                   What Makes for a Successful Protocol?

Status of This Memo

   This memo provides information for the Internet community. It does
   not specify an Internet standard of any kind. Distribution of this
   memo is unlimited.

Abstract

   The Internet community has specified a large number of protocols to
   date, and these protocols have achieved varying degrees of success.
   Based on case studies, this document attempts to ascertain factors
   that contribute to or hinder a protocol's success. It is hoped that
   these observations can serve as guidance for future protocol work.

Table of Contents

1. Introduction

   One of the goals of the Internet Engineering Task Force (IETF) is to
   define protocols that successfully meet their implementation and
   deployment goals. Based on case studies, this document identifies
   some of the factors influencing success and failure of protocol
   designs. It is hoped that this document will be of use to the
   following audiences:

   o IESG members deciding whether to charter a Working Group to do
      work on a specific protocol;

   o Working Group participants selecting among protocol proposals;

   o Document authors developing a new protocol specification;

   o Anyone evaluating the success of a protocol experiment.

1.1. What is Success?

   In discussing the factors that help or hinder the success of a
   protocol, we need to first define what we mean by "success". A
   protocol can be successful and still not be widely deployed, if it
   meets its original goals. However, in this document, we consider a
   successful protocol to be one that both meets its original goals and
   is widely deployed. Note that "widely deployed" does not mean
   "inter-domain"; successful protocols (e.g., DHCP [RFC2131]) may be
   widely deployed solely for intra-domain use.

   The following are examples of successful protocols:

      Inter-domain: IPv4 [RFC0791], TCP [RFC0793], HTTP [RFC2616], DNS
      [RFC1035], BGP [RFC4271], UDP [RFC0768], SMTP [RFC2821], SIP
      [RFC3261].

      Intra-domain: ARP [RFC0826], PPP [RFC1661], DHCP [RFC2131], RIP
      [RFC1058], OSPF [RFC2328], Kerberos [RFC4120], NAT [RFC3022].

1.2. Success Dimensions

   Two major dimensions on which a protocol can be evaluated are scale
   and purpose. When designed, a protocol is intended for some range of
   purposes and was designed for use on a particular scale.

   Figure 1 graphically depicts these concepts.

```
      Scale ^
            |
            |             +-----------+
            |             |           |
            |             |  Original |
            |             |  Protocol |
            |             |   Design  |
            |             |   Space   |
            |             |           |
            <------------------------------------------------> Purpose
```
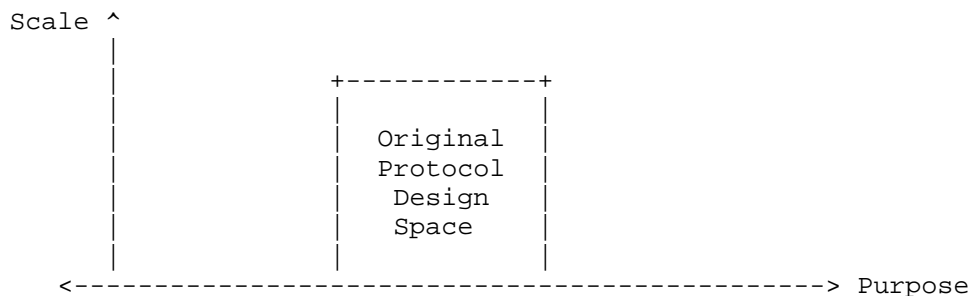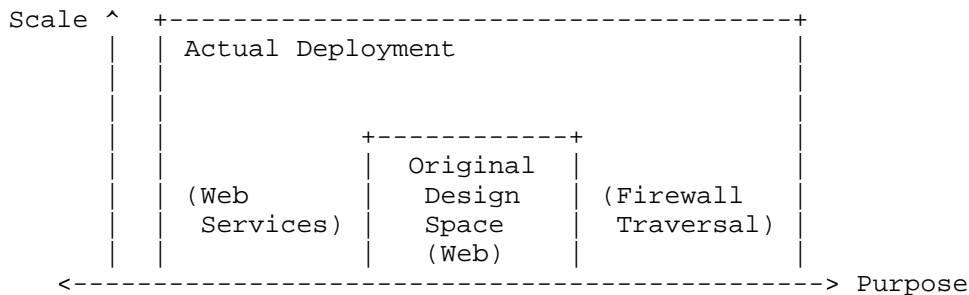
                              Figure 1

   According to these metrics, a "successful" protocol is one that is
   used for its original purpose and at the originally intended scale.
   A "wildly successful" protocol far exceeds its original goals, in
   terms of purpose (being used in scenarios far beyond the initial
   design), in terms of scale (being deployed on a scale much greater
   than originally envisaged), or both. That is, it has overgrown its
   bounds and has ventured out "into the wild".

1.2.1. Examples

   HTTP is an example of a "wildly successful" protocol that exceeded
   its design in both purpose and scale:

```
      Scale ^   +---------------------------------------+
            |   | Actual Deployment                     |
            |   |                                       |
            |   |                                       |
            |   |         +-----------+                 |
            |   |         | Original  |                 |
            |   | (Web    |  Design   | (Firewall   |
            |   |  Services) |  Space  |  Traversal) |
            |   |         |  (Web)    |                 |
            <------------------------------------------------> Purpose
```

   Another example of a wildly successful protocol is IPv4. Although it
   was designed for all purposes ("Everything over IP and IP over
   Everything"), it has been deployed on a far greater scale than that
   for which it was originally designed; the limited address space only
   became an issue after it had already vastly surpassed its original
   design.

