated with managing the content of the root zone, or administering the root nameservers.

The separation described in this document resolves operational impacts of synchronizing edits to the root zone and the "arpa" zone by eliminating the current dependency and allowing more tailored operations based on the unique requirements of each zone.



## 2. Requirements for the "arpa" zone

The "arpa" domain continues to play a role in critical Internet operations, and this change does not propose weakening operational requirements described in [RFC3172] for the domain. Future operational requirements for the "arpa" domain are encouraged to follow strong baseline requirements such as those documented in [RFC7720].

Changes to the administration of the "arpa" zone do not alter the management practices of other zones delegated within the "arpa" namespace. For example, "ip6.arpa" would continue to be managed in accordance with [RFC5855].

## 3. Transition Process

The process will dedicate new hostnames to the servers authoritative for the "arpa" zone, but will initially serve the "arpa" zone from the same hosts.

Once completed, subsequent transitional phases could include using new hosts to replace or augment the existing root nameserver hosts, and separation of the editing and distribution of the "arpa" zone from necessarily being connected to the root zone. Any future management considerations regarding how such changes may be performed are beyond the scope of this document.

### 3.1. Dedicated nameserver hostnames

Consistent with the use of the "arpa" namespace itself to host name servers for other delegations in the "arpa" zone ([RFC5855]), this document specifies a new namespace of "ns.arpa", with the nameserver set for the "arpa" zone to be initially labelled as follows:

```
a.ns.arpa
b.ns.arpa
c.ns.arpa
...
```

Dedicated hostnames eliminate a logical dependency that requires the coordinated editing of the nameservers for the "arpa" zone and the root zone. This component of this transition does not require the underlying hosts that provide "arpa" name service (that is, the root nameservers) be altered. The "arpa" zone will initially map the new hostnames to the same IP addresses that already provide service under the respective hostnames within root-servers.net.

Because these nameservers are completely within the "arpa" zone, they will require glue records in the root zone. This is consistent with current practice and requires no operational changes to the root zone.

### 3.2. Separation of infrastructure

After initially migrating the "arpa" zone to use hostnames that are not shared with the root zone, the underlying name service is expected to evolve such that it no longer directly aligns to a subset of root nameserver instances. With no shared infrastructure between the root nameservers and the "arpa" nameservers, future novel applications for the "arpa" zone may be possible.

Any subsequent changes to the parties providing name service for the zone is considered a normal management responsibility, and would be performed in accordance with [RFC3172].

### 3.3. Zone administration

Publication of the "arpa" zone file to the authoritative "arpa" name servers is currently undertaken alongside the root zone maintenance functions. Upon the separation of the "arpa" infrastructure from the root nameserver infrastructure, publication of the "arpa" zone no longer necessarily needs to be technically linked or inter-related to the root zone publication mechanisms.

### 3.4. Conclusion of process

Full technical separation of operations of the "arpa" zone and root zone minimally requires the following to be satisfied:

- o The "arpa" zone no longer shares any hostnames in its NS-set with the root zone;
- o The hosts that provide authoritative name service are not the same hosts as the root nameservers, do not share any IPv4 or IPv6 addresses with the root servers, and are sufficiently separately provisioned such that any unique "arpa" zone requirements can be deployed without affecting how root zone service is provided;
- o The editorial and publication process for the "arpa" zone has any common dependencies with the root zone process removed, so that the "arpa" zone can be managed, edited and provisioned wholly independently of the root zone.

Such separation is ultimately sought to allow for novel uses of the "arpa" zone without the risk of inadvertently impacting root zone and

root server operations. It is recognized that achieving this state requires a deliberative process involving significant coordination to ensure impacts are minimized.

#### 4. IANA Considerations

The IANA shall coordinate the creation of the "ns.arpa" namespace and populate it with address records that reflect the IP addresses of the contemporary root servers documented within "root-servers.net" as its initial state. The namespace may either be provisioned directly within the "arpa" zone (as an empty non-terminal), or through establishing a dedicated "ns.arpa" zone, according to operational requirements.

The IANA will initially migrate the 12 NS records for the "arpa" zone to point to their respective new entries in the "ns.arpa" domain.

Subsequently, the IAB and IANA will consult and coordinate with all relevant parties on activity to reduce or eliminate reliance upon root zone and root server infrastructure for serving the "arpa" zone. Such changes will be performed in compliance with [RFC3172] and shall be conducted with all due care and deliberation to mitigate potential impacts on critical infrastructure.

#### 5. Security Considerations

The security of the "arpa" zone is not necessarily impacted by any aspects of these changes. Robust practices associated with administering the content of the zone (including signing the zone with DNSSEC) as well as its distribution will continue to be necessary.

#### 6. References

##### 6.1. Normative References

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <<https://www.rfc-editor.org/info/rfc3172>>.

##### 6.2. Informative References

[RFC5855] Abley, J. and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", BCP 155, RFC 5855, DOI 10.17487/RFC5855, May 2010, <<https://www.rfc-editor.org/info/rfc5855>>.

[RFC7720] Blanchet, M. and L-J. Liman, "DNS Root Name Service Protocol and Deployment Requirements", BCP 40, RFC 7720, DOI 10.17487/RFC7720, December 2015, <<https://www.rfc-editor.org/info/rfc7720>>.

#### Acknowledgments

Thank you Alyssa Cooper, Michelle Cotton, Lars-Johan Liman, Wes Hardaker, Ted Hardie, Paul Hoffman, Russ Housley, Oscar Robles-Garay, Duane Wessels and Suzanne Woolf for providing review and feedback.

#### Authors' Addresses

Kim Davies  
Internet Assigned Numbers Authority  
PTI/ICANN  
12025 Waterfront Drive  
Los Angeles 90094  
United States of America

Email: [kim.davies@iana.org](mailto:kim.davies@iana.org)

Jari Arkko  
Ericsson Research  
02700 Kauniainen  
Finland

Email: [jari.arkko@ericsson.com](mailto:jari.arkko@ericsson.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 12 November 2022

M. Thomson  
Mozilla  
D. Schinazi  
Google LLC  
11 May 2022

The Harmful Consequences of the Robustness Principle  
draft-iab-protocol-maintenance-06

Abstract

The robustness principle, often phrased as "be conservative in what you send, and liberal in what you accept", has long guided the design and implementation of Internet protocols. The posture this statement advocates promotes interoperability in the short term, but can negatively affect the protocol ecosystem over time. For a protocol that is actively maintained, the robustness principle can, and should, be avoided.

About This Document

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

Status information for this document may be found at  
<https://datatracker.ietf.org/doc/draft-iab-protocol-maintenance/>.

Discussion of this document takes place on the EDM IAB Program mailing list (<mailto:edm@iab.org>), which is archived at <https://www.iab.org/mailman/listinfo/edm>.

