Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 14, 2017

M. Thomson
Mozilla

June 12, 2017

The Harmful Consequences of Postel's Maxim
draft-thomson-postel-was-wrong-01

Abstract

Jon Postel's famous statement in RFC 1122 of "Be liberal in what you accept, and conservative in what you send" - is a principle that has long guided the design of Internet protocols and implementations of those protocols. The posture this statement advocates might promote interoperability in the short term, but that short-term advantage is outweighed by negative consequences that affect the long-term maintenance of a protocol and its ecosystem.

Table of Contents

1. Introduction

Of the great many contributions Jon Postel made to the Internet, his remarkable technical achievements are often ignored in favor of the design and implementation philosophy that he first captured in the original IPv4 specification [RFC0760]:

> In general, an implementation should be conservative in its sending behavior, and liberal in its receiving behavior.

In comparison, his contributions to the underpinnings of the Internet, which are in many respects more significant, enjoy less conscious recognition. Postel's principle has been hugely influential in shaping the Internet and the systems that use Internet protocols. Many consider this principle to be instrumental in the success of the Internet as well as the design of interoperable protocols in general.

Over time, considerable changes have occurred in both the scale of the Internet and the level of skill and experience available to protocol and software designers. Much of that experience is with protocols that were designed, informed by Postel's maxim, in the early phases of the Internet.

That experience shows that there are negative long-term consequences to interoperability if an implementation applies Postel's advice. Correcting the problems caused by divergent behavior in implementations can be difficult.

It might be suggested that the posture Postel advocates was indeed necessary during the formative years of the Internet, and even key to its success. This document takes no position on that claim.

This document instead describes the negative consequences of the application of Postel's principle to the modern Internet. A replacement design principle is suggested.

There is good evidence to suggest that designers of protocols in the IETF widely understand the limitations of Postel's principle. This document serves primarily as a record of the shortcomings of His principle for the wider community.

2. Protocol Decay

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 new and unanticipated uses, and ambiguities or errors in the specification are often confounding factors.

Of course, situations where two peers disagree are common, and 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 advised by Postel sets up a feedback cycle:

- Over time, implementations progressively add new code to constrain how data is transmitted, or to permit variations in what is received.

- Errors in implementations, or confusion about semantics can thereby be masked.

- These errors can become entrenched, forcing other implementations to be tolerant of those errors.

For example, the original JSON specification [RFC4627] omitted critical details on a range of points including Unicode handling, ordering and duplication of object members, and number encoding. Consequently, a range of interpretations were used by implementations. An update [RFC7159] was unable to correct these

errors, instead concentrating on defining the interoperable subset of JSON. I-JSON [RFC7493] defines a new format that is substantially similar to JSON without the interoperability flaws. I-JSON also intentionally omits some interoperability: an I-JSON implementation will fail to accept some valid JSON texts. Consequently, most JSON parsers do not implement I-JSON.

An entrenched flaw can become 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 Postel's advice, and a product of a natural reluctance to avoid fatal error conditions. This is colloquially referred to as being "bug for bug compatible".

It is debatable as to whether decay can be completely avoided, but Postel's maxim encourages a reaction that compounds this issue.

3. The Long Term Costs

Once deviations become entrenched, there is little that can be done to rectify the situation.

For widely used protocols, the massive scale of the Internet makes large-scale interoperability testing infeasible for all but a privileged few. Without good maintenance, new implementations can be restricted to niche uses, where the problems arising from interoperability issues can be more closely managed.

This has a negative impact on the ecosystem of a protocol. New implementations are important in ensuring the continued viability of a protocol. New protocol implementations are also more likely to be developed for new and diverse use cases and often are the origin of features and capabilities that can be of benefit to existing users. These problems also reduce the ability of established implementations to change.

Protocol maintenance can help by carefully documenting divergence and recommending limits on what is both acceptable and interoperable. The time-consuming process of documenting the actual protocol - rather than the protocol as it was originally conceived - can restore the ability to create and maintain interoperable implementations.

Such a process was undertaken for HTTP/1.1 [RFC7230]. This effort took more than 6 years to document protocol variations and describe what has - over time - become a far more complex protocol.

4. A New Design Principle

The following principle applies not just to the implementation of a protocol, but to the design and specification of the protocol.

> Protocol designs and implementations should fail noisily in response to bad or undefined inputs.

Though less pithy than Postel's formulation, this principle is based on the lessons of protocol deployment. The principle is also based on valuing early feedback, a practice central to modern engineering discipline.

4.1. Fail Fast and Hard

Protocols need to include error reporting mechanisms that ensure errors are surfaced in a visible and expedient fashion.

Generating fatal errors in place of recovering from a possible fault is preferred, especially if there is any risk that the error represents an implementation flaw. A fatal error provides excellent motivation for addressing problems.

In contrast, generating warnings provide no incentive to fix a problem as the system remains operational. Users can become inured to frequent use of warnings and thus systematically ignore them, whereas a fatal error can only happen once and will demand attention.

On the whole, implementations already have ample motivation to prefer interoperability over correctness. The primary function of a specification is to proscribe behavior in the interest of interoperability. Specifications should mandate fast failure where possible.

4.2. Implementations Are Ultimately Responsible

Implementers are encouraged to expose errors immediately and prominently, especially in cases of underspecification.

Exposing errors is particularly important for early implementations of a protocol. If preexisting implementations generate errors in response to divergent behaviour, then new implementations will be able to detect and correct their own flaws quickly.

An implementer that discovers a scenario that is not covered by the specification does the community a greater service by generating a fatal error than by attempted to interpret and adapt. Hiding errors

can cause long-term problems. Ideally, specification shortcomings are taken to protocol maintainers.

Unreasoning strictness can be detrimental. Protocol designers and implementers expected to exercise judgment in determining what level of strictness is ultimately appropriate. In every case, documenting the decision to deviate from what is specified can avoid later issues.

4.3. Protocol Maintenance is Important

Protocol designers are strongly encouraged to continue to maintain and evolve protocols beyond their initial inception and definition. If protocol implementations are less tolerant of variation, protocol maintenance becomes critical. Good extensibility [RFC6709] can relieve some of the pressure on maintenance.

5. 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.

Designers and implementers of security protocols generally understand these concerns. However, general-purpose protocols are not exempt from careful consideration of security issues. Furthermore, because general-purpose protocols tend to deal with flaws or obsolescence in a less urgent fashion than security protocols, there can be fewer opportunities to correct problems in protocols that develop interoperability problems.

6. IANA Considerations

This document has no IANA actions.

7. Informative References

[RFC0760] Postel, J., "DoD standard Internet Protocol", RFC 760, DOI 10.17487/RFC0760, January 1980, <http://www.rfc-editor.org/info/rfc760>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <http://www.rfc-editor.org/info/rfc4627>.

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

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] 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, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <http://www.rfc-editor.org/info/rfc7493>.

Author's Address

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com