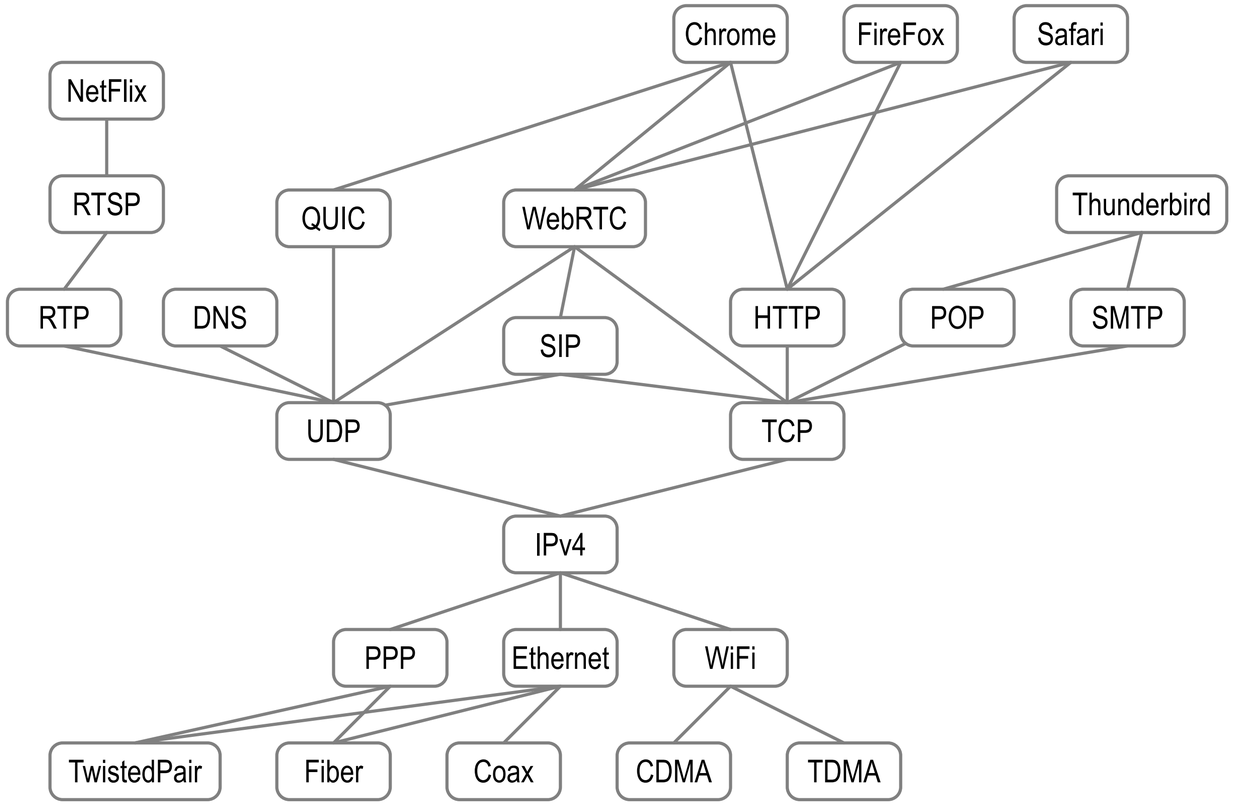
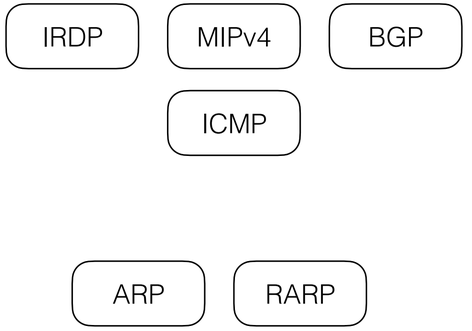
ToIP
Trust Spanning-layer Protocol (TSP) Proposal

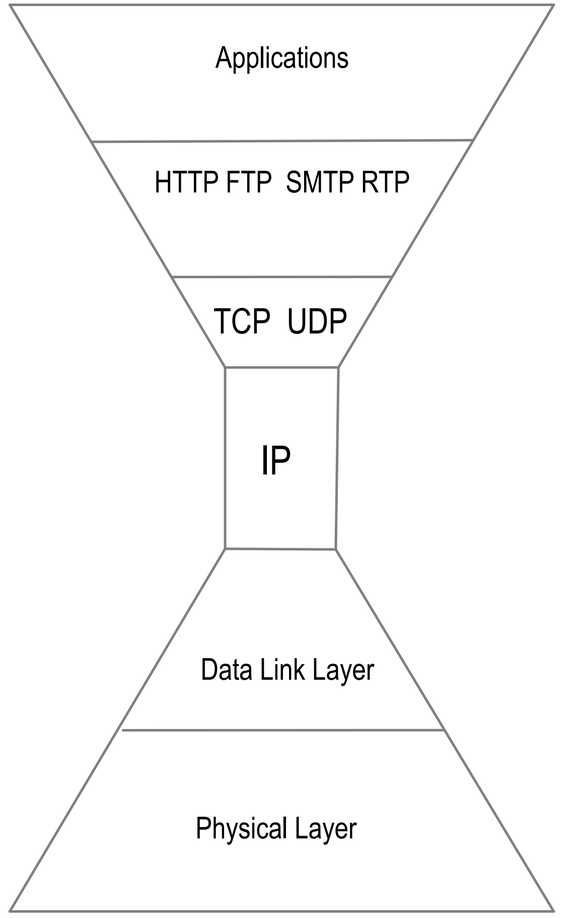
A Secure Identifier Overlay for the Internet

Samuel M. Smith Ph.D.
sam@keri.one
https://keri.one
2023/02/08

Spanning Layer







https://web.archive.org/web/20050415042854/http://www.csd.uch.gr/~hy490-05/lectures/Clark_interoperation.htm