Source for this draft and an issue tracker can be found at <https://github.com/intarchboard/draft-protocol-maintenance>.

Status of This Memo

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

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

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

This Internet-Draft will expire on 12 November 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

#### Table of Contents

1. Introduction . . . . .	2
2. Fallibility of Specifications . . . . .	3
3. Protocol Decay . . . . .	4
4. Ecosystem Effects . . . . .	5
5. Active Protocol Maintenance . . . . .	7
6. Extensibility . . . . .	8
7. Virtuous Intolerance . . . . .	9
8. Exclusion . . . . .	10
9. Security Considerations . . . . .	11
10. IANA Considerations . . . . .	11
11. Informative References . . . . .	11
Acknowledgments . . . . .	13
Authors' Addresses . . . . .	13

#### 1. Introduction

The robustness principle has been hugely influential in shaping the design of the Internet. As stated in the IAB document on Architectural Principles of the Internet [RFC1958], the robustness principle advises to:

Be strict when sending and tolerant when receiving.  
Implementations must follow specifications precisely when sending to the network, and tolerate faulty input from the network. When in doubt, discard faulty input silently, without returning an error message unless this is required by the specification.

This simple statement captures a significant concept in the design of interoperable systems. Many consider the application of the robustness principle to be instrumental in the success of the Internet as well as the design of interoperable protocols in general.

Time and experience shows that negative consequences to interoperability accumulate over time if implementations apply the robustness principle. This problem originates from an assumption implicit in the principle that it is not possible to affect change in a system the size of the Internet. That is, the idea that once a protocol specification is published, changes that might require existing implementations to change are not feasible.

Many problems that might lead to applications of the robustness principle are avoided for protocols under active maintenance. Active protocol maintenance is where a community of protocol designers, implementers, and deployers work together to continuously improve and evolve protocol specifications alongside implementations and deployments of those protocols. A community that takes an active role in the maintenance of protocols will no longer need to rely on the robustness principle to avoid interoperability issues.

There is good evidence to suggest that many important protocols are routinely maintained beyond their inception. In particular, a sizeable proportion of IETF activity is dedicated to the stewardship of existing protocols. This document serves primarily as a record of the hazards inherent in applying the robustness principle and to offer an alternative strategy for handling interoperability problems in deployments.

Ideally, protocol implementations never have to apply the robustness principle. Or, where it is unavoidable, use of the robustness principle is viewed as a short term workaround that needs to be quickly reverted.

## 2. Fallibility of Specifications

The context from which the robustness principle was developed provides valuable insights into its intent and purpose. The earliest form of the principle in the RFC series (the Internet Protocol specification [RFC0760]) is preceded by a sentence that reveals the motivation for the principle:

While the goal of this specification is to be explicit about the protocol there is the possibility of differing interpretations. In general, an implementation should be conservative in its sending behavior, and liberal in its receiving behavior.

This formulation of the principle expressly recognizes the possibility that the specification could be imperfect. This contextualizes the principle in an important way.

An imperfect specification is natural, largely because it is more important to proceed to implementation and deployment than it is to perfect a specification. A protocol benefits greatly from experience with its use. A deployed protocol is immeasurably more useful than a perfect protocol. The robustness principle is a tool that is suited to early phases of system design.

As demonstrated by the IAB document on Successful Protocols [RFC5218], success or failure of a protocol depends far more on factors like usefulness than on technical excellence. Timely publication of protocol specifications, even with the potential for flaws, likely contributed significantly to the eventual success of the Internet.

The problem is therefore not with the premise, but with its conclusion: the robustness principle itself.

### 3. Protocol Decay

The application of the robustness principle to the early Internet, or any system that is in early phases of deployment, is expedient. Applying the principle defers the effort of dealing with interoperability problems, which prioritizes progress. However, deferral can amplify the ultimate cost of handling interoperability problems.

Divergent implementations of a specification emerge over time. When variations occur in the interpretation or expression of semantic components, implementations cease to be perfectly interoperable.

Implementation bugs are often identified as the cause of variation, though it is often a combination of factors. Application of a protocol to uses that were not anticipated in the original design, or ambiguities and errors in the specification are often confounding factors. Disagreements on the interpretation of specifications should be expected over the lifetime of a protocol.

Even with the best intentions, the pressure to interoperate can be significant. No implementation can hope to avoid having to trade correctness for interoperability indefinitely.

An implementation that reacts to variations in the manner recommended in the robustness principle sets up a feedback cycle. Over time:

- \* Implementations progressively add logic to constrain how data is transmitted, or to permit variations in what is received.



- \* Errors in implementations or confusion about semantics are permitted or ignored.
- \* These errors can become entrenched, forcing other implementations to be tolerant of those errors.

A flaw can become entrenched as a de facto standard. Any implementation of the protocol is required to replicate the aberrant behavior, or it is not interoperable. This is both a consequence of applying the robustness principle, and a product of a natural reluctance to avoid fatal error conditions. Ensuring interoperability in this environment is often referred to as aiming to be "bug for bug compatible".

For example, in TLS [TLS], extensions use a tag-length-value format and they can be added to messages in any order. However, some server implementations terminated connections if they encountered a TLS ClientHello message that ends with an empty extension. To maintain interoperability, client implementations were required to be aware of this bug and ensure that a ClientHello message ends in a non-empty extension.

The original JSON specification [RFC4627] demonstrates the effect of specification shortcomings: it did not tightly specify some important details including Unicode handling, ordering and duplication of object members, and number encoding. Consequently, a range of interpretations were used by implementations. An updated JSON specification [RFC7159] did not correct these errors, concentrating instead on identifying the interoperable subset of JSON. I-JSON [RFC7493] takes that subset and defines a new format that prohibits the problematic parts of JSON. Of course, that means that I-JSON is not fully interoperable with JSON. Consequently, I-JSON is not widely implemented in parsers. Many JSON parsers now implement the more precise algorithm specified in [ECMA262].

The robustness principle therefore encourages a chain reaction that can create interoperability problems. In particular, the application of the robustness principle is particularly deleterious for early implementations of new protocols as quirks in early implementations can affect all subsequent deployments.

#### 4. Ecosystem Effects

From observing widely deployed protocols, it appears there are two stable points on the spectrum between being strict versus permissive in the presence of protocol errors:

- \* If implementations predominantly enforce strict compliance with specifications, newer implementations will experience failures if they do not comply with protocol requirements. Newer implementations need to fix compliance issues in order to be successfully deployed. This ensures that most deployments are compliant.
- \* Conversely, if non-compliance is tolerated by existing implementations, non-compliant implementations can be deployed successfully. Newer implementations then have strong incentive to tolerate any existing non-compliance in order to be successfully deployed. This ensures that most deployments are tolerant of the same non-compliant behavior.

