What makes for a successful protocol?

draft-iab-protocol-success-01.txt

Dave Thaler

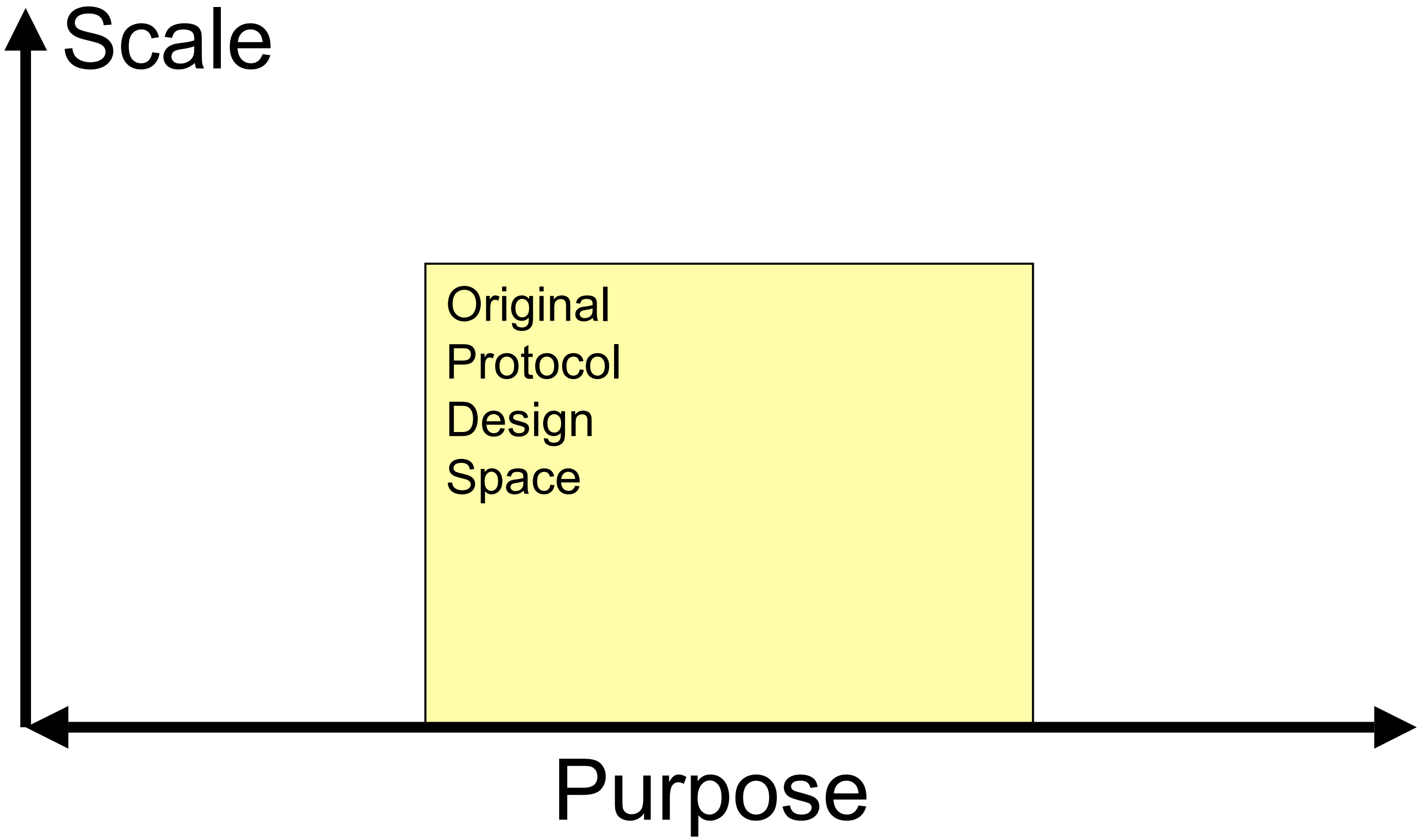
dthaler@microsoft.com

What is success?