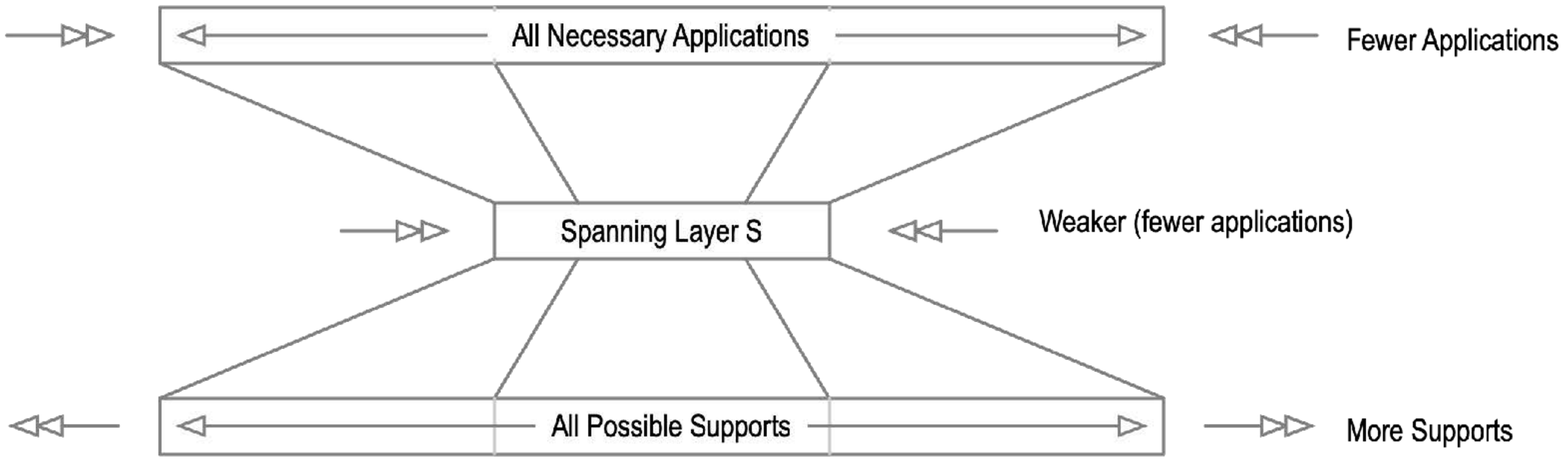
Logical Weakness Property

A weaker spanning layer (less functionality) has fewer supported applications but more possible supports than a stronger spanning layer

Weaker means less functionality.

A stronger spanning layer (more functionality) has fewer possible supports but more supported applications

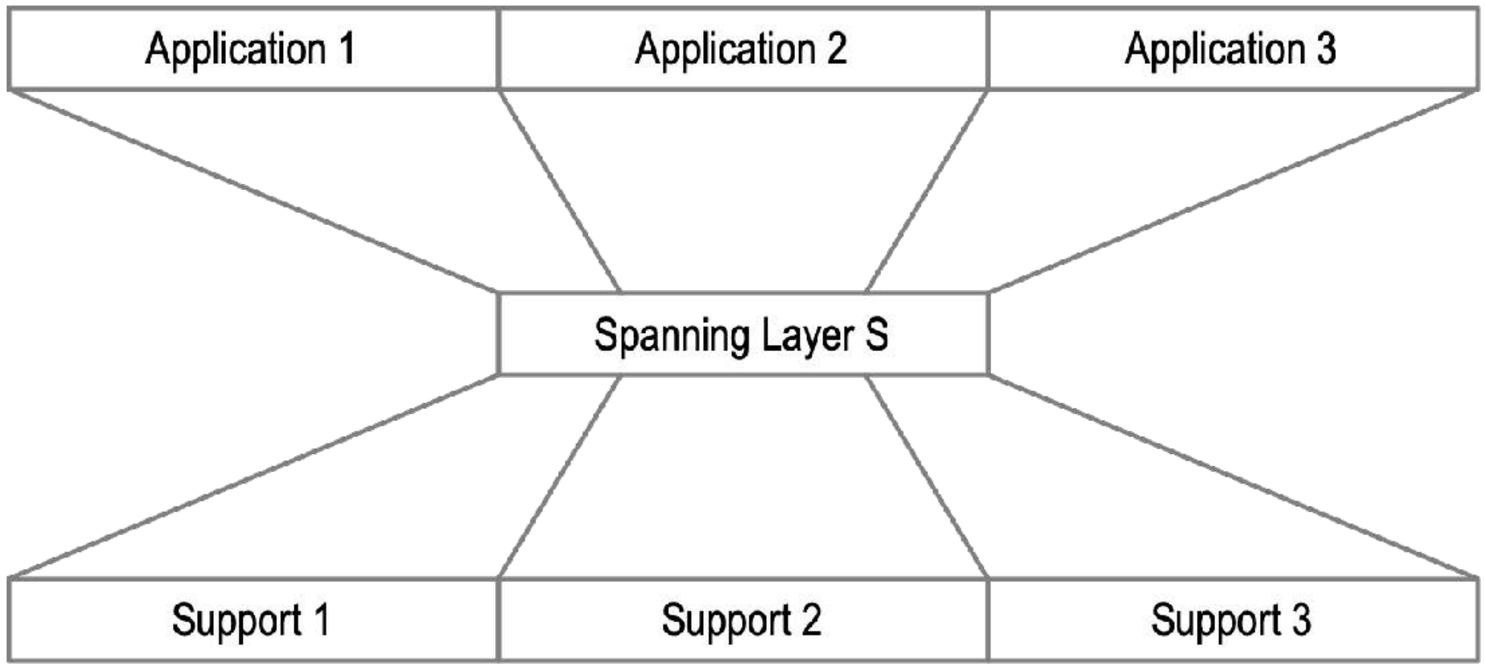
A minimally sufficient spanning layer is the weakest possible layer that still supports the necessary applications