   Another example of a successful protocol is ARP. Originally intended
   for a more general purpose (namely, resolving network layer addresses
   to link layer addresses, regardless of the media type or network
   layer protocol), ARP was widely deployed for a narrower scope of uses

(resolution of IPv4 addresses to Ethernet MAC addresses), but then
was adopted for other uses such as detecting network attachment
(Detecting Network Attachment in IPv4 (DNAv4) [RFC4436]). Also, like
IPv4, ARP is deployed on a much greater scale (in terms of number of
machines, but not number on the same subnet) than originally
expected.

```
    Scale ^  +-------------------+
          |  | Actual Deployment |
          |  |                   |
          |  |                   |
          |  |                   |          Original Design Space
          |  |    +-------------+-------------+
          |  |    |(IP/Ethernet)|(Non-IP)     |
          |  |(DNA)|             |             |
          |  |    |             |(Non-Ethernet)|
          |  |    |             |             |
          <---------------------------------------------> Purpose
```
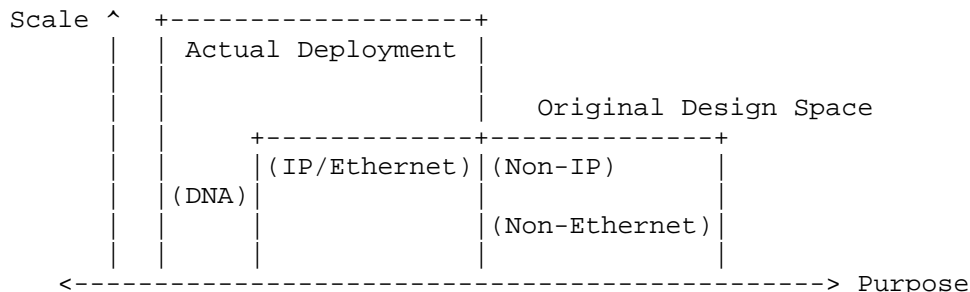
1.3. Effects of Wild Success

   Wild success can be both good and bad. A wildly successful protocol
   is so useful that it can solve more problems or address more
   scenarios or devices. This may indicate that it is time to revise
   the protocol to better accommodate the new design space.

   However, if a protocol is used for a purpose other than what it was
   designed for:

   o There may be undesirable side effects because of design decisions
      that are appropriate for the originally intended purpose, but
      inappropriate for the new purpose.

   o There may be performance problems if the protocol was not designed
      to scale to the extent to which it was deployed.

   o Implementers may attempt to add or change functionality to work
      around the design limitations without complete understanding of
      their effect on the overall protocol behavior and invariants.

   o Wildly successful protocols become high value targets for
      attackers because of their popularity and the potential for
      exploitation of uses or extensions that are less well understood
      and tested than the original protocol.

   A wildly successful protocol is therefore vulnerable to "death by
   success", collapsing as a result of attacks or scaling limitations.

1.4. Failure

   Failure, or the lack of success, cannot be determined before allowing
   sufficient time to pass (e.g., 5-10 years for an average protocol).
   Failure criteria include:

   o No mainstream implementations. There is little or no support in
      hosts, routers, or other classes of relevant devices.

   o No deployment. Devices that support the protocol are not
      deployed, or if they are, then the protocol is not enabled.

   o No use. While the protocol may be deployed, there are no
      applications or scenarios that actually use the protocol.

   At the time a protocol is first designed, the three above conditions
   hold, which is why it is important to allow sufficient time to pass
   before evaluating the success or failure of a protocol.

   The lack of a value chain can make it difficult for a new protocol to
   progress from implementation to deployment to use. While the term
   "chicken-and-egg" problem is sometimes used to describe the lack of a
   value chain, the lack of implementation, deployment, or use is not
   the cause of failure, it is merely a symptom.

   There are many strategies that have been used in the past for
   overcoming the initial lack of implementations, deployment, and use,
   although none of these guarantee success. For example:

   o Address a critical and imminent problem. If the need is severe
      enough, the industry is incented to adopt it as soon as
      implementations exist, and well-known need is sufficient to
      motivate implementations. For example, NAT provided an immediate
      address sharing capability to the individual deploying it
      (Appendix A.8). Thus, when creating a protocol, consider whether
      it can be easily tailored or expanded to directly target a
      critical problem; if it only solves part of the problem, consider
      what would be needed in addition.

   o Provide a "killer app" with low deployment costs. This strategy
      can be used to generate demand where none existed before. See the
      HTTP case study in Appendix A.1 for an example.

   o Provide value for existing unmodified applications. This solves
      the chicken-and-egg problem by ensuring that use exists as soon as
      the protocol is deployed, and therefore, the benefit can be
      realized immediately. See the Wired Equivalent Privacy (WEP) case
      study in Appendix A.6 for an example.

o Reduce complexity and cost by narrowing the intended purpose
   and/or scope to an area where it is easiest to succeed. This may
   allow removing complexity that is not required for the narrow
   purpose. Removing complexity reduces the cost of implementation
   and deployment to where the resulting cost may be very low
   compared to the benefit. For example, link-scoped multicast is
   far more successful than, say, inter-domain multicast (see
   Appendix A.4).

o A government or other entity may provide incentives or
   disincentives that motivate implementation and deployment. For
   example, specific cryptographic algorithms may be mandated. As
   another example, Japan started an economic incentive program to
   generate IPv6 [RFC2460] implementations and deployment.