- A protocol can be successful and still not be widely deployed, if ***it meets its original goals***
  - However, it's not very interesting to us for this discussion, so let's just look at things that are widely deployed.
  - Widely deployed ≠ inter-domain

- We might consider the following as some examples of successes:
  - Inter-domain: IPv4, TCP, HTTP, DNS, BGP, UDP, SMTP, SIP, etc
  - Intra-domain: ARP, PPP, DHCP, RIP, OSPF, Kerberos, etc

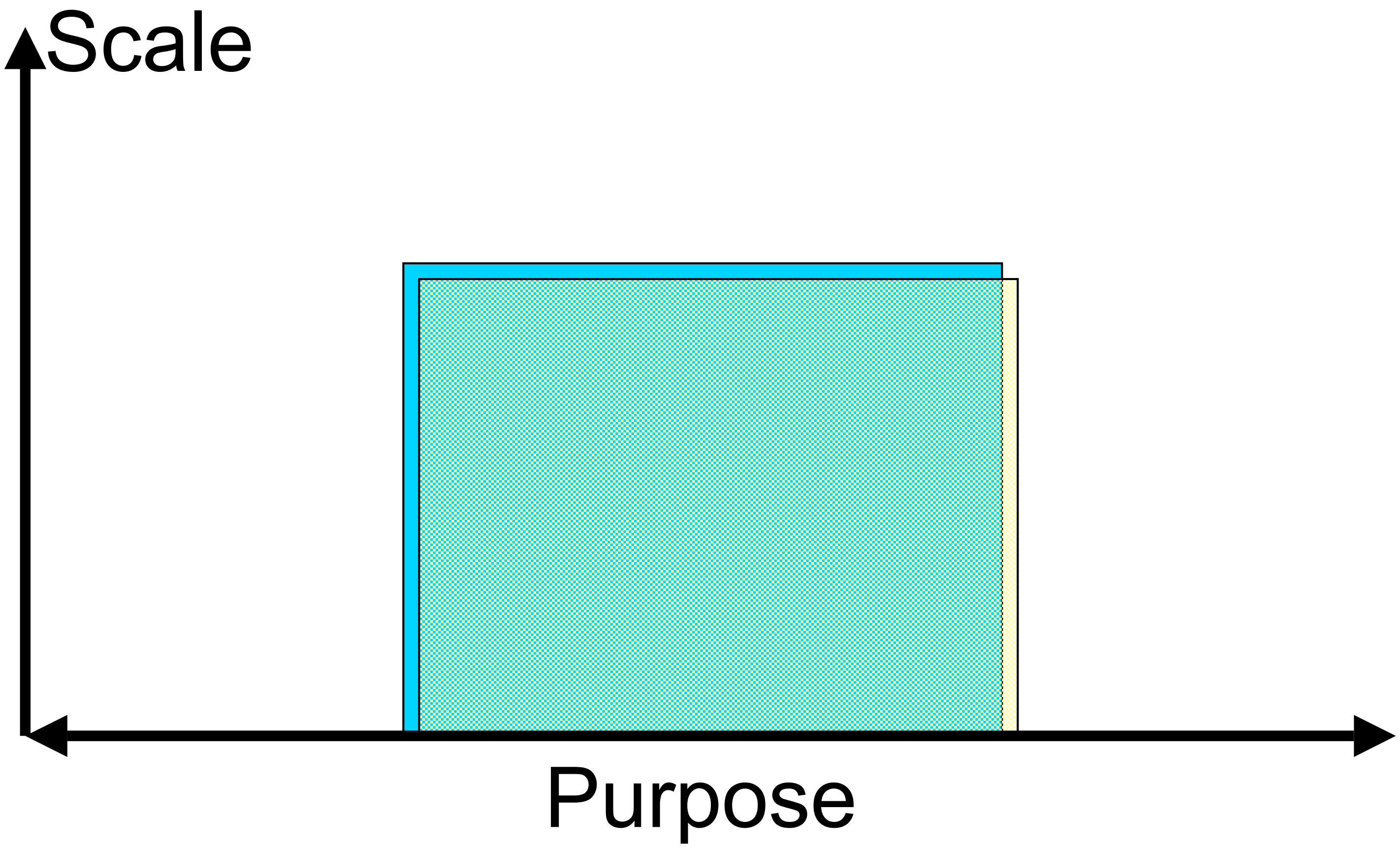
Success Axes



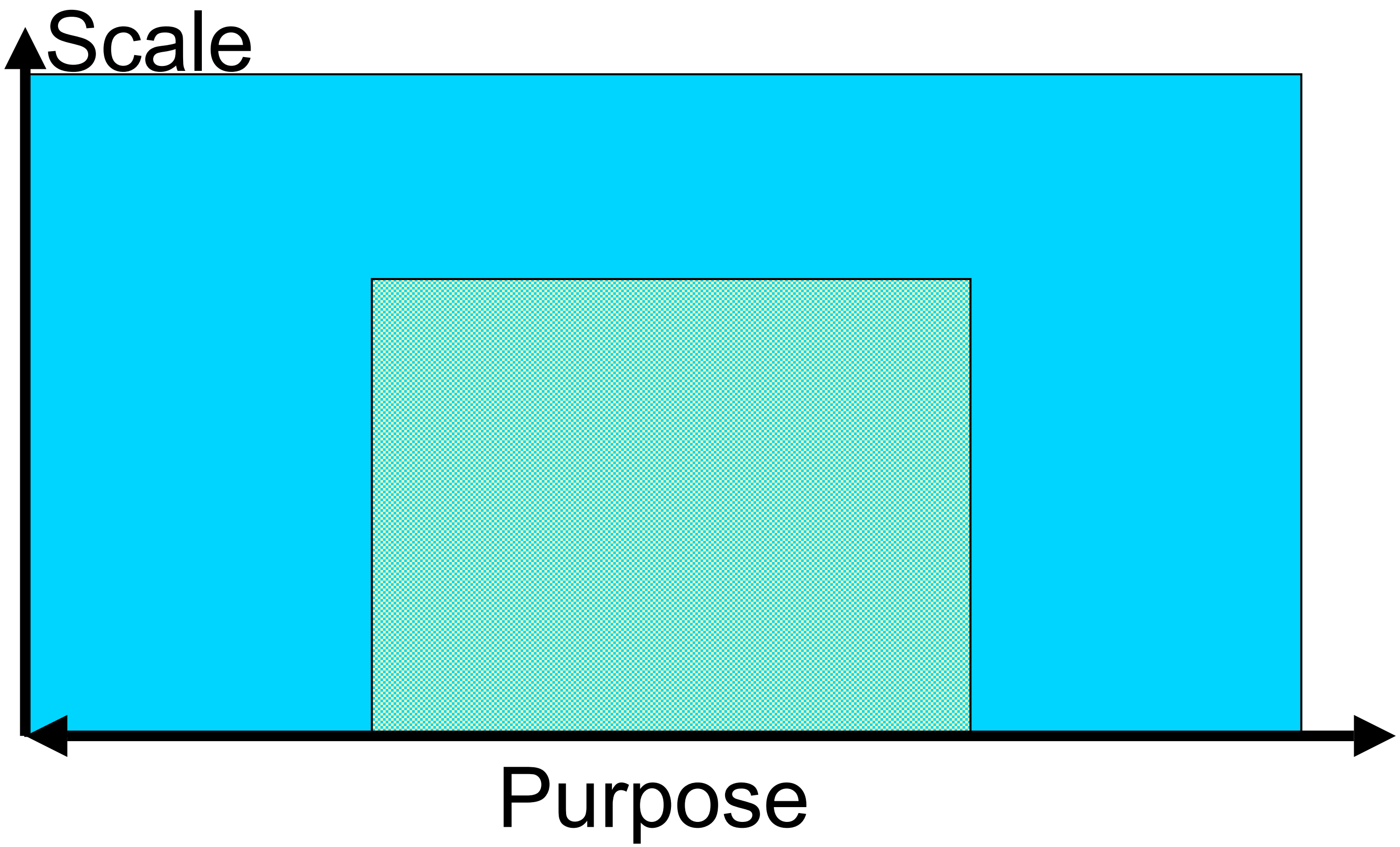
Some Definitions

- "successful": a protocol that is used in the way it was originally envisioned, and to the scale it was originally envisioned
- "wildly successful": a successful protocol that is deployed on a scale much greater than originally envisioned and/or in ways beyond what it was originally designed for.