https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/fulltext

Features of a Spanning Layer

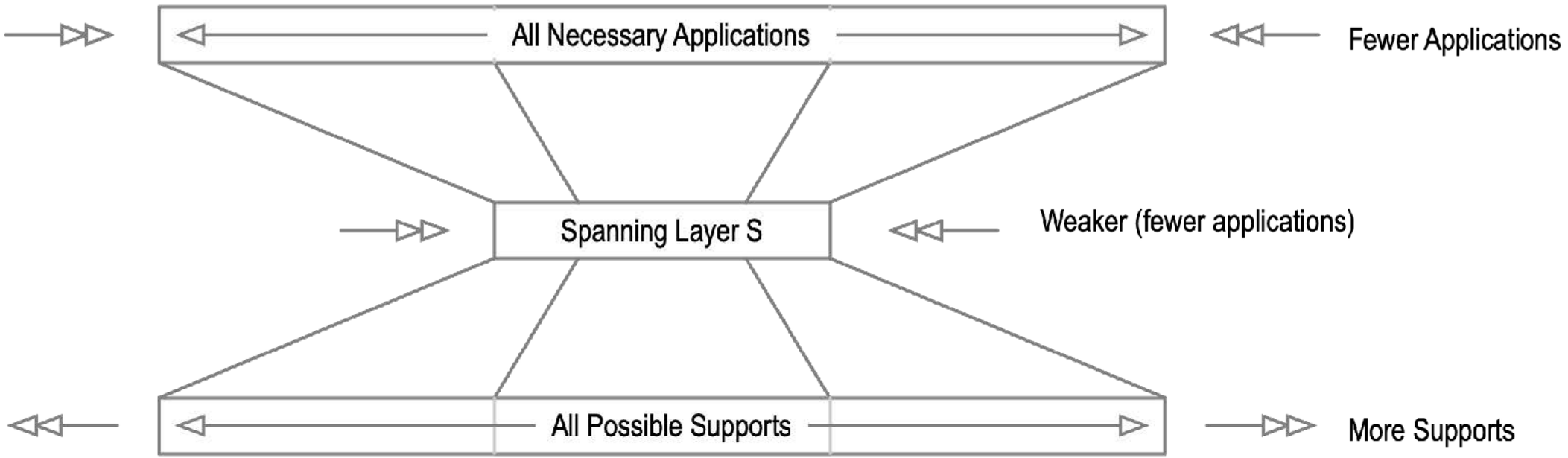
Simplicity: Only one way to access (orthogonality) any service or resource through the spanning layer
Generality: The spanning layer has the richest set of possible applications without increasing its logical strength
Resource Limiting: The spanning layer abstracts and limits resource access by supported applications
Deployment Scalability: widespread adoption though minimally essential functionality



https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/fulltext

Implications of the Hourglass Theorem

Deployment scalability is enabled by a spanning layer that has implementations using as many different supports as possible
Broader adoption comes from increasing the support
A minimally sufficient spanning layer may not be defensibly monetizable because monetization disincentivizes broader support



Insights

Adoption of a common service interface is a key to interoperability, portability, and “future proofing” in the face of rapid technological change.

The design of both the internet and Unix followed the hourglass principle leading to dominance while still allowing disruptive innovation.

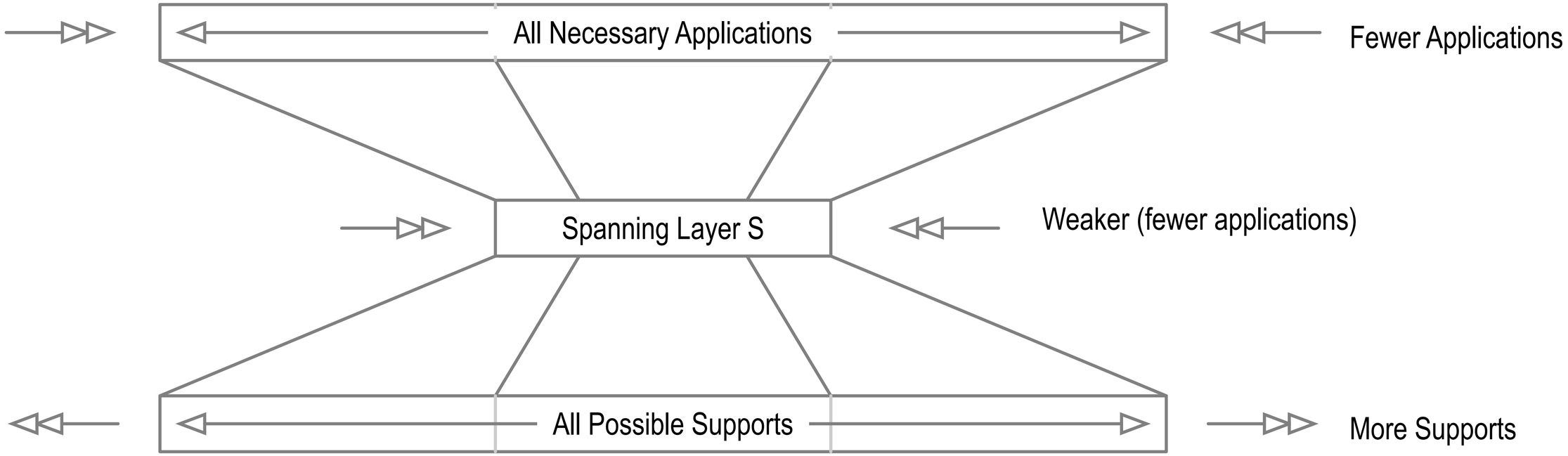
Successful interface design adheres to the discipline of simplicity, generality, and resource limitation of the common interface

The deployment scalability trade-off is that broader adoption comes from thinning or weakening the spanning layer