This happens because interoperability requirements for protocol implementations are set by other deployments. Specifications and - where they exist - conformance test suites might guide the initial development of implementations, but implementations ultimately need to interoperate with deployed implementations.

For widely used protocols, the massive scale of the Internet makes large-scale interoperability testing infeasible for all but a privileged few. The cost of building a new implementation using reverse engineering increases as the number of implementations and bugs increases. Worse, the set of tweaks necessary for wide interoperability can be difficult to discover. In the worst case, a new implementer might have to choose between deployments that have diverged so far as to no longer be interoperable.

Consequently, new implementations might be forced into niche uses, where the problems arising from interoperability issues can be more closely managed. However, restricting new implementations into limited deployments risks causing forks in the protocol. If implementations do not interoperate, little prevents those implementations from diverging more over time.

This has a negative impact on the ecosystem of a protocol. New implementations are key to the continued viability of a protocol. New protocol implementations are also more likely to be developed for new and diverse use cases and are often the origin of features and capabilities that can be of benefit to existing users.

The need to work around interoperability problems also reduces the ability of established implementations to change. An accumulation of mitigations for interoperability issues makes implementations more difficult to maintain and can constrain extensibility (see also the IAB document on the Long-Term Viability of Protocol Extension Mechanisms [RFC9170]).

Sometimes what appear to be interoperability problems are symptomatic of issues in protocol design. A community that is willing to make changes to the protocol, by revising or extending it, makes the protocol better in the process. Applying the robustness principle instead conceals problems, making it harder, or even impossible, to fix them later.

## 5. Active Protocol Maintenance

The robustness principle can be highly effective in safeguarding against flaws in the implementation of a protocol by peers. Especially when a specification remains unchanged for an extended period of time, incentive to be tolerant of errors accumulates over time. Indeed, when faced with divergent interpretations of an immutable specification, the only way for an implementation to remain interoperable is to be tolerant of differences in interpretation and implementation errors.

From this perspective, application of the robustness principle to the implementation of a protocol specification that does not change is logical, even necessary. But that conclusion relies on an assumption that existing specifications and implementations cannot change. Applying the robustness principle in this way disproportionately values short-term gains over the negative effects on future implementations and the protocol as a whole.

For a protocol to have sustained viability, it is necessary for both specifications and implementations to be responsive to changes, in addition to handling new and old problems that might arise over time.

Maintaining specifications so that they closely match deployments ensures that implementations are consistently interoperable and removes needless barriers for new implementations. Maintenance also enables continued improvement of the protocol. New use cases are an indicator that the protocol could be successful [RFC5218].

Protocol designers are strongly encouraged to continue to maintain and evolve protocol specifications beyond their initial inception and definition. This might require the development of revised specifications, extensions, or other supporting material that documents the current state of the protocol. Involvement of those who implement and deploy the protocol is a critical part of this process, as they provide input on their experience with how the protocol is used.

Most interoperability problems do not require revision of protocols or protocol specifications. For instance, the most effective means of dealing with a defective implementation in a peer could be to

email the developer responsible. It is far more efficient in the long term to fix one isolated bug than it is to deal with the consequences of workarounds.

Early implementations of protocols have a stronger obligation to closely follow specifications as their behavior will affect all subsequent implementations. In addition to specifications, later implementations will be guided by what existing deployments accept. Tolerance of errors in early deployments is most likely to result in problems. Protocol specifications might need more frequent revision during early deployments to capture feedback from early rounds of deployment.

Neglect can quickly produce the negative consequences this document describes. Restoring the protocol to a state where it can be maintained involves first discovering the properties of the protocol as it is deployed, rather than the protocol as it was originally documented. This can be difficult and time-consuming, particularly if the protocol has a diverse set of implementations. Such a process was undertaken for HTTP [HTTP] after a period of minimal maintenance. Restoring HTTP specifications to relevance took significant effort.

Maintenance is most effective if it is responsive, which is greatly affected by how rapidly protocol changes can be deployed. For protocol deployments that operate on longer time scales, temporary workarounds following the spirit of the robustness principle might be necessary. For this, improvements in software update mechanisms ensure that the cost of reacting to changes is much lower than it was in the past. Alternatively, if specifications can be updated more readily than deployments, details of the workaround can be documented, including the desired form of the protocols once the need for workarounds no longer exists and plans for removing the workaround.

## 6. Extensibility

Good extensibility [EXT] can make it easier to respond to new use cases or changes in the environment in which the protocol is deployed.

The ability to extend a protocol is sometimes mistaken for an application of the robustness principle. After all, if one party wants to start using a new feature before another party is prepared to receive it, it might be assumed that the receiving party is being tolerant of unexpected inputs.

A well-designed extensibility mechanism establishes clear rules for the handling of things like new messages or parameters. This depends on precisely specifying the handling of malformed or illegal inputs so that implementations behave consistently in all cases that might affect interoperation. If extension mechanisms and error handling are designed and implemented correctly, new protocol features can be deployed with confidence in the understanding of the effect they have on existing implementations.

In contrast, relying on implementations to consistently apply the robustness principle is not a good strategy for extensibility. Using undocumented or accidental features of a protocol as the basis of an extensibility mechanism can be extremely difficult, as is demonstrated by the case study in Appendix A.3 of [EXT].

A protocol could be designed to permit a narrow set of valid inputs, or it could allow a wide range of inputs as a core feature (see for example [HTML]). Specifying and implementing a more flexible protocol is more difficult; allowing less variability is preferable in the absence of strong reasons to be flexible.

## 7. Virtuous Intolerance

A well-specified protocol includes rules for consistent handling of aberrant conditions. This increases the chances that implementations will have consistent and interoperable handling of unusual conditions.

Choosing to generate fatal errors for unspecified conditions instead of attempting error recovery can ensure that faults receive attention. This intolerance can be harnessed to reduce occurrences of aberrant implementations.

Intolerance toward violations of specification improves feedback for new implementations in particular. When a new implementation encounters a peer that is intolerant of an error, it receives strong feedback that allows the problem to be discovered quickly.

To be effective, intolerant implementations need to be sufficiently widely deployed that they are encountered by new implementations with high probability. This could depend on multiple implementations deploying strict checks.

This does not mean that intolerance of errors in early deployments of protocols have the effect of preventing interoperability. On the contrary, when existing implementations follow clearly specified error handling, new implementations or features can be introduced more readily as the effect on existing implementations can be easily predicted; see also Section 6.

Any intolerance also needs to be strongly supported by specifications, otherwise they encourage fracturing of the protocol community or proliferation of workarounds; see Section 8.

Intolerance can be used to motivate compliance with any protocol requirement. For instance, the `INADEQUATE_SECURITY` error code and associated requirements in HTTP/2 [HTTP/2] resulted in improvements in the security of the deployed base.