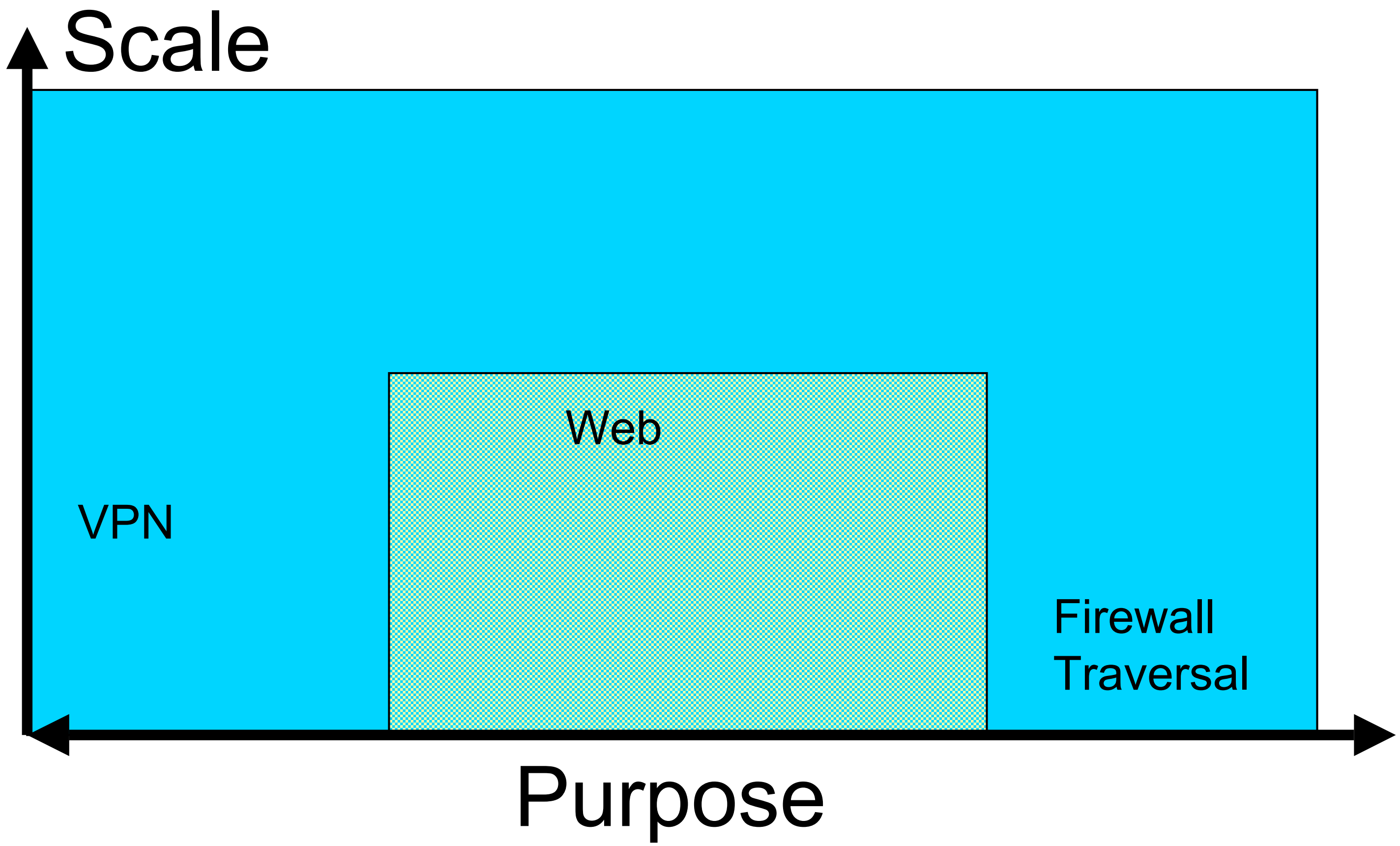
“Successful”