As we will see, such strategies are often successful because they
directly target the top success factors.

2. Initial Success Factors

   In this section, we identify factors that contribute to success and
   "wild" success.

   Note that a successful protocol will not necessarily include all the
   success factors, and some success factors may be present even in
   failed designs. Nevertheless, experience appears to indicate that
   the presence of success factors seems to improve the probability of
   success.

   The success factors, and their relative importance, were suggested by
   a series of case studies (Appendix A).

2.1. Basic Success Factors

2.1.1. Positive Net Value (Meet a Real Need)

   It is critical to the success of a protocol that the benefits of
   deploying the protocol (monetary or otherwise) outweigh the costs,
   which include:

o Hardware cost: Protocols that don't require hardware changes are
   easier to deploy than those that do. Overlay networks are one way
   to avoid requiring hardware changes. However, often hardware
   updates are required even for protocols whose functionality could
   be provided solely in software. Vendors often implement new

functionality only within later branches of the code tree, which
may only run on new hardware. As a result, the safest way to
avoid hardware upgrade cost is to design for backward
compatibility with both existing hardware and software.

o Operational interference: Protocols that don't require changes to
   other operational processes and tools are easier to deploy than
   ones that do. For example, IPsec [RFC4301] interferes with
   NetFlow [RFC3954] deep packet inspection, which can be important
   to operators.

o Retraining: Protocols that have no configuration, or are very easy
   to configure/manage, are cheaper to deploy.

o Business dependencies: Protocols that don't require changes to a
   business model (whether for implementers or deployers) are easier
   to deploy than ones that do. There are costs associated with
   changing billing and accounting systems and retraining of
   associated personnel, and in addition, the assumptions on which
   the previous business model was based may change. For example,
   some time ago many service providers had business models built
   around dial-up with an assumption that machines were not connected
   all the time; protocols that desired always-on connectivity
   required the business model to change since the networks were not
   optimized for always-on. Similarly, some service providers have
   business models that assume that upstream bandwidth is
   underutilized; peer-to-peer protocols may require this business
   model to change. Finally, many service providers have business
   models based on charging for the amount of bandwidth consumed on
   the link to a customer; multicast protocols interfere with this
   business model since they provide a way for a customer to consume
   less bandwidth on the source link by sending multicast traffic, as
   opposed to paying more to source many unicast streams, without
   having some other mechanism to cover the cost of replication in
   the network (e.g., router CPU, downstream link bandwidth, extra
   management). Multicast protocols also complicate business models
   based on charging the source of traffic based on the amount of
   multicast replication, since the source may not be able to predict
   the cost until a bill is received.

Similarly, there are many types of benefits, including:

o Relieving pain: Protocols that drastically lower costs (monetary
   or otherwise) that exist prior to deploying the protocol are
   easier to show direct benefit from, since they address a burning
   need.

   o Enabling new scenarios: Protocols that enable new capabilities,
      scenarios, or user experiences can provide significant value,
      although the benefit may be harder to realize, as there may be
      more risk involved.

   o Incremental improvements: Protocols that provide incremental
      improvements (e.g., better video quality) generate a small
      benefit, and hence can be successful as long as the cost is small.

   There are at least two example cases of cost/benefits tradeoffs. In
   the first case, even upon initial deployment, the benefit outweighs
   the cost. In the second case, there is an upfront cost that
   outweighs the initial benefit, but the benefit grows over time (e.g.,
   as more nodes or applications support it). The former model is much
   easier to get initial deployment, but over time both can be
   successful. The second model has a danger for the initial
   deployments, that if others don't deploy the protocol then the
   initial deployers have lost value, and so they must take on some risk
   in deploying the protocol.

   Success most easily comes when the natural incentive structure is
   aligned with the deployment requirements. That is, those who are
   required to deploy, manage, or configure something are the same as
   those who gain the most benefit. In summary, it is best if there is
   significant positive net value at each organization where a change is
   required.

2.1.2. Incremental Deployability

   A protocol is incrementally deployable if early adopters gain some
   benefit even though the rest of the Internet does not support the
   protocol. There are several aspects to this.

   Protocols that can be deployed by a single group or team (e.g.,
   intra-domain) have a greater chance of success than those that
   require cooperation across organizations (or, in the worst case
   require a "flag day" where everyone has to change simultaneously).
   For example, protocols that don't require changes to infrastructure
   (e.g., router changes, service provider support, etc.) have a greater
   chance of success than ones that do, since less coordination is
   needed, NAT being a canonical example. Similarly, protocols that
   provide benefit when only one end changes have a greater chance of
   success than ones that require both ends of communication to support
   the protocol.

   Finally, protocol updates that are backward compatible with older
   implementations have a greater chance of success than ones that
   aren't.

2.1.3. Open Code Availability

   Protocols with freely available implementation code have a greater
   chance of success than protocols without. Often, this is more
   important than any technical consideration. For example, it can be
   argued that when deciding between IPv4 and Internetwork Packet
   Exchange (IPX) [IPX], this was the determining factor, even though,
   in many ways, IPX was technically superior to IPv4. Similar
   arguments have been made for the success of RADIUS [RFC2865] over
   TACACS+ [TACACS+]. See Appendix A for further discussion.