## 8. Exclusion

Any protocol participant that is affected by changes arising from maintenance might be excluded if they are unwilling or unable to implement or deploy changes that are made to the protocol.

Deliberate exclusion of problematic implementations is an important tool that can ensure that the interoperability of a protocol remains viable. While compatible changes are always preferable to incompatible ones, it is not always possible to produce a design that protects the ability of all current and future protocol participants to interoperate. Developing and deploying changes that risk exclusion of previously interoperating implementations requires some care, but changes to a protocol should not be blocked on the grounds of the risk of exclusion alone.

Exclusion is a direct goal when choosing to be intolerant of errors (see Section 7). Exclusionary actions are employed with the deliberate intent of protecting future interoperability.

Excluding implementations or deployments can lead to a fracturing of the protocol system that could be more harmful than any divergence resulting from following the robustness principle. The IAB document on Uncoordinated Protocol Development Considered Harmful [RFC5704] describes how conflict or competition in the maintenance of protocols can lead to similar problems.

## 9. Security Considerations

Sloppy implementations, lax interpretations of specifications, and uncoordinated extrapolation of requirements to cover gaps in specification can result in security problems. Hiding the consequences of protocol variations encourages the hiding of issues, which can conceal bugs and make them difficult to discover.

The consequences of the problems described in this document are especially acute for any protocol where security depends on agreement about semantics of protocol elements. For instance, use of unsafe security mechanisms, such as weak primitives [MD5] or obsolete mechanisms [SSL3], are good examples of where forcing exclusion (Section 8) can be desirable.

## 10. IANA Considerations

This document has no IANA actions.

## 11. Informative References

- [ECMA262] "ECMAScript(R) 2018 Language Specification", ECMA-262 9th Edition, June 2018, <<https://www.ecma-international.org/publications/standards/Ecma-262.htm>>.
- [EXT] Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, DOI 10.17487/RFC6709, September 2012, <<https://www.rfc-editor.org/rfc/rfc6709>>.
- [HTML] "HTML", WHATWG Living Standard, 8 March 2019, <<https://html.spec.whatwg.org/>>.
- [HTTP] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/rfc/rfc7230>>.
- [HTTP/2] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/rfc/rfc7540>>.
- [MD5] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/rfc/rfc6151>>.

- [RFC0760] Postel, J., "DoD standard Internet Protocol", RFC 760, DOI 10.17487/RFC0760, January 1980, <<https://www.rfc-editor.org/rfc/rfc760>>.
- [RFC1958] Carpenter, B., Ed., "Architectural Principles of the Internet", RFC 1958, DOI 10.17487/RFC1958, June 1996, <<https://www.rfc-editor.org/rfc/rfc1958>>.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <<https://www.rfc-editor.org/rfc/rfc4627>>.
- [RFC5218] Thaler, D. and B. Aboba, "What Makes for a Successful Protocol?", RFC 5218, DOI 10.17487/RFC5218, July 2008, <<https://www.rfc-editor.org/rfc/rfc5218>>.
- [RFC5704] Bryant, S., Ed., Morrow, M., Ed., and IAB, "Uncoordinated Protocol Development Considered Harmful", RFC 5704, DOI 10.17487/RFC5704, November 2009, <<https://www.rfc-editor.org/rfc/rfc5704>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/rfc/rfc7159>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/rfc/rfc7493>>.
- [RFC9170] Thomson, M. and T. Pauly, "Long-Term Viability of Protocol Extension Mechanisms", RFC 9170, DOI 10.17487/RFC9170, December 2021, <<https://www.rfc-editor.org/rfc/rfc9170>>.
- [SSL3] Barnes, R., Thomson, M., Pironti, A., and A. Langley, "Deprecating Secure Sockets Layer Version 3.0", RFC 7568, DOI 10.17487/RFC7568, June 2015, <<https://www.rfc-editor.org/rfc/rfc7568>>.
- [TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/rfc/rfc8446>>.



## Acknowledgments

Constructive feedback on this document has been provided by a surprising number of people including Bernard Aboba, Brian Carpenter, Stuart Cheshire, Mark Nottingham, Russ Housley, Eric Rescorla, Henning Schulzrinne, Robert Sparks, Brian Trammell, and Anne Van Kesteren. Please excuse any omission.

## Authors' Addresses

Martin Thomson  
Mozilla  
Email: [mt@lowentropy.net](mailto:mt@lowentropy.net)

David Schinazi  
Google LLC  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
United States of America  
Email: [dschinazi.ietf@gmail.com](mailto:dschinazi.ietf@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 16 April 2022

M. Thomson  
Mozilla  
T. Pauly  
Apple  
13 October 2021

Long-term Viability of Protocol Extension Mechanisms  
draft-iab-use-it-or-lose-it-04

## Abstract

The ability to change protocols depends on exercising the extension and version negotiation mechanisms that support change. This document explores how regular use of new protocol features can ensure that it remains possible to deploy changes to a protocol. Examples are given where lack of use caused changes to be more difficult or costly.

## Discussion Venues

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

Discussion of this document takes place on the EDM Program mailing list ([edm@iab.org](mailto:edm@iab.org)), which is archived at <https://mailarchive.ietf.org/arch/browse/edm/>.

Source for this draft and an issue tracker can be found at <https://github.com/intarchboard/use-it-or-lose-it>.

## Status of This Memo

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

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

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

This Internet-Draft will expire on 16 April 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Imperfect Implementations Limit Protocol Evolution . . . . .	3
2.1. Good Protocol Design is Not Itself Sufficient . . . . .	4
2.2. Disuse Can Hide Problems . . . . .	5
2.3. Multi-Party Interactions and Middleboxes . . . . .	5
3. Active Use . . . . .	6
3.1. Dependency is Better . . . . .	7
3.2. Version Negotiation . . . . .	7
3.3. Falsifying Active Use . . . . .	8
3.4. Examples of Active Use . . . . .	9
3.5. Restoring Active Use . . . . .	10
4. Complementary Techniques . . . . .	10
4.1. Fewer Extension Points . . . . .	10
4.2. Invariants . . . . .	11
4.3. Limiting Participation . . . . .	11
4.4. Effective Feedback . . . . .	12
5. Security Considerations . . . . .	12
6. IANA Considerations . . . . .	13
7. Informative References . . . . .	13
Appendix A. Examples . . . . .	17
A.1. DNS . . . . .	17
A.2. HTTP . . . . .	18
A.3. IP . . . . .	18
A.4. SNMP . . . . .	19
A.5. TCP . . . . .	19
A.6. TLS . . . . .	19
Acknowledgments . . . . .	20
Authors' Addresses . . . . .	20

## 1. Introduction

A successful protocol [SUCCESS] needs to change in ways that allow it to continue to fulfill the changing needs of its users. New use cases, conditions and constraints on the deployment of a protocol can render a protocol that does not change obsolete.