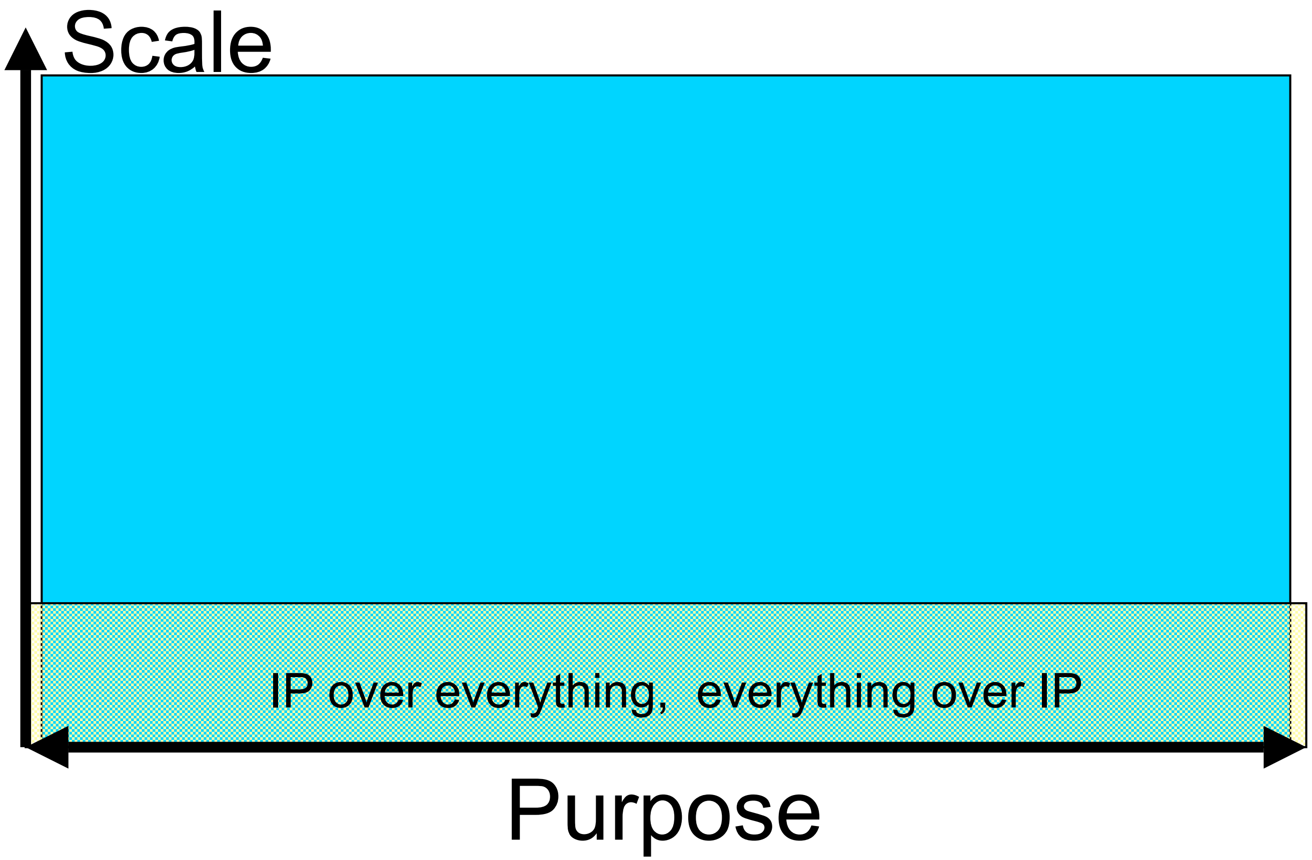
“Wildly Successful”