2.1.4. Freedom from Usage Restrictions

   Freedom from usage restrictions means that anyone who wishes to
   implement or deploy can do so without legal or financial hindrance.
   Within the IETF, this point often comes up when evaluating between
   technologies, one of which has known Intellectual Property associated
   with it. Often the industry chooses the one with no known
   Intellectual Property, even if it is technically inferior.

2.1.5. Open Specification Availability

   Open specification availability means the protocol specification is
   made available to anyone who wishes to use it. This is true for all
   Internet Drafts and RFCs, and it has contributed to the success of
   protocol specifications developed within or contributed to the IETF.

   The various aspects of this factor include:

   o World-wide distribution: Is the specification accessible from
      anywhere in the world?

   o Unrestricted distribution: Are there no legal restrictions on
      getting the specification?

   o Permanence: Does the specification remain even after the creator
      is gone?

   o Stability: Is there a stable version of the specification that
      does not change?

2.1.6. Open Maintenance Processes

   This factor means that the protocol is maintained by open processes,
   mechanisms exist for public comment on the protocol, and the protocol
   maintenance process allows the participation of all constituencies
   that are affected by the protocol.

2.1.7. Good Technical Design

   This factor means that the protocol follows good design principles
   that lead to ease of implementation and interoperability, such as
   those described in "Architectural Principles of the Internet"
   [RFC1958]. For example, simplicity, modularity, and robustness to
   failures are all key design factors. Similarly, clarity in
   specifications is another aspect of good technical design that
   facilitates interoperability and ease of implementation. However,
   experience shows that good technical design has minimal impact on
   initial success compared with other factors.

2.2. Wild Success Factors

   The following factors do not seem to significantly affect initial
   success, but can affect whether a protocol becomes wildly successful.

2.2.1. Extensible

   Protocols that are extensible are more likely to be wildly successful
   in terms of being used for purposes outside their original design.
   An extensible protocol may carry general purpose payloads/options, or
   may be easy to add a new payload/option type. Such extensibility is
   desirable for protocols that intend to apply to all purposes (like
   IP). However, for protocols designed for a specialized purpose,
   extensibility should be carefully considered before including it.

2.2.2. No Hard Scalability Bound

   Protocols that have no inherent limit near the edge of the originally
   envisioned scale are more likely to be wildly successful in terms of
   scale. For example, IPv4 had no inherent limit near its originally
   envisioned scale; the address space limit was not hit until it was
   already wildly successful in terms of scale. Another type of
   inherent limit would be a performance "knee" that causes a meltdown
   (e.g., a broadcast storm) once a scaling limit is passed.

2.2.3. Threats Sufficiently Mitigated

   The more successful a protocol becomes, the more attractive a target
   it will be. Protocols with security flaws may still become wildly
   successful provided that they are extensible enough to allow the
   flaws to be addressed in subsequent revisions. Examples include
   Secure SHell version 1 (SSHv1) and IEEE 802.11 with WEP. However,
   the combination of security flaws and limited extensibility tends to
   be deadly. For example, some early server-based multiplayer games
   ultimately failed due to insufficient protections against cheating,
   even though they were initially successful.

3. Conclusions

   The case studies described in Appendix A indicate that the most
   important initial success factors are filling a real need and being
   incrementally deployable. When there are competing proposals of
   comparable benefit and deployability, open specifications and code
   become significant success factors. Open source availability is
   initially more important than open specification maintenance.

   In most cases, technical quality was not a primary factor in initial
   success. Indeed, many successful protocols would not pass IESG
   review today. Technically inferior proposals can win if they are
   openly available. Factors that do not seem to be significant in
   determining initial success (but may affect wild success) include
   good design, security, and having an open specification maintenance
   process.

   Many of the case studies concern protocols originally developed
   outside the IETF, which the IETF played a role in improving only
   after initial success was certain. While the IETF focuses on design
   quality, which is not a factor in determining initial protocol
   success, once a protocol succeeds, a good technical design may be key
   to it staying successful, or in dealing with wild success. Allowing
   extensibility in an initial design enables initial shortcomings to be
   addressed.

   Security vulnerabilities do not seem to limit initial success, since
   vulnerabilities often become interesting to attackers only after the
   protocol becomes widely deployed enough to become a useful target.
   Finally, open specification maintenance is not important to initial
   success since many successful protocols were initially developed
   outside the IETF or other standards bodies, and were only
   standardized later.

   In light of our conclusions, we recommend that the following
   questions be asked when evaluating protocol designs:

   o Does the protocol exhibit one or more of the critical initial
      success factors?

   o Are there implementers who are ready to implement the technology
      in ways that are likely to be deployed?

   o Are there customers (especially high-profile customers) who are
      ready to deploy the technology?

   o Are there potential niches where the technology is compelling?

   o If so, can complexity be removed to reduce cost?

   o Is there a potential killer app? Or can the technology work
      underneath existing unmodified applications?

   o Is the protocol sufficiently extensible to allow potential
      deficiencies to be addressed in the future?

   o If it is not known whether the protocol will be successful, should
      the market decide first? Or should the IETF work on multiple
      alternatives and let the market decide among them? Are there
      factors listed in this document that may predict which is more
      likely to succeed?