KERI design guidance from hourglass principle: address only = namespace agnostic, portable, primary root-of-trust is portable verifiable data structure, secondary roots-of-trust are replaceable

Hourglass Theorem Summary

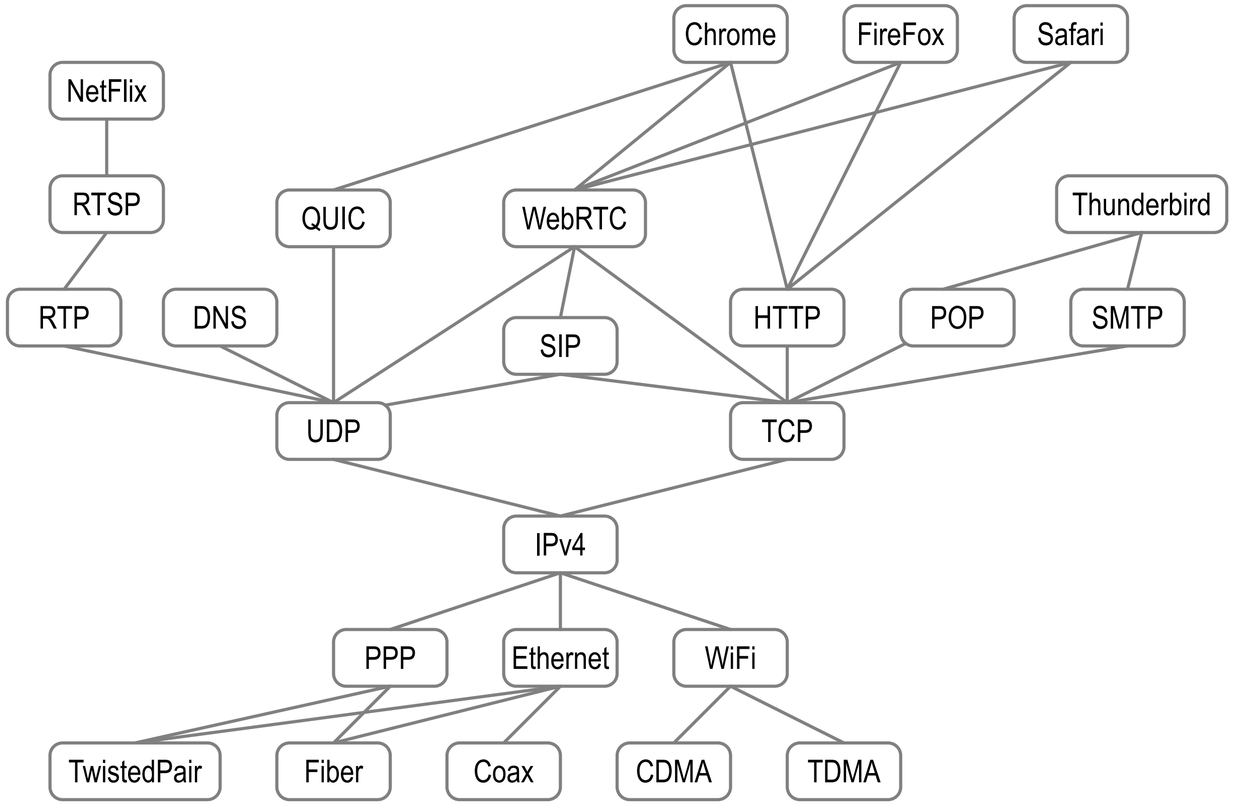
https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/fulltext

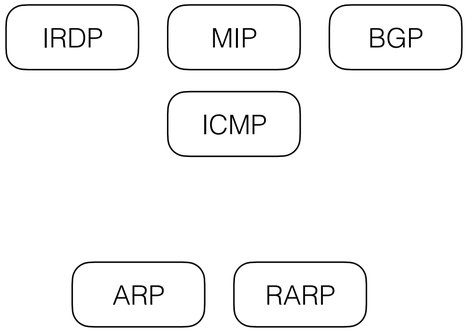


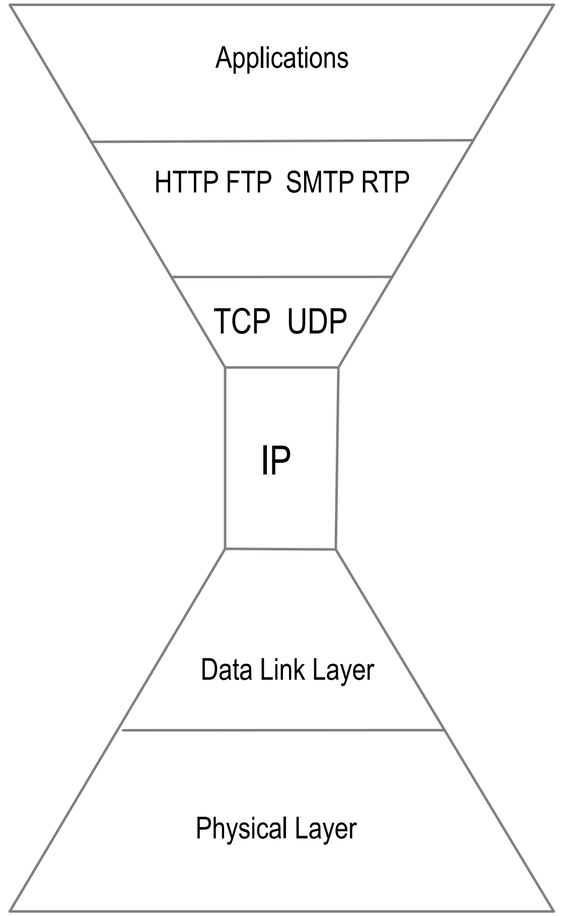
Minimally sufficient spanning layer singularly spans supporting protocols below it and applying protocols above it.