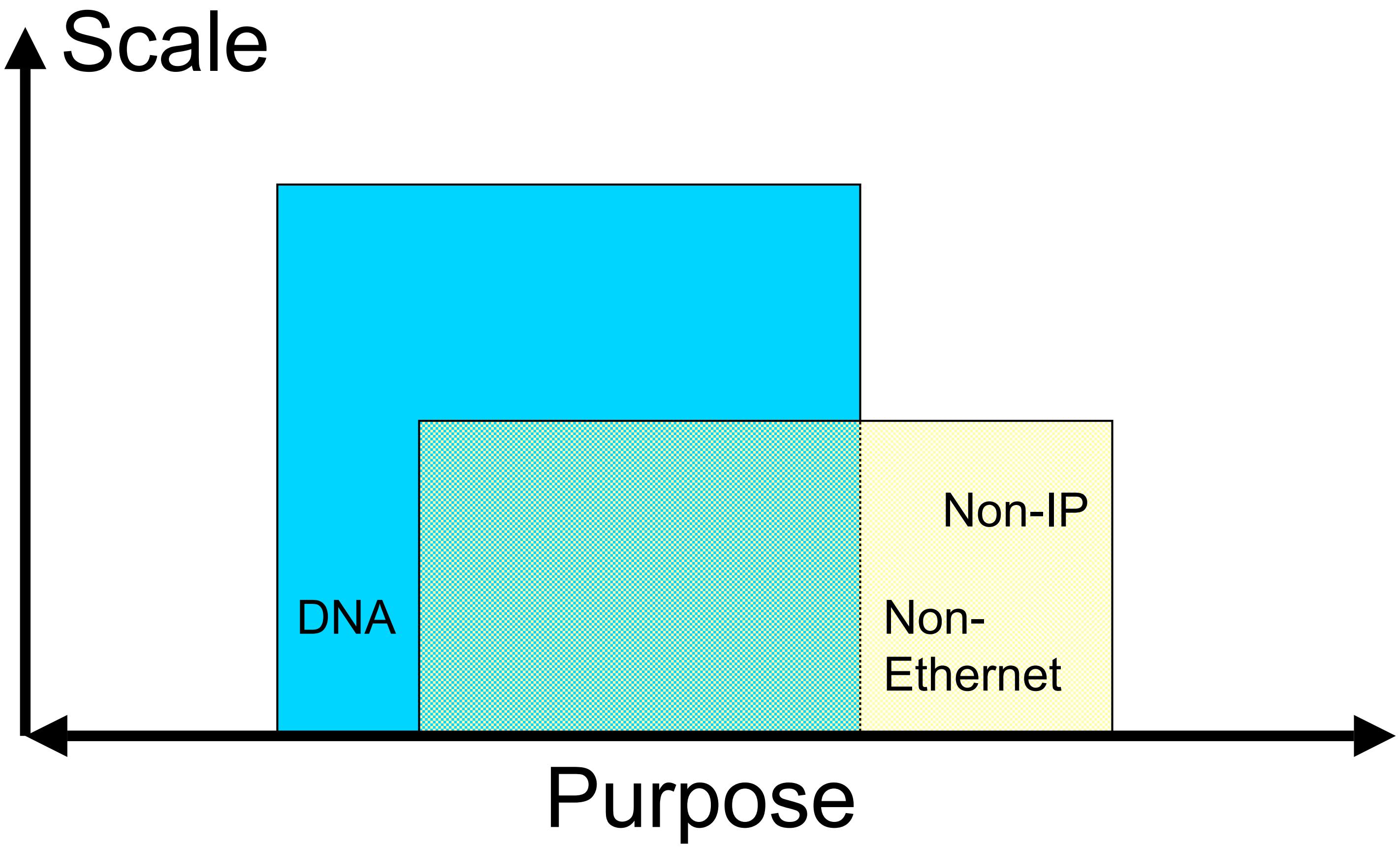
HTTP



IPv4



ARP



Wild success

- Can be both good and bad
  - Undesirable side effects when used outside intended purpose
  - Performance problems
  - Ugly hacks to work around design limitations
  - High value target for attackers
  - “Death by success”

What is failure?