   In the early stages (e.g., BOFs, design of new protocols), evaluating
   the initial success factors is important in facilitating success.
   Similarly, efforts to revise unsuccessful protocols should evaluate
   whether the initial success factors (or enough of them) were present,
   rather than focusing on wild success, which is not yet a problem.
   For a revision of a successful protocol, on the other hand, focusing
   on the wild success factors is more appropriate.

4. Security Considerations

   This document discusses attributes that affect the success of
   protocols. It has no specific security implications.
   Recommendations on security in protocol design can be found in
   [RFC3552].

5. Informative References

   [IEEE-802.11] IEEE, "Wireless LAN Medium Access Control (MAC) and
                  Physical Layer (PHY) Specifications", ANSI/IEEE
                  Std 802.11, 2007.

   [IMODE]   NTT DoCoMo, "i-mode",
                  <http://www.nttdocomo.com/services/imode/index.html>.

   [IPX]   Novell, "IPX Router Specification", Novell Part
                  Number 107-000029-001, 1992.

   [ISO-8879]   ISO, "Information processing -- Text and office
                  systems -- Standard Generalized Markup Language
                  (SGML)", ISO 8879, 1986.

   [RFC0768]   Postel, J., "User Datagram Protocol", STD 6, RFC 768,
                  August 1980.

   [RFC0791]   Postel, J., "Internet Protocol", STD 5, RFC 791,
                   September 1981.

   [RFC0793]   Postel, J., "Transmission Control Protocol", STD 7,
                   RFC 793, September 1981.

   [RFC0826]   Plummer, D., "Ethernet Address Resolution Protocol: Or
                   converting network protocol addresses to 48.bit
                   Ethernet address for transmission on Ethernet
                   hardware", STD 37, RFC 826, November 1982.

   [RFC0959]   Postel, J. and J. Reynolds, "File Transfer Protocol",
                   STD 9, RFC 959, October 1985.

   [RFC1035]   Mockapetris, P., "Domain names - implementation and
                   specification", STD 13, RFC 1035, November 1987.

   [RFC1058]   Hedrick, C., "Routing Information Protocol", RFC 1058,
                   June 1988.

   [RFC1436]   Anklesaria, F., McCahill, M., Lindner, P., Johnson,
                   D., Torrey, D., and B. Alberti, "The Internet Gopher
                   Protocol (a distributed document search and retrieval
                   protocol)", RFC 1436, March 1993.

   [RFC1661]   Simpson, W., "The Point-to-Point Protocol (PPP)",
                   STD 51, RFC 1661, July 1994.

   [RFC1866]   Berners-Lee, T. and D. Connolly, "Hypertext Markup
                   Language - 2.0", RFC 1866, November 1995.

   [RFC1958]   Carpenter, B., "Architectural Principles of the
                   Internet", RFC 1958, June 1996.

   [RFC2131]   Droms, R., "Dynamic Host Configuration Protocol",
                   RFC 2131, March 1997.

   [RFC2328]   Moy, J., "OSPF Version 2", STD 54, RFC 2328,
                   April 1998.

   [RFC2460]   Deering, S. and R. Hinden, "Internet Protocol, Version
                   6 (IPv6) Specification", RFC 2460, December 1998.

   [RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
                   Masinter, L., Leach, P., and T. Berners-Lee,
                   "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616,
                   June 1999.

   [RFC2821]   Klensin, J., "Simple Mail Transfer Protocol",
                  RFC 2821, April 2001.

   [RFC2865]   Rigney, C., Willens, S., Rubens, A., and W. Simpson,
                  "Remote Authentication Dial In User Service (RADIUS)",
                  RFC 2865, June 2000.

   [RFC3022]   Srisuresh, P. and K. Egevang, "Traditional IP Network
                  Address Translator (Traditional NAT)", RFC 3022,
                  January 2001.

   [RFC3261]   Rosenberg, J., Schulzrinne, H., Camarillo, G.,
                  Johnston, A., Peterson, J., Sparks, R., Handley, M.,
                  and E. Schooler, "SIP: Session Initiation Protocol",
                  RFC 3261, June 2002.

   [RFC3552]   Rescorla, E. and B. Korver, "Guidelines for Writing
                  RFC Text on Security Considerations", BCP 72,
                  RFC 3552, July 2003.

   [RFC3954]   Claise, B., "Cisco Systems NetFlow Services Export
                  Version 9", RFC 3954, October 2004.

   [RFC4120]   Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The
                  Kerberos Network Authentication Service (V5)",
                  RFC 4120, July 2005.

   [RFC4251]   Ylonen, T. and C. Lonvick, "The Secure Shell (SSH)
                  Protocol Architecture", RFC 4251, January 2006.

   [RFC4271]   Rekhter, Y., Li, T., and S. Hares, "A Border Gateway
                  Protocol 4 (BGP-4)", RFC 4271, January 2006.

   [RFC4301]   Kent, S. and K. Seo, "Security Architecture for the
                  Internet Protocol", RFC 4301, December 2005.

   [RFC4436]   Aboba, B., Carlson, J., and S. Cheshire, "Detecting
                  Network Attachment in IPv4 (DNAv4)", RFC 4436,
                  March 2006.