The degree of dominance of a spanning layer is eco-system validation of the preferred design

Spanning Layer

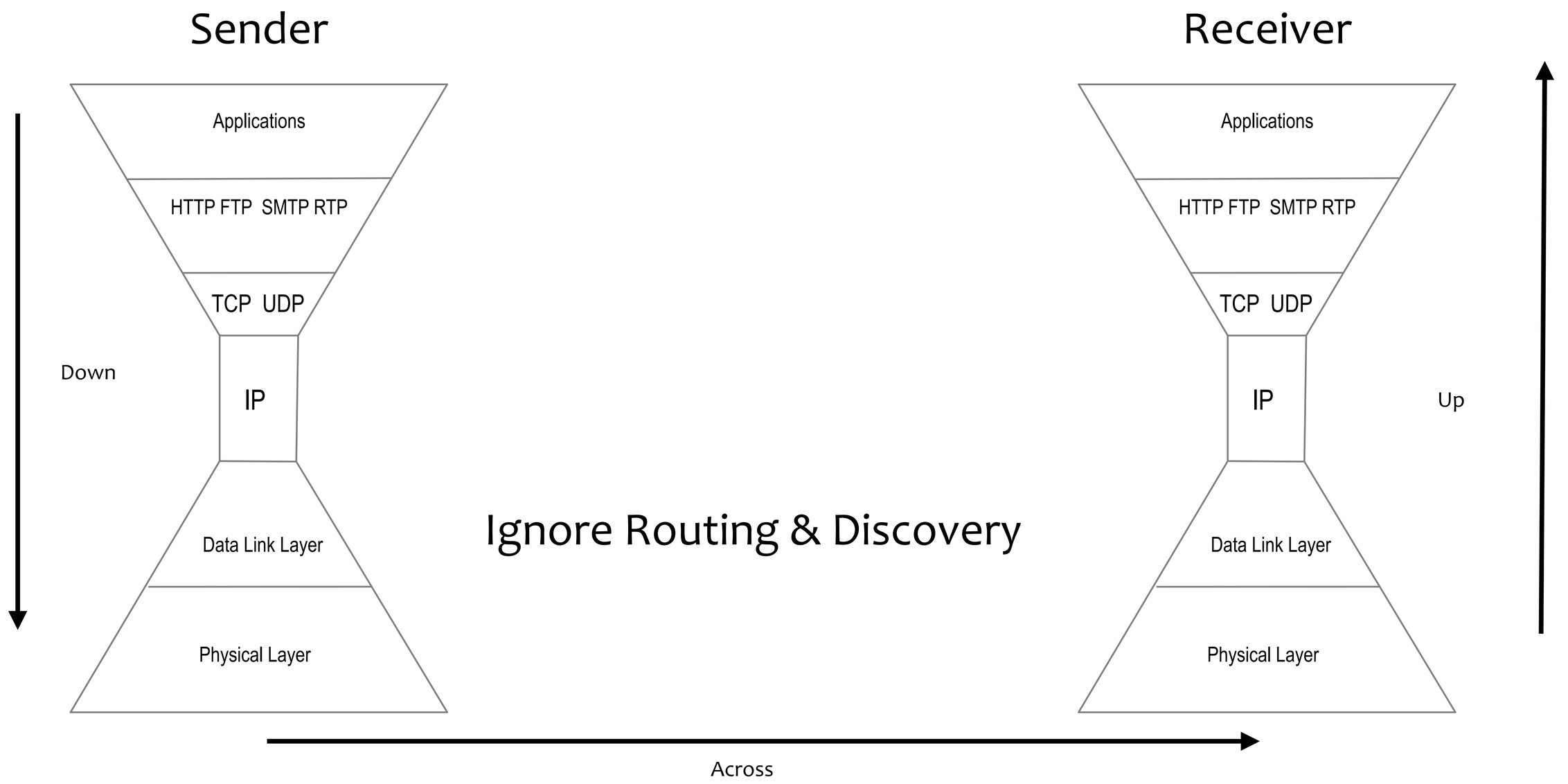




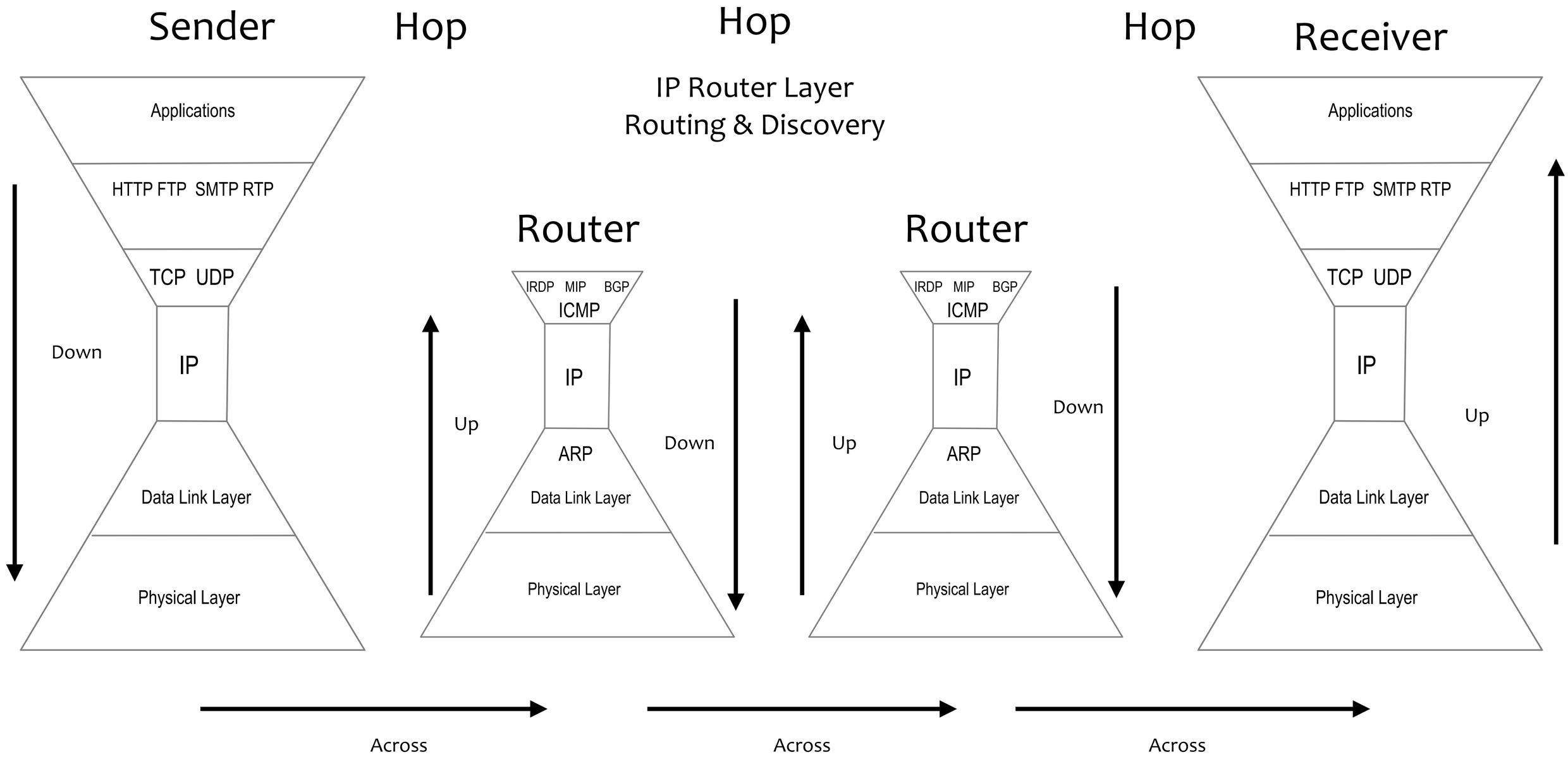


https://web.archive.org/web/20050415042854/http://www.csd.uch.gr/~hy490-05/lectures/Clark_interoperation.htm