Usage patterns and requirements for a protocol shift over time. In response, implementations might adjust usage patterns within the constraints of the protocol, the protocol could be extended, or a replacement protocol might be developed. Experience with Internet-scale protocol deployment shows that each option comes with different costs. [TRANSITIONS] examines the problem of protocol evolution more broadly.

An extension point is a mechanism that allows a protocol to be changed or enhanced. This document examines the specific conditions that determine whether protocol maintainers have the ability to design and deploy new or modified protocols via their specified extension points. Section 2 highlights some historical examples of difficulties in transitions to new protocol features. Section 3 argues that ossified protocols are more difficult to update and describes how successful protocols make frequent use of new extensions and code-points. Section 4 outlines several additional strategies that might aid in ensuring that protocol changes remain possible over time.

The experience that informs this document is predominantly at "higher" layers of the network stack, in protocols with limited numbers of participants. Though similar issues are present in many protocols that operate at scale, the tradeoffs involved with applying some of the suggested techniques can be more complex when there are many participants, such as at the network layer or in routing systems.

## 2. Imperfect Implementations Limit Protocol Evolution

It can be extremely difficult to deploy a change to a protocol if implementations with which the new deployment needs to interoperate do not operate predictably. Variation in how new codepoints or extensions are handled can be the result of bugs in implementation or specifications. Unpredictability can manifest as abrupt termination of sessions, errors, crashes, or disappearances of endpoints and timeouts.

The risk of interoperability problems can in turn make it infeasible to deploy certain protocol changes. If deploying a new codepoint or extension makes an implementation less reliable than others, even if only in rare cases, it is far less likely that implementations will adopt the change.

Deploying a change to a protocol could require implementations to fix a substantial proportion of the bugs that the change exposes. This can involve a difficult process that includes identifying the cause of these errors, finding the responsible implementation(s), coordinating a bug fix and release plan, contacting users and/or the operator of affected services, and waiting for the fix to be deployed.

Given the effort involved in fixing problems, the existence of these sorts of bugs can outright prevent the deployment of some types of protocol changes, especially for protocols involving multiple parties or that are considered critical infrastructure (e.g., IP, BGP, DNS, or TLS). It could even be necessary to come up with a new protocol design that uses a different method to achieve the same result.

This document only addresses cases where extensions are not deliberately blocked. Some deployments or implementations apply policies that explicitly prohibit the use of unknown capabilities. This is especially true of functions that seek to make security guarantees, like firewalls.

The set of interoperable features in a protocol is often the subset of its features that have some value to those implementing and deploying the protocol. It is not always the case that future extensibility is in that set.

## 2.1. Good Protocol Design is Not Itself Sufficient

It is often argued that the careful design of a protocol extension point or version negotiation capability is critical to the freedom that it ultimately offers.

RFC 6709 [EXTENSIBILITY] contains a great deal of well-considered advice on designing for extension. It includes the following advice:

This means that, to be useful, a protocol version-negotiation mechanism should be simple enough that it can reasonably be assumed that all the implementers of the first protocol version at least managed to implement the version-negotiation mechanism correctly.

There are a number of protocols for which this has proven to be insufficient in practice. These protocols have imperfect implementations of these mechanisms. Mechanisms that aren't used are the ones that fail most often. The same paragraph from RFC 6709 acknowledges the existence of this problem, but does not offer any remedy:

The nature of protocol version-negotiation mechanisms is that, by definition, they don't get widespread real-world testing until after the base protocol has been deployed for a while, and its deficiencies have become evident.

Indeed, basic interoperability is considered critical early in the deployment of a protocol. A desire to deploy can result in early focus on a reduced feature set, which could result in deferring implementation of version negotiation and extension mechanisms. This leads to these mechanisms being particularly affected by this problem.

## 2.2. Disuse Can Hide Problems

There are many examples of extension points in protocols that have been either completely unused, or their use was so infrequent that they could no longer be relied upon to function correctly.

Appendix A includes examples of disuse in a number of widely deployed Internet protocols.

Even where extension points have multiple valid values, if the set of permitted values does not change over time, there is still a risk that new values are not tolerated by existing implementations. If the set of values for a particular field or the order in which these values appear of a protocol remains fixed over a long period, some implementations might not correctly handle a new value when it is introduced. For example, implementations of TLS broke when new values of the `signature_algorithms` extension were introduced.

## 2.3. Multi-Party Interactions and Middleboxes

One of the key challenges in deploying new features is ensuring compatibility with all actors that could be involved in the protocol. Even the most superficially simple protocols can often involve more actors than is immediately apparent.

The design of extension points needs to consider what actions middleboxes might take in response to a protocol change, as well as the effect those actions could have on the operation of the protocol.

Deployments of protocol extensions also need to consider the impact of the changes on entities beyond protocol participants and middleboxes. Protocol changes can affect the behavior of applications or systems that don't directly interact with the protocol, such as when a protocol change modifies the formatting of data delivered to an application.

### 3. Active Use

The design of a protocol for extensibility and eventual replacement [EXTENSIBILITY] does not guarantee the ability to exercise those options. The set of features that enable future evolution need to be interoperable in the first implementations and deployments of the protocol. Implementation of mechanisms that support evolution is necessary to ensure that they remain available for new uses, and history has shown this occurs almost exclusively through active mechanism use.

Only by using the extension capabilities of a protocol is the availability of that capability assured. "Using" here includes specifying, implementing, and deploying capabilities that rely on the extension capability. Protocols that fail to use a mechanism, or a protocol that only rarely uses a mechanism, could lead to that mechanism being unreliable.

Implementations that routinely see new values are more likely to correctly handle new values. More frequent changes will improve the likelihood that incorrect handling or intolerance is discovered and rectified. The longer an intolerant implementation is deployed, the more difficult it is to correct.

Protocols that routinely add new extensions and code points rarely have trouble adding additional ones, especially when the handling of new versions or extensions are well defined. The definition of mechanisms alone is insufficient; it is the assured implementation and active use of those mechanisms that determines their availability.

What constitutes "active use" can depend greatly on the environment in which a protocol is deployed. The frequency of changes necessary to safeguard some mechanisms might be slow enough to attract ossification in another protocol deployment, while being excessive in others.

### 3.1. Dependency is Better

The easiest way to guarantee that a protocol mechanism is used is to make the handling of it critical to an endpoint participating in that protocol. This means that implementations must rely on both the existence of extension mechanisms and their continued, repeated expansion over time.

For example, the message format in SMTP relies on header fields for most of its functions, including the most basic delivery functions. A deployment of SMTP cannot avoid including an implementation of header field handling. In addition to this, the regularity with which new header fields are defined and used ensures that deployments frequently encounter header fields that they do not yet (and may never) understand. An SMTP implementation therefore needs to be able to both process header fields that it understands and ignore those that it does not.