   [RFC4864]   Van de Velde, G., Hain, T., Droms, R., Carpenter, B.,
                  and E. Klein, "Local Network Protection for IPv6",
                  RFC 4864, May 2007.

   [TACACS+]   Carrel, D. and L. Grant, "The TACACS+ Protocol,
                  Version 1.78", Work in Progress, January 1997.

   [WAP]   Open Mobile Alliance, "Wireless Application Protocol
                      Architecture Specification", <http://
                      www.openmobilealliance.org/tech/affiliates/
                      LicenseAgreement.asp?DocName=/wap/
                      wap-210-waparch-20010712-a.pdf>.

Appendix A. Case Studies

   In this Appendix, we include several case studies to illustrate the
   importance of potential success factors. Many other equally good
   case studies could have been included, but, in the interests of
   brevity, only a sampling is included here that is sufficient to
   justify the conclusions in the body of this document.

A.1. HTML/HTTP vs. Gopher and FTP

A.1.1. Initial Success Factors

   Positive net value: HTTP [RFC2616] with HTML [RFC1866] provided
   substantially more value than Gopher [RFC1436] and FTP [RFC0959].
   Among other things, HTML/HTTP provided support for forms, which
   opened the door for commercial uses of the technology. In this
   sense, it enabled new scenarios. Furthermore, it only required
   changes by entities that received benefits; hence, the cost and
   benefits were aligned.

   Incremental deployability: Browsers and servers were incrementally
   deployable, but initial browsers were also backward compatible with
   existing protocols such as FTP and Gopher.

   Open code availability: Server code was open. Client source code was
   initially open to academic use only.

   Restriction-free: Academic use licenses were freely available. HTML
   encumbrance only surfaced later.

   Open specification availability: Yes.

   Open maintenance process: Not at first, but eventually. This
   illustrates that it is not necessary to have an open maintenance
   process at first to achieve success. The maintenance process becomes
   important after initial success.

   Good technical design: Fair. Initially, there was no support for
   graphics, HTML was missing many SGML [ISO-8879] features, and HTTP
   1.0 had issues with congestion control and proxy support. These
   sorts of issues would typically prevent IESG approval today.
   However, they did not prevent the protocol from becoming successful.

A.1.2. Wild Success Factors

   Extensible: Extensibility was excellent along multiple dimensions,
   including HTTP, HTML, graphics, forms, Java, JavaScript, etc.

   No hard scalability bound: Excellent. There was no registration
   process, as there was with Gopher, which allowed it to scale much
   better.

   Threats sufficiently mitigated: No. There was initially no support
   for confidentiality (e.g., protection of credit card numbers), and
   HTTP 1.0 had cleartext passwords in Basic auth.

A.1.3. Discussion

   HTML/HTTP addressed scenarios that no other protocol addressed.
   Since deployment was easy, the protocol quickly took off. Only after
   HTML/HTTP became successful did security become an issue. HTML/
   HTTP's initial success occurred outside the IETF, although HTTP was
   later standardized and refined, addressing some of the limitations.

A.2. IPv4 vs. IPX

A.2.1. Initial Success Factors

   Positive net value: There were initially many competitors, including
   IPX, AppleTalk, NetBEUI, OSI, and DECNet. All of them had positive
   net value. However, NetBEUI and DECNet were not designed for
   internetworking, which limited scalability and eventually stunted
   their growth.

   Incremental deployability: None of the competitors (including IPv4)
   had incremental deployability, although there were few enough nodes
   that a flag day was manageable at the time.

   Open code availability: IPv4 had open code from BSD, whereas IPX did
   not. Many argue that this was the primary reason for IPv4's success.

   Restriction-free: Yes for IPv4; No for IPX.

   Open specification availability: Yes for IPv4; No for IPX.

   Open maintenance process: Yes for IPv4; No for IPX.

   Good technical design: The initial design of IPv4 was fair, but
   arguably IPX was initially better. Improvements to IPv4 such as DHCP
   came much later.

A.2.2. Wild Success Factors

   Extensible: Both IPv4 and IPX were extensible to new transports, new
   link types, and new applications.

   No hard scalability bound: Neither had a hard scalability bound close
   to the design goals. IPX arguably scaled higher before hitting any
   bound.

   Threats sufficiently mitigated: Neither IPv4 nor IPX had threats
   sufficiently mitigated.

A.2.3. Discussion

   Initially, it wasn't clear that IPv4 would win. There were also
   other competitors, such as OSI. However, the Advanced Research
   Projects Agency (ARPA) funded IPv4 implementation on BSD and this
   open source initiative led to many others picking up IPv4, which
   ultimately made a difference in IPv4 succeeding rather than its
   competitors. Even though IPX initially had a technically superior
   design, IPv4 succeeded because of its openness.

A.3. SSH

A.3.1. Initial Success Factors

   Positive net value: SSH [RFC4251] provided greater value than
   competitors. Kerberized telnet required deployment of a Kerberos
   server. IPsec required a public key infrastructure (PKI) or pre-
   shared key authentication. While the benefits were comparable, the
   overall costs of the alternatives were much higher, and they
   potentially required deployment by entities that did not directly
   receive benefit. Hence, unlike the alternatives, the cost and
   benefits of SSH were aligned.

   Incremental deployability: Yes, SSH required SSH clients and servers,
   but did not require a key distribution center (KDC) or credential
   pre-configuration.