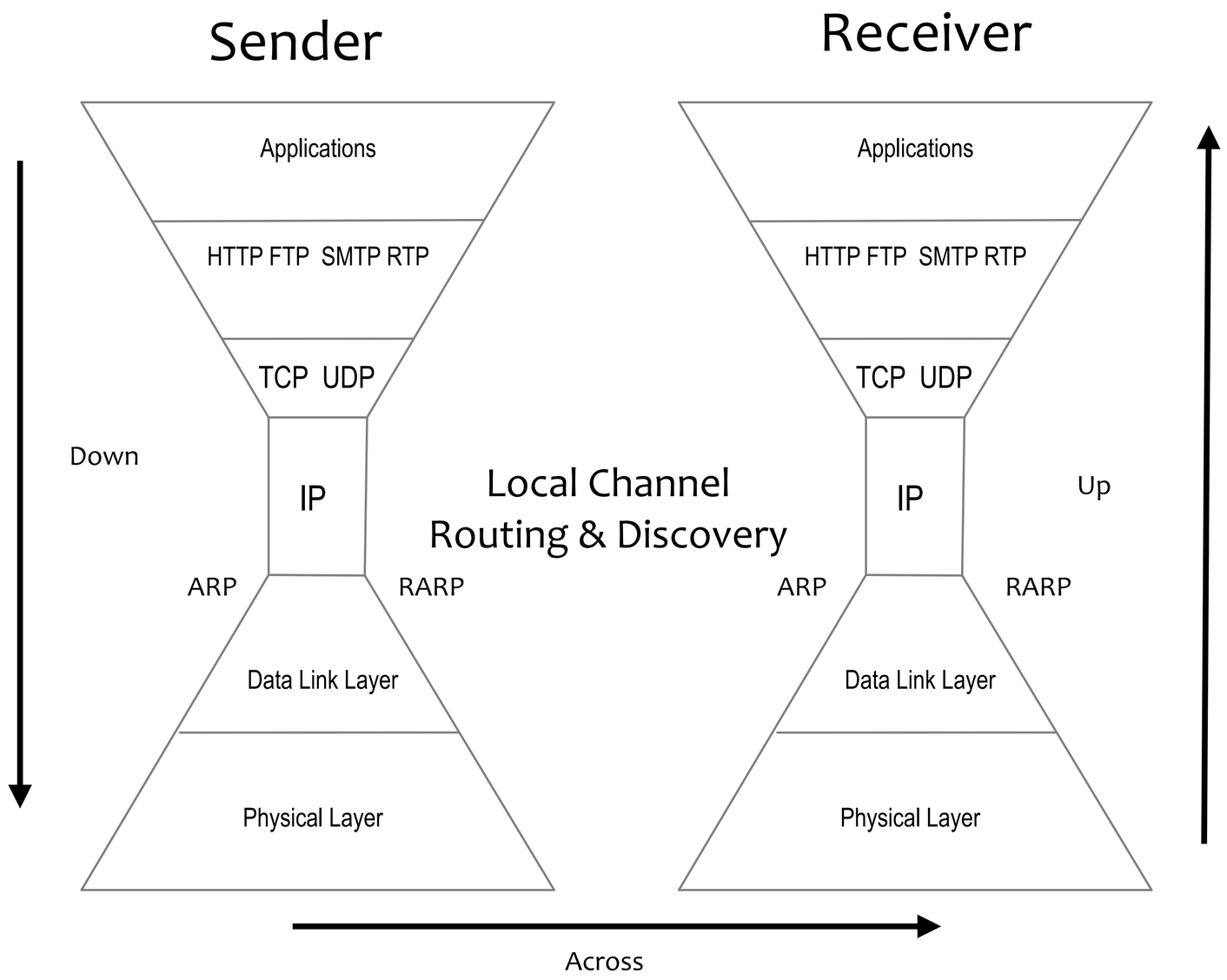
Application Layer Endpoint Spanning



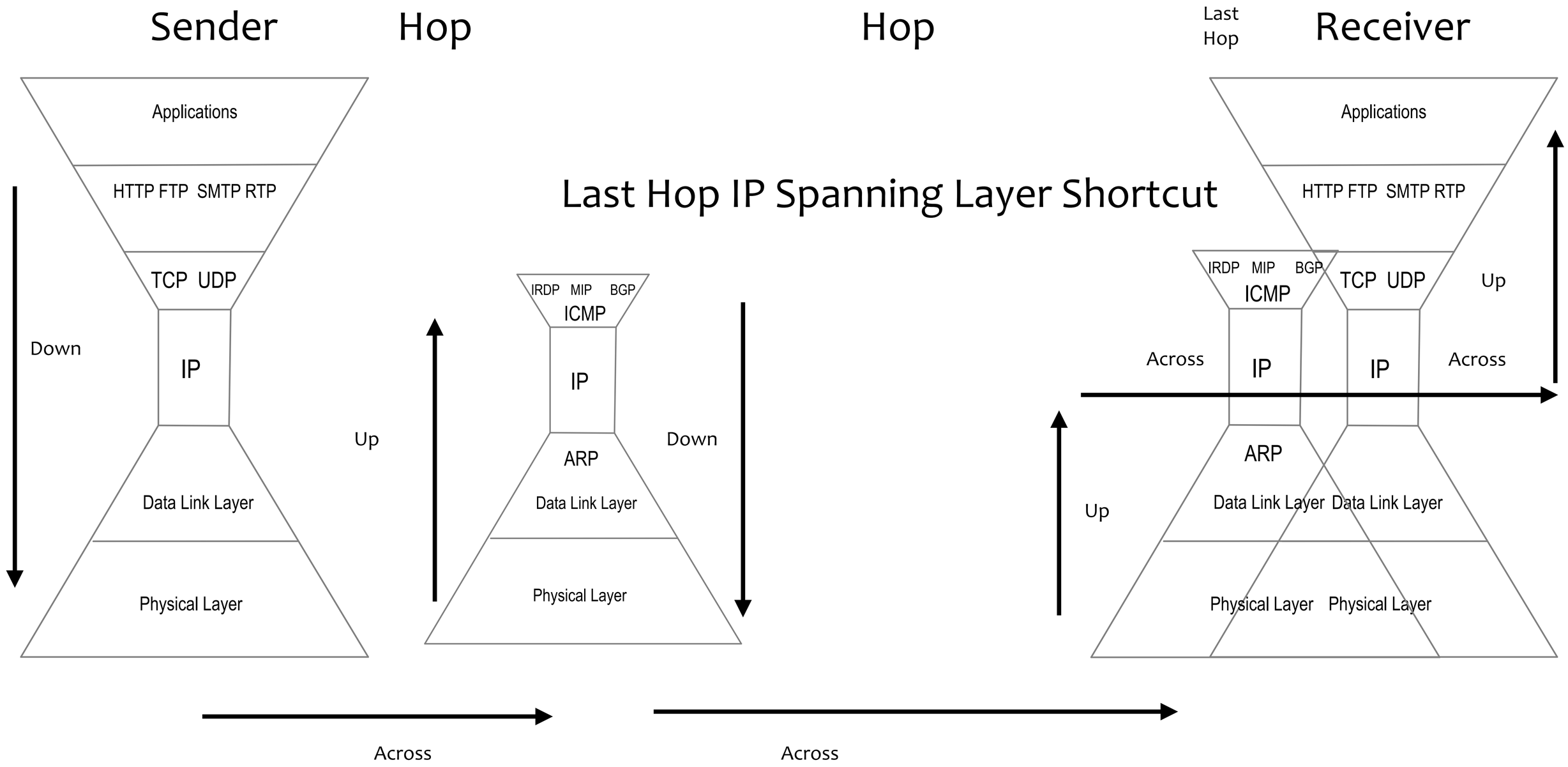