In this way, implementing the extensibility mechanism is not merely mandated by the specification, it is crucial to the functioning of a protocol deployment. Should an implementation fail to correctly implement the mechanism, that failure would quickly become apparent.

Caution is advised to avoid assuming that building a dependency on an extension mechanism is sufficient to ensure availability of that mechanism in the long term. If the set of possible uses is narrowly constrained and deployments do not change over time, implementations might not see new variations or assume a narrower interpretation of what is possible. Those implementations might still exhibit errors when presented with new variations.

### 3.2. Version Negotiation

As noted in Section 2.1, protocols that provide version negotiation mechanisms might not be able to test that feature until a new version is deployed. One relatively successful design approach has been to use the protocol selection mechanisms built into a lower-layer protocol to select the protocol. This could allow a version negotiation mechanism to benefit from active use of the extension point by other protocols.

For instance, all published versions of IP contain a version number as the four high bits of the first header byte. However, version selection using this field proved to be unsuccessful. Ultimately, successful deployment of IPv6 over Ethernet [RFC2464] required a different EtherType from IPv4. This change took advantage of the already-diverse usage of EtherType.



Other examples of this style of design include Application-Layer Protocol Negotiation ([ALPN]) and HTTP content negotiation (Section 12 of [HTTP]).

This technique relies on the codepoint being usable. For instance, the IP protocol number is known to be unreliable and therefore not suitable [NEW-PROTOCOLS].

### 3.3. Falsifying Active Use

"Grease" was originally defined for TLS [GREASE], but has been adopted by other protocols, such as QUIC [QUIC]. Grease identifies lack of use as an issue (protocol mechanisms "rusting" shut) and proposes reserving values for extensions that have no semantic value attached.

The design in [GREASE] is aimed at the style of negotiation most used in TLS, where one endpoint offers a set of options and the other chooses the one that it most prefers from those that it supports. An endpoint that uses grease randomly offers options - usually just one - from a set of reserved values. These values are guaranteed to never be assigned real meaning, so its peer will never have cause to genuinely select one of these values.

More generally, greasing is used to refer to any attempt to exercise extension points without changing endpoint behavior, other than to encourage participants to tolerate new or varying values of protocol elements.

The principle that grease operates on is that an implementation that is regularly exposed to unknown values is less likely to be intolerant of new values when they appear. This depends largely on the assumption that the difficulty of implementing the extension mechanism correctly is as easy or easier than implementing code to identify and filter out reserved values. Reserving random or unevenly distributed values for this purpose is thought to further discourage special treatment.

Without reserved greasing codepoints, an implementation can use code points from spaces used for private or experimental use if such a range exists. In addition to the risk of triggering participation in an unwanted experiment, this can be less effective. Incorrect implementations might still be able to identify these code points and ignore them.

In addition to advertising bogus capabilities, an endpoint might also selectively disable non-critical protocol elements to test the ability of peers to handle the absence of certain capabilities.

This style of defensive design is limited because it is only superficial. As greasing only mimics active use of an extension point, it only exercises a small part of the mechanisms that support extensibility. More critically, it does not easily translate to all forms of extension points. For instance, HMSV negotiation cannot be greased in this fashion. Other techniques might be necessary for protocols that don't rely on the particular style of exchange that is predominant in TLS.

Grease is deployed with the intent of quickly revealing errors in implementing the mechanisms it safeguards. Though it has been effective at revealing problems in some cases with TLS, the efficacy of greasing isn't proven more generally. Where implementations are able to tolerate a non-zero error rate in their operation, greasing offers a potential option for safeguarding future extensibility. However, this relies on there being a sufficient proportion of participants that are willing to invest the effort and tolerate the risk of interoperability failures.

### 3.4. Examples of Active Use

Header fields in email [SMTP], HTTP [HTTP] and SIP [SIP] all derive from the same basic design, which amounts to a list name/value pairs. There is no evidence of significant barriers to deploying header fields with new names and semantics in email and HTTP as clients and servers generally ignore headers they do not understand or need. The widespread deployment of SIP B2BUAs, which generally do not ignore unknown fields, means that new SIP header fields do not reliably reach peers. This does not necessarily cause interoperability issues in SIP but rather causes features to remain unavailable until the B2BUA is updated. All three protocols are still able to deploy new features reliably, but SIP features are deployed more slowly due to the larger number of active participants that need to support new features.

As another example, the attribute-value pairs (AVPs) in Diameter [DIAMETER] are fundamental to the design of the protocol. Any use of Diameter requires exercising the ability to add new AVPs. This is routinely done without fear that the new feature might not be successfully deployed.

These examples show extension points that are heavily used are also being relatively unaffected by deployment issues preventing addition of new values for new use cases.

These examples show that a good design is not required for success. On the contrary, success is often despite shortcomings in the design. For instance, the shortcomings of HTTP header fields are significant enough that there are ongoing efforts to improve the syntax [HTTP-HEADERS].

### 3.5. Restoring Active Use

With enough effort, active use can be used to restore capabilities.

EDNS [EDNS] was defined to provide extensibility in DNS. Intolerance of the extension in DNS servers resulted in a fallback method being widely deployed (see Section 6.2.2 of [EDNS]). This fallback resulted in EDNS being disabled for affected servers. Over time, greater support for EDNS and increased reliance on it for different features motivated a flag day [DNSFLAGDAY] where the workaround was removed.

The EDNS example shows that effort can be used to restore capabilities. This is in part because EDNS was actively used with most resolvers and servers. It was therefore possible to force a change to ensure that extension capabilities would always be available. However, this required an enormous coordination effort. A small number of incompatible servers and the names they serve also became inaccessible to most clients.

## 4. Complementary Techniques

The protections to protocol evolution that come from active use (Section 3) can be improved through the use of other defensive techniques. The techniques listed here might not prevent ossification on their own, but can make active use more effective.

### 4.1. Fewer Extension Points

A successful protocol will include many potential types of extension. Designing multiple types of extension mechanism, each suited to a specific purpose, might leave some extension points less heavily used than others.

Disuse of a specialized extension point might render it unusable. In contrast, having a smaller number of extension points with wide applicability could improve the use of those extension points. Use of a shared extension point for any purpose can protect rarer or more specialized uses.

Both extensions and core protocol elements use the same extension points in protocols like HTTP [HTTP] and DIAMETER [DIAMETER]; see Section 3.4.

#### 4.2. Invariants

Documenting aspects of the protocol that cannot or will not change as extensions or new versions are added can be a useful exercise. Section 2.2 of [RFC5704] defines invariants as:

Invariants are core properties that are consistent across the network and do not change over extremely long time-scales.

Understanding what aspects of a protocol are invariant can help guide the process of identifying those parts of the protocol that might change. [QUIC-INVARIANTS] and Section 9.3 of [TLS13] are both examples of documented invariants.