   Open code availability: Yes

   Restriction-free: It is unclear whether SSH was truly restriction-
   free or not.

   Open specification availability: Not at first, but eventually.

   Open maintenance process: Not at first, but eventually.

   Good technical design: SSHv1 was fair. It had a number of technical
   issues that were addressed in SSHv2.

A.3.2. Wild Success Factors

   Extensibility: Good. SSH allowed adding new authentication
   mechanisms.

   No hard scalability bound: SSH had excellent scalability properties.

   Threats sufficiently mitigated: No. SSHv1 was vulnerable to man-in-
   the-middle attacks.

A.3.3. Discussion

   The "leap of faith" trust model (accept an untrusted certificate the
   first time you connect) was initially criticized by "experts", but
   was popular with users. It provided vastly more functionality and
   didn't require a KDC and so was easy to deploy. These factors made
   SSH a clear winner.

A.4. Inter-Domain IP Multicast vs. Application Overlays

   We now look at a protocol that has not been successful (i.e., has not
   met its original design goals) after a long period of time has
   passed. Note that this discussion applies only to inter-domain
   multicast, not intra-domain or intra-subnet multicast.

A.4.1. Initial Success Factors

   Positive net value: Unclear. When many receivers of the same stream
   exist, the benefit relieves pain near the sender, and in some cases
   enables new scenarios. However, when few receivers exist, the
   benefits are only incremental improvements when compared with
   multiple streams. While there was positive value in bandwidth
   savings, this was offset by the lack of viable business models, and
   lack of tools. Hence, the costs generally outweighed the benefits.

   Furthermore, the costs are not necessarily aligned with the benefits.
   Inter-domain Multicast requires changes by (at least) applications,
   hosts, and routers. However, it is the applications that get the
   primary benefit. For application layer overlays, on the other hand,
   only the applications need to change, and hence the cost is lower
   (and so are the benefits), and cost and benefits are aligned.

   Incremental deployability: Poor for inter-domain multicast, since it
   required every router in the end-to-end path between a source and any
   receiver to support multicast. This severely limited the

deployability of native multicast. Initially, the strategy was to
use an overlay network (the Multicast Backbone (MBone)) to work
around this. However, the overlay network eventually suffered from
performance problems at high fan-out points, and so adding another
node required more coordination with other organizations to find
someone that was not overloaded and agreed to forward traffic on
behalf of the new node.

Incremental deployability was good for application-layer overlays,
since only the applications need to change. However, benefit only
exists when the sender(s) and receivers both support the overlay
mechanism.

Open code availability: Yes.

Restriction-free: Yes.

Open specification availability: Yes for inter-domain multicast.
Application-layer overlays are not standardized, but left to each
application.

Open maintenance process: Yes for inter-domain multicast.
Application-layer overlays are not standardized, but left to each
application.

Good technical design: This is debatable for inter-domain multicast.
In many respects, the technical design is very efficient. In other
respects, it results in per-connection state in all intermediate
routers, which is questionable at best. Application-layer overlays
do not have the disadvantage, but receive a smaller benefit.

A.4.2. Wild Success Factors

Extensible: Yes.

No hard scalability bound: Inter-domain multicast had scalability
issues in terms of number of groups, and in terms of number of
sources. It scaled extremely well in terms of number of receivers.
Application-layer overlays scale well in all dimensions, except that
they do not scale as well as inter-domain multicast in terms of
bandwidth since they still result in multiple streams over the same
link.

Threats sufficiently mitigated: No for inter-domain-multicast, since
untrusted hosts can create state in intermediate routers along an
entire path. Better for application-layer multicast.

A.4.3. Discussion

   Because the benefits weren't enough to outweigh the costs for
   entities (service providers and application developers) to use it,
   instead the industry has tended to choose application overlays with
   replicated unicast.

A.5. Wireless Application Protocol (WAP)

   The Wireless Application Protocol (WAP) [WAP] is another protocol
   that has not been successful, but is worth comparing against other
   protocols.

A.5.1. Initial Success Factors

   Positive net value: Compared to competitors such as HTTP/TCP/IP, and
   NTT DoCoMo's i-mode [IMODE], the relative value of WAP is unclear.
   It suffered from a poor experience, and a lack of tools.

   Incremental deployability: Poor. WAP required a WAP-to-HTTP proxy in
   the service provider and WAP support in phones; adding a new site
   often required participation by the service provider.

   Open code availability: No.

   Restriction-free: No. WAP has two patents with royalties required.

   Open specification availability: No.

   Open maintenance process: No, there was a US$27000 entrance fee.

   Good technical design: No, a common complaint was that WAP was
   underspecified and hence interoperability was difficult.

A.5.2. Wild Success Factors

   Extensible: Unknown.

   No hard scalability bound: Excellent.

   Threats sufficiently mitigated: Unknown.

A.5.3. Discussion

   There were a number of close competitors to WAP. Incremental
   deployability was easier with the competitors, and the restrictions
   on code and specification access were significant factors that
   hindered its ability to succeed.

A.6. Wired Equivalent Privacy (WEP)

   WEP is a part of the IEEE 802.11 standard [IEEE-802.11], which
   succeeded in being widely deployed in spite of its faults.

A.6.1. Initial Success Factors

   Positive net value: Yes. WEP provided security when there was no
   alternative, and it only required changes by entities that got
   benefit.