IP Router Layer Intermediation



Same Local Channel Routing Shortcut

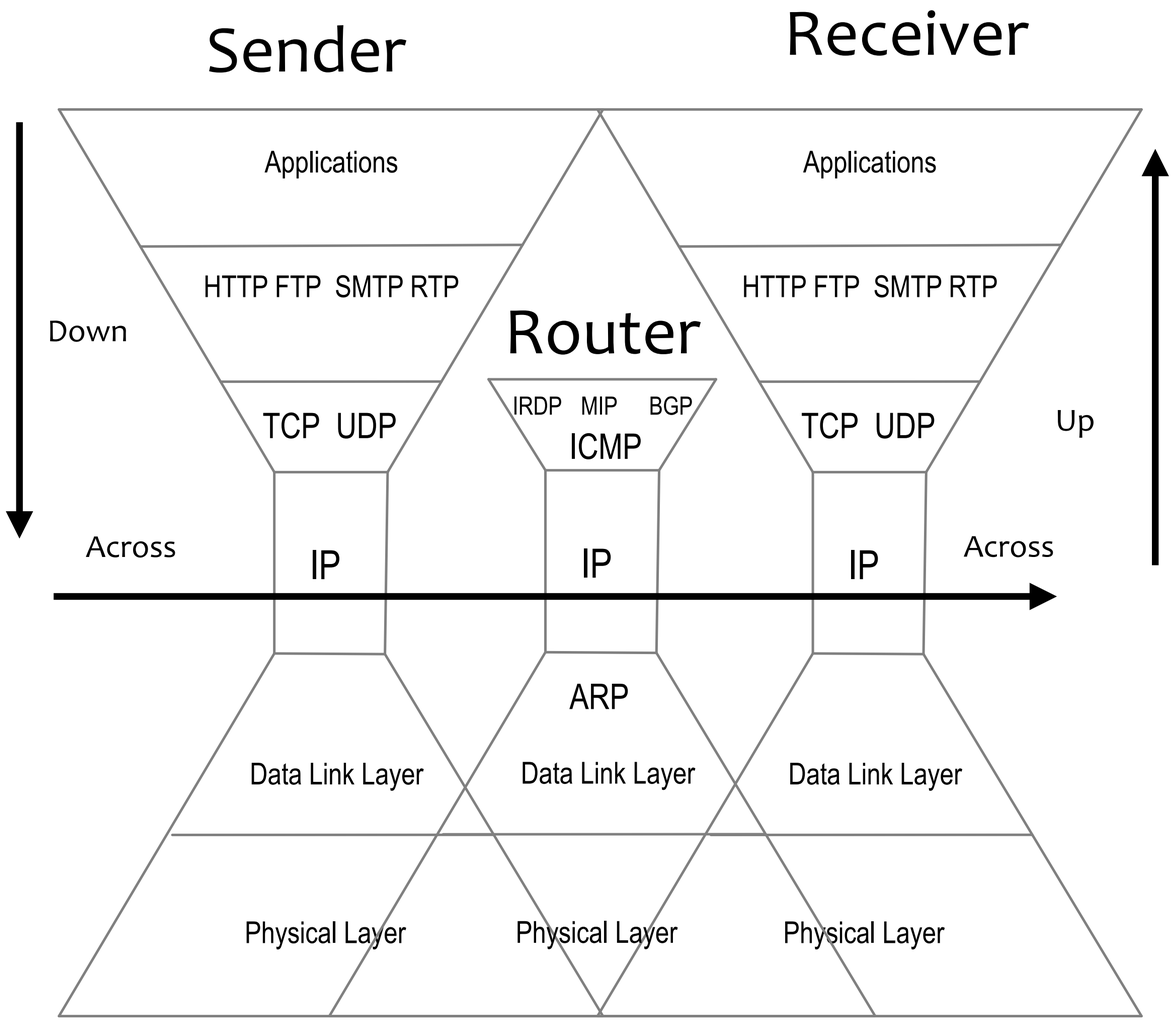


Last Hop Routing Shortcut via Spanning Layer