As a means of protecting extensibility, a declaration of protocol invariants is useful only to the extent that protocol participants are willing to allow new uses for the protocol. A protocol that declares protocol invariants relies on implementations understanding and respecting those invariants. If active use is not possible for all non-invariant parts of the protocol, greasing (Section 3.3) might be used to improve the chance that invariants are respected.

Protocol invariants need to be clearly and concisely documented. Including examples of aspects of the protocol that are not invariant, such as Appendix A of [QUIC-INVARIANTS], can be used to clarify intent.

#### 4.3. Limiting Participation

Reducing the number of entities that can participate in a protocol or limiting the extent of participation can reduce the number of entities that might affect extensibility. Using TLS or other cryptographic tools can therefore reduce the number of entities that can influence whether new features are usable.

[PATH-SIGNALS] also recommends the use of encryption and integrity protection to limit participation. For example, encryption is used by the QUIC protocol [QUIC] to limit the information that is available to middleboxes and integrity protection prevents modification.

#### 4.4. Effective Feedback

While not a direct means of protecting extensibility mechanisms, feedback systems can be important to discovering problems.

Visibility of errors is critical to the success of techniques like grease (see Section 3.3). The grease design is most effective if a deployment has a means of detecting and reporting errors. Ignoring errors could allow problems to become entrenched.

Feedback on errors is more important during the development and early deployment of a change. It might also be helpful to disable automatic error recovery methods during development.

Automated feedback systems are important for automated systems, or where error recovery is also automated. For instance, connection failures with HTTP alternative services [ALT-SVC] are not permitted to affect the outcome of transactions. An automated feedback system for capturing failures in alternative services is therefore necessary for failures to be detected.

How errors are gathered and reported will depend greatly on the nature of the protocol deployment and the entity that receives the report. For instance, end users, developers, and network operations each have different requirements for how error reports are created, managed, and acted upon.

Automated delivery of error reports can be critical for rectifying deployment errors as early as possible, such as seen in [DMARC] and [SMTP-TLS-Reporting].

#### 5. Security Considerations

Many of the problems identified in this document are not the result of deliberate actions by an adversary, but more the result of mistakes, decisions made without sufficient context, or simple neglect. Problems therefore not the result of opposition by an adversary. In response, the recommended measures generally assume that other protocol participants will not take deliberate action to prevent protocol evolution.

The use of cryptographic techniques to exclude potential participants is the only strong measure that the document recommends. However, authorized protocol peers are most often responsible for the identified problems, which can mean that cryptography is insufficient to exclude them.

The ability to design, implement, and deploy new protocol mechanisms can be critical to security. In particular, it is important to be able to replace cryptographic algorithms over time [AGILITY]. For example, preparing for replacement of weak hash algorithms was made more difficult through misuse [HASH].

## 6. IANA Considerations

This document makes no request of IANA.

## 7. Informative References

- [AGILITY] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <<https://www.rfc-editor.org/rfc/rfc7696>>.
- [ALPN] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/rfc/rfc7301>>.
- [ALT-SVC] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <<https://www.rfc-editor.org/rfc/rfc7838>>.
- [DIAMETER] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <<https://www.rfc-editor.org/rfc/rfc6733>>.
- [DMARC] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <<https://www.rfc-editor.org/rfc/rfc7489>>.
- [DNSFLAGDAY] "DNS Flag Day 2019", May 2019, <<https://dnsflagday.net/2019/>>.
- [EDNS] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/rfc/rfc6891>>.
- [EXT-TCP] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda, "Is it still possible to extend TCP?", Proceedings of the 2011 ACM SIGCOMM

conference on Internet measurement conference - IMC '11,  
DOI 10.1145/2068816.2068834, 2011,  
<<https://doi.org/10.1145/2068816.2068834>>.

[EXTENSIBILITY]

Carpenter, B., Aboba, B., Ed., and S. Cheshire, "Design Considerations for Protocol Extensions", RFC 6709, DOI 10.17487/RFC6709, September 2012, <<https://www.rfc-editor.org/rfc/rfc6709>>.

[GREASE]

Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <<https://www.rfc-editor.org/rfc/rfc8701>>.

[HASH]

Bellovin, S. and E. Rescorla, "Deploying a New Hash Algorithm", Proceedings of NDSS '06 , 2006, <<https://www.cs.columbia.edu/~smb/papers/new-hash.pdf>>.

[HTTP]

Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-semantics-19>>.

[HTTP-HEADERS]

Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <<https://www.rfc-editor.org/rfc/rfc8941>>.

[HTTP11]

Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP/1.1", Work in Progress, Internet-Draft, draft-ietf-httpbis-messaging-19, 12 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-messaging-19>>.

[INTOLERANCE]

Kario, H., "Re: [TLS] Thoughts on Version Intolerance", 20 July 2016, <<https://mailarchive.ietf.org/arch/msg/tls/bOJ2JQc3HjAHFFWCiNTIb0JuMZc>>.

[MPTCP]

Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/rfc/rfc6824>>.

## [MPTCP-HOW-HARD]

Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and M. Handley, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP", April 2012, <<https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/raiciu>>.

## [NEW-PROTOCOLS]

Barik, R., Welzl, M., Fairhurst, G., Elmokashfi, A., Dreibholz, T., and S. Gjessing, "On the usability of transport protocols other than TCP: A home gateway and internet path traversal study", Computer Networks Vol. 173, pp. 107211, DOI 10.1016/j.comnet.2020.107211, May 2020, <<https://doi.org/10.1016/j.comnet.2020.107211>>.

## [PATH-SIGNALS]

Hardie, T., Ed., "Transport Protocol Path Signals", RFC 8558, DOI 10.17487/RFC8558, April 2019, <<https://www.rfc-editor.org/rfc/rfc8558>>.

## [QUIC]

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>>.

## [QUIC-INVARIANTS]

Thomson, M., "Version-Independent Properties of QUIC", RFC 8999, DOI 10.17487/RFC8999, May 2021, <<https://www.rfc-editor.org/rfc/rfc8999>>.

## [RAv4]

Katz, D., "IP Router Alert Option", RFC 2113, DOI 10.17487/RFC2113, February 1997, <<https://www.rfc-editor.org/rfc/rfc2113>>.

## [RAv6]

Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<https://www.rfc-editor.org/rfc/rfc2711>>.

## [RFC0791]

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

## [RFC0988]

Deering, S., "Host extensions for IP multicasting", RFC 988, DOI 10.17487/RFC0988, July 1986, <<https://www.rfc-editor.org/rfc/rfc988>>.



- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/rfc/rfc2464>>.
- [RFC5704] Bryant, S., Ed., Morrow, M., Ed., and IAB, "Uncoordinated Protocol Development Considered Harmful", RFC 5704, DOI 10.17487/RFC5704, November 2009, <<https://www.rfc-editor.org/rfc/rfc5704>>.
- [RRTYPE] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/rfc/rfc3597>>.
- [SIP] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <<https://www.rfc-editor.org/rfc/rfc3261>>.
- [SMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/rfc/rfc5321>>.
- [SMTP-TLS-Reporting] Margolis, D., Brotman, A., Ramakrishnan, B., Jones, J., and M. Risher, "SMTP TLS Reporting", RFC 8460, DOI 10.17487/RFC8460, September 2018, <<https://www.rfc-editor.org/rfc/rfc8460>>.
- [SNI] Langley, A., "Accepting that other SNI name types will never work", 3 March 2016, <[https://mailarchive.ietf.org/arch/msg/tls/1t79gzNIItZd71DwwoaqcQQ\\_4Yxc](https://mailarchive.ietf.org/arch/msg/tls/1t79gzNIItZd71DwwoaqcQQ_4Yxc)>.
- [SNMPv1] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <<https://www.rfc-editor.org/rfc/rfc1157>>.
- [SPF] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <<https://www.rfc-editor.org/rfc/rfc7208>>.
- [SUCCESS] Thaler, D. and B. Aboba, "What Makes for a Successful Protocol?", RFC 5218, DOI 10.17487/RFC5218, July 2008, <<https://www.rfc-editor.org/rfc/rfc5218>>.

- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/rfc/rfc793>>.
- [TFO] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/rfc/rfc7413>>.
- [TLS-EXT] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/rfc/rfc6066>>.
- [TLS12] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/rfc/rfc5246>>.
- [TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/rfc/rfc8446>>.
- [TRANSITIONS] Thaler, D., Ed., "Planning for Protocol Adoption and Subsequent Transitions", RFC 8170, DOI 10.17487/RFC8170, May 2017, <<https://www.rfc-editor.org/rfc/rfc8170>>.

## Appendix A. Examples

This appendix contains a brief study of problems in a range of Internet protocols at different layers of the stack.

### A.1. DNS

Ossified DNS code bases and systems resulted in new Resource Record Codes (RRCodes) being unusable. A new codepoint would take years of coordination between implementations and deployments before it could be relied upon. Consequently, overloading use of the TXT record was used to avoid effort and delays involved, a method used in the Sender Policy Framework [SPF] and other protocols.

It was not until after the standard mechanism for dealing with new RRCodes [RRTYPE] was considered widely deployed that new RRCodes can be safely created and used.

## A.2. HTTP

HTTP has a number of very effective extension points in addition to the aforementioned header fields. It also has some examples of extension points that are so rarely used that it is possible that they are not at all usable.

Extension points in HTTP that might be unwise to use include the extension point on each chunk in the chunked transfer coding Section 7.1 of [HTTP11], the ability to use transfer codings other than the chunked coding, and the range unit in a range request Section 14 of [HTTP].

## A.3. IP

The version field in IP was rendered useless when encapsulated over Ethernet, requiring a new ethertype with IPv6 [RFC2464], due in part to layer 2 devices making version-independent assumptions about the structure of the IPv4 header.

Protocol identifiers or codepoints that are reserved for future use can be especially problematic. Reserving values without attributing semantics to their use can result in diverse or conflicting semantics being attributed without any hope of interoperability. An example of this is the 224/3 "class E" address space in IPv4 [RFC0988]. This space was originally reserved in [RFC0791] without assigning any semantics and has since been partially reclaimed for use in multicast (224/4), but otherwise has not been successfully reclaimed for any purpose (240/4) [RFC0988].

For protocols that can use negotiation to attribute semantics to values, it is possible that unused codepoints can be reclaimed for active use, though this requires that the negotiation include all protocol participants. For something as fundamental as addressing, negotiation is difficult or even impossible, as all nodes on the network path plus potential alternative paths would need to be involved.

IP Router Alerts [RAv4][RAv6] use IP options or extension headers to indicate that data is intended for consumption by the next hop router rather than the addressed destination. In part, the deployment of router alerts was unsuccessful due to the realities of processing IP packets at line rates, combined with bad assumptions in the protocol design about these performance constraints. However, this was not exclusively down to design problems or bugs as the capability was also deliberately blocked at some routers.

#### A.4. SNMP

As a counter example, the first version of the Simple Network Management Protocol (SNMP) [SNMPv1] defines that unparseable or unauthenticated messages are simply discarded without response:

It then verifies the version number of the SNMP message. If there is a mismatch, it discards the datagram and performs no further actions.

When SNMP versions 2, 2c and 3 came along, older agents did exactly what the protocol specifies. Deployment of new versions was likely successful because the handling of newer versions was both clear and simple.

#### A.5. TCP

Extension points in TCP [TCP] have been rendered difficult to use, largely due to middlebox interactions; see [EXT-TCP].

For instance, multipath TCP [MPTCP] can only be deployed opportunistically; see [MPTCP-HOW-HARD]. As multipath TCP enables progressive enhancement of the protocol, this largely only causes the feature to not be available if the path is intolerant of the extension.

In comparison, the deployment of Fast Open [TFO] critically depends on extension capability being widely available. Though very few network paths were intolerant of the extension in absolute terms, TCP Fast Open could not be deployed as a result.

#### A.6. TLS

Transport Layer Security (TLS) [TLS12] provides examples of where a design that is objectively sound fails when incorrectly implemented. TLS provides examples of failures in protocol version negotiation and extensibility.

Version negotiation in TLS 1.2 and earlier uses the "Highest mutually supported version (HMSV)" scheme exactly as it is described in [EXTENSIBILITY]. However, clients are unable to advertise a new version without causing a non-trivial proportion of sessions to fail due to bugs in server and middlebox implementations.

Intolerance to new TLS versions is so severe [INTOLERANCE] that TLS 1.3 [TLS13] abandoned HMSV version negotiation for a new mechanism.

The server name indication (SNI) [TLS-EXT] in TLS is another excellent example of the failure of a well-designed extensibility point. SNI uses the same technique for extension that is used successfully in other parts of the TLS protocol. The original design of SNI anticipated the ability to include multiple names of different types.

SNI was originally defined with just one type of name: a domain name. No other type has ever been standardized, though several have been proposed. Despite an otherwise exemplary design, SNI is so inconsistently implemented that any hope for using the extension point it defines has been abandoned [SNI].

#### Acknowledgments

Toerless Eckert, Wes Hardaker, Mirja Kuehlewind, Eliot Lear, Mark Nottingham, and Brian Trammell made significant contributions to this document.

#### Authors' Addresses

Martin Thomson  
Mozilla

Email: [mt@lowentropy.net](mailto:mt@lowentropy.net)

Tommy Pauly  
Apple

Email: [tpauly@apple.com](mailto:tpauly@apple.com)