   Incremental deployability: Yes. Although one needed to configure
   both the access point and stations, each wireless network could
   independently deploy WEP.

   Open code availability: Essentially no, because of Rivest Cipher 4
   (RC4).

   Restriction-free: No for RC4, but otherwise yes.

   Open specification availability: No for RC4, but otherwise yes.

   Open maintenance process: Yes.

   Good technical design: No, WEP had an inappropriate use of RC4.

A.6.2. Wild Success Factors

   Extensible: IEEE 802.11 was extensible enough to enable development
   of replacements for WEP. However, WEP itself was not extensible.

   No hard scalability bound: No.

   Threats sufficiently mitigated: No.

A.6.3. Discussion

   Even though WEP was not completely open and restriction free, and did
   not have a good technical design, it still became successful because
   it was incrementally deployable and it provided significant value
   when there was no alternative. This again shows that value and
   deployability are more significant success factors than technical
   design or openness, particularly when no alternatives exist.

A.7. RADIUS vs. TACACS+

A.7.1. Initial Success Factors

   Positive net value: Yes for both, and it only required changes by
   entities that got benefit.

   Incremental deployability: Yes for both (just change clients and
   servers).

   Open code availability: Yes for RADIUS; initially no for TACACS+, but
   eventually yes.

   Restriction-free: Yes for RADIUS; unclear for TACACS+.

   Open specification availability: Yes for RADIUS; initially no for
   TACACS+, but eventually yes.

   Open maintenance process: Initially no for RADIUS, but eventually
   yes. No for TACACS+.

   Good technical design: Fair for RADIUS (there was no confidentiality
   support, and no authentication of Access Requests; it had home grown
   ciphersuites based on MD5). Good for TACACS+.

A.7.2. Wild Success Factors

   Extensible: Yes for both.

   No hard scalability bound: Excellent for RADIUS (UDP-based); good for
   TACACS+ (TCP-based).

   Threats sufficiently mitigated: No for RADIUS (no support for
   confidentiality, existing implementations are vulnerable to
   dictionary attacks, use of MD5 now vulnerable to collisions).
   TACACS+ was better since it supported encryption.

A.7.3. Discussion

   Since both provided positive net value and were incrementally
   deployable, those factors were not significant. Even though TACACS+
   had a better technical design in most respects, and eventually
   provided openly available specifications and source code, the fact
   that RADIUS had an open maintenance process as well as openly
   available specifications and source code early on was the determining
   factor. This again shows that having a better technical design is
   less important in determining success than other factors.

A.8. Network Address Translators (NATs)

   Although NAT is not, strictly speaking, a "protocol" per se, but
   rather a "mechanism" or "algorithm", we include a case study since
   there are many mechanisms that only require a single entity to change
   to reap the benefit (TCP congestion control algorithms are another
   example in this class), and it is important to include this class of
   mechanisms in the discussion since it exemplifies a key point in the
   discussion of incremental deployability.

A.8.1. Initial Success Factors

   Positive net value: Yes. NATs provided the ability to connect
   multiple devices when only a limited number of addresses were
   available, and also provided a (limited) security boundary as a side
   effect. Hence, it both relieved pain involved with acquiring
   multiple addresses, as well as enabled new scenarios. Finally, it
   only required deployment by the entity that got the benefit.

   Incremental deployability: Yes. One could deploy a NAT without
   coordinating with anyone else, including a service provider.

   Open code availability: Yes.

   Restriction-free: Yes at first (patents subsequently surfaced).

   Open specification availability: Yes.

   Open maintenance process: Yes.

   Good technical design: Fair. NAT functionality was underspecified,
   leading to unpredictable behavior in general. In addition, NATs
   caused problems for certain classes of applications.

A.8.2. Wild Success Factors

   Extensible: Fair. NATs supported some but not all UDP and TCP
   applications. Adding application layer gateway functionality was
   difficult.

   No hard scalability bound: Good. There is a scalability bound
   (number of ports available), but none near the original design goals.

   Threats sufficiently mitigated: Yes.

A.8.3. Discussion

   The absence of an unambiguous specification was not a hindrance to
   initial success since the test cases weren't well defined; therefore,
   each implementation could decide for itself what scenarios it would
   handle correctly.

   Even with its technical problems, NAT succeeded because of the value
   it provided. Again, this shows that the industry is willing to
   accept technically problematic solutions when there is no alternative
   and the technology is easy to deploy.

   Indeed, NAT became wildly successful by being used for additional
   purposes [RFC4864], and to a large scale including multiple levels of
   NATs in places.

Appendix B. IAB Members at the Time of This Writing

   Loa Andersson
   Leslie Daigle
   Elwyn Davies
   Kevin Fall
   Russ Housley
   Olaf Kolkman
   Barry Leiba
   Kurtis Lindqvist
   Danny McPherson
   David Oran
   Eric Rescorla
   Dave Thaler
   Lixia Zhang

Authors' Addresses

   Dave Thaler
   IAB
   One Microsoft Way
   Redmond, WA 98052
   USA

   Phone: +1 425 703 8835
   EMail: dthaler@microsoft.com

   Bernard Aboba
   IAB
   One Microsoft Way
   Redmond, WA 98052
   USA

   Phone: +1 425 706 6605
   EMail: bernarda@microsoft.com