- Sufficient time has passed (e.g. >10 years)
- No mainstream implementations exist
  - No support in hosts/routers/whatever
- No deployment exists
  - Boxes which support it are not deployed, or
  - Protocol is not enabled on boxes that are
- No use exists
  - No applications exist that can utilize

- Cycle between the last three known as the “chicken-and-egg” problem
  - Not a cause of failure, just a term used to explain lack of a value chain in existence

Some ways people try to solve the initial chicken-and-egg problem

1. Address a critical and imminent problem
2. Provide a "killer app" with low deployment costs
3. Provide value under existing unmodified apps
4. Narrow the intended purpose to an area where it is easiest to succeed
   – Reduce cost by removing complexity not required for that purpose
5. Governmental (dis)incentives: promise of long-term economic or military benefits
   – Increase financial benefits (or penalties) to industry
   – E.g. strong crypto for US DoD, IPv6 incentives in Japan, etc.

Success Factors

- What factors contribute to protocol success?
- What additional factors contribute to “wild” success?

- A successful protocol won’t necessarily meet all criteria
  - Each one met may facilitate success
  - Each one not met may hinder success

Potential Success Factors

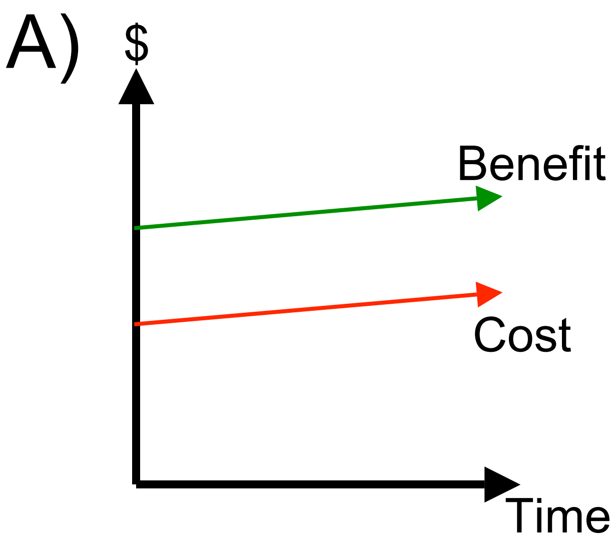
1. Positive net value (meet a real need)
2. Incremental deployability
3. Open code availability
4. Freedom from usage restrictions
5. Open spec availability
6. Open development and maintenance processes
7. Good technical design

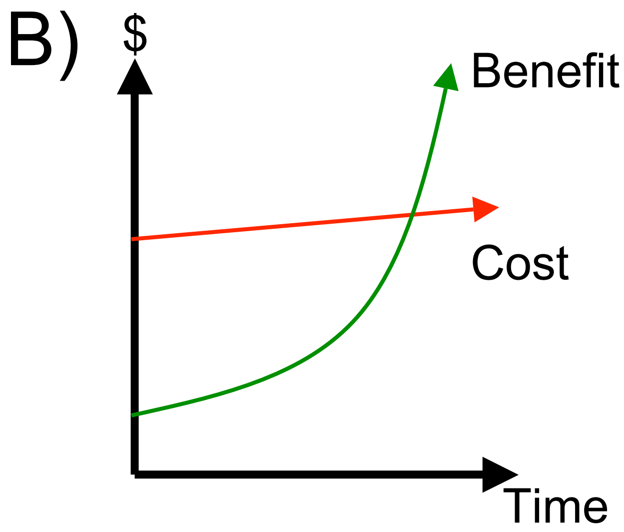
Additional “wild” success factors:

8. Extensible
9. No hard scalability bound
10. Threats sufficiently mitigated

1. Positive net value (1/4)

- The benefits (e.g., monetary) of deploying the protocol clearly outweigh the costs of deploying it.