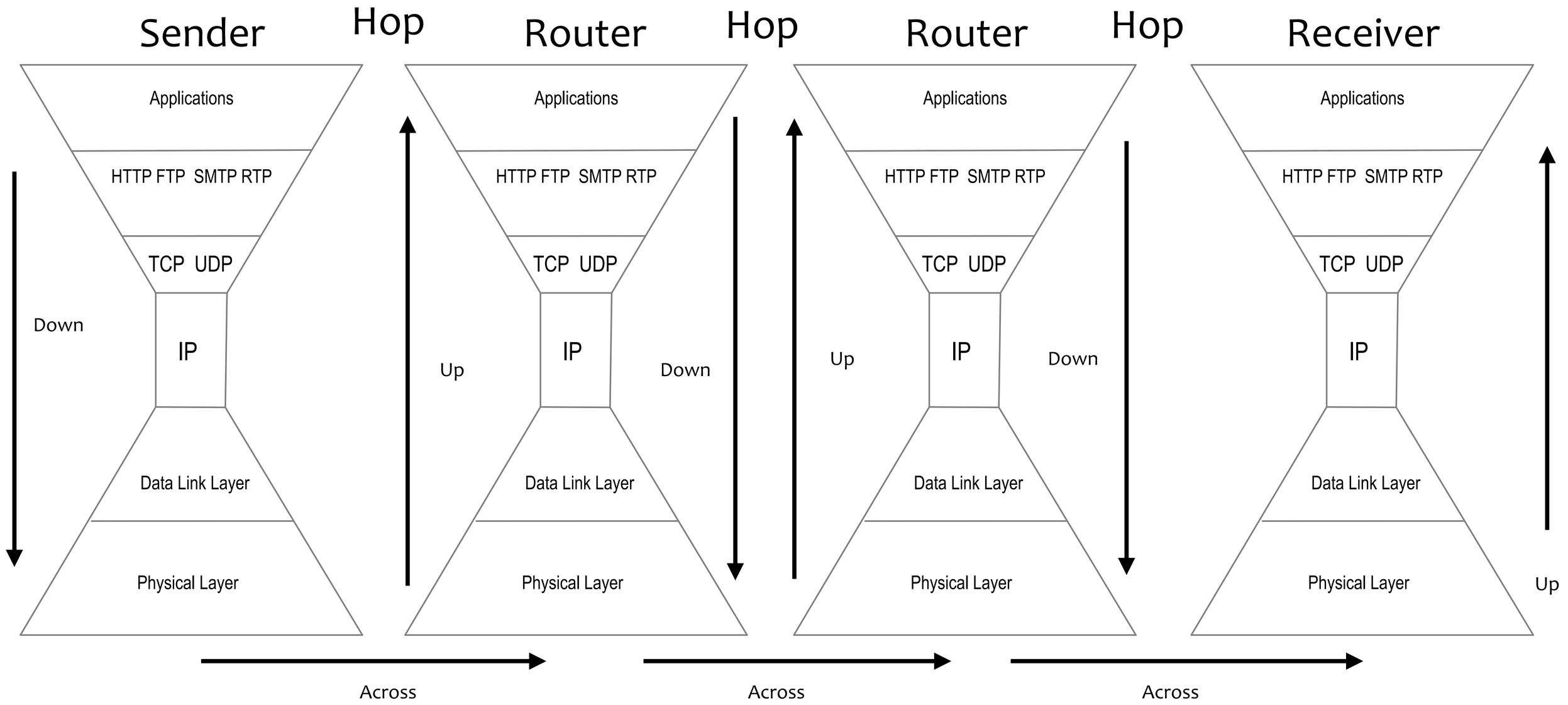
Local Hop Routing Shortcut via Spanning Layer

Local Hop IP Spanning Layer Shortcut