1. Positive net value (2/4)

- Three types of benefits:
  1. Relieving pain
  2. Enabling new scenarios
     - Often higher risk than type 1
  3. Incremental improvements
     - Often lower gain than type 1

1. Positive net value (3/4)

- Some costs:
  - Hardware cost: protocol changes that don't require changes to hardware are easier to deploy than those that do.
    - Overlays are one way to avoid
  - Operational interference: protocol changes that don’t require changes to other operational processes and tools are easier to deploy than ones that do. (e.g., IPsec interferes with netflow, deep packet inspection, etc.)
    - Overlays are one way to partially mitigate
  - Retraining: protocols that have no configuration, or are easy to configure/manage are easier to deploy

1. Positive net value (4/4)

- Business dependencies: protocols that don't require changes to a business model (whether for implementors or deployers) are easier to deploy than ones that do
  - Dialup and always-on
  - Multicast
  - Provisioning and Peer-to-peer

- Pay to play: The natural incentive structure is aligned with the deployment requirements.
  - Those who are required to deploy/manage/configure something are the same as those who gain the most benefit.
  - That is, <u>there must be positive net value at *each* organization where change is required</u>

2. Incremental deployability

- Early adopters gain some benefit even though the rest of the Internet does not yet support
  - Autonomy: protocols that can be deployed by a single group/team are easier than those that require cooperation across multiple organizations (no flag day)
  - One-end benefit: protocols that provide benefit when only one end changes are easier to deploy than ones that don't (e.g., MIPv6 vs. HIP)
  - Backward compatibility: protocol updates that are backward compatible with legacy implementations are easier to deploy than ones that aren't.

3. Open code availability

- Implementation code freely available
- Often this is more important than technical considerations
  – IPv4 vs IPX
  – RADIUS vs TACACS+

4. Freedom from usage restrictions

- Anyone who wishes to implement or deploy the protocol can do so without legal or financial hindrance.

5. Open spec availability

- The protocol is published and made available in a way that ensures it is accessible to anyone who wishes to use it.
  - World-wide distribution: accessible from anywhere in the world
  - Unrestricted distribution: no legal restrictions on getting spec
  - Permanence: stays even after creator goes away
  - Stable: document doesn’t change
- This is of course true for everything that's an RFC.

6. Open development and maintenance processes

- The protocol is developed and maintained by open processes.
- Mechanisms exist for public commentary on the protocol.
- The protocol maintenance process allows the participation of all constituencies that are affected by the protocol.
- This is of course true for IETF RFCs.

6. Good technical design

- Follows good design principles that lead to ease of implementation, interoperability, etc.
  - Simplicity
  - Modularity
  - Robust to failures

8. Extensible

- Can carry general purpose payloads/options
- Easy to add a new payload/option type

9. No hard scalability bound

- No inherent limit near the edge of the originally envisioned scale
  - Size of “address” fields
  - Performance “knee” that causes meltdown

10. Threats sufficiently mitigated

- The more successful a protocol becomes, the more attacks there will be to mitigate

How Important Are The Success Factors?

- Very Important
  - (Very) positive net value (i.e., Fills a perceived need)
  - Incremental deployability
  - Open code availability
    - Open source availability initially more important than open spec maintenance
  - Open spec availability
    - Technically inferior proposals can win if they are openly available.
  - Restriction free
    - IP did not become a wild success until removal of NSF restrictions.

- Less important for Initial success
  - Open spec maintenance
    - Many successful protocols initially developed outside the IETF
  - Technical design
    - Many successful protocols would not pass IESG review today

- Less important for *Initial* success, but important for *Wild* success
  - Extensibility
  - No hard scalability bound
  - Threats mitigated
    - Security vulnerabilities do not seem to limit initial success

How/when might we apply learnings?

- Focus on initial success factors in early stages:
  - WG charter time (if specific protocol in charter)
  - Protocol selection time (if WG selects among proposals)
  - Protocol creation time
- Focus on wild success factors when revising successful protocols
- Possible questions to ask:
  - Do the success factors exist?
  - Can the technology help potential high-profile customers?
  - Are there potential niches in desperate need?
  - How extensible should the protocol be?
  - If success is uncertain, should IETF wait or work on multiple alternatives?

What is the role of the IETF?

- Most of the success stories are ones which originated outside the IETF, and where technical quality was not a primary factor in success
- IETF had a role in improving many of these, often after success of v1 was certain
- Key is that v1 had to be extensible to allow IETF to fix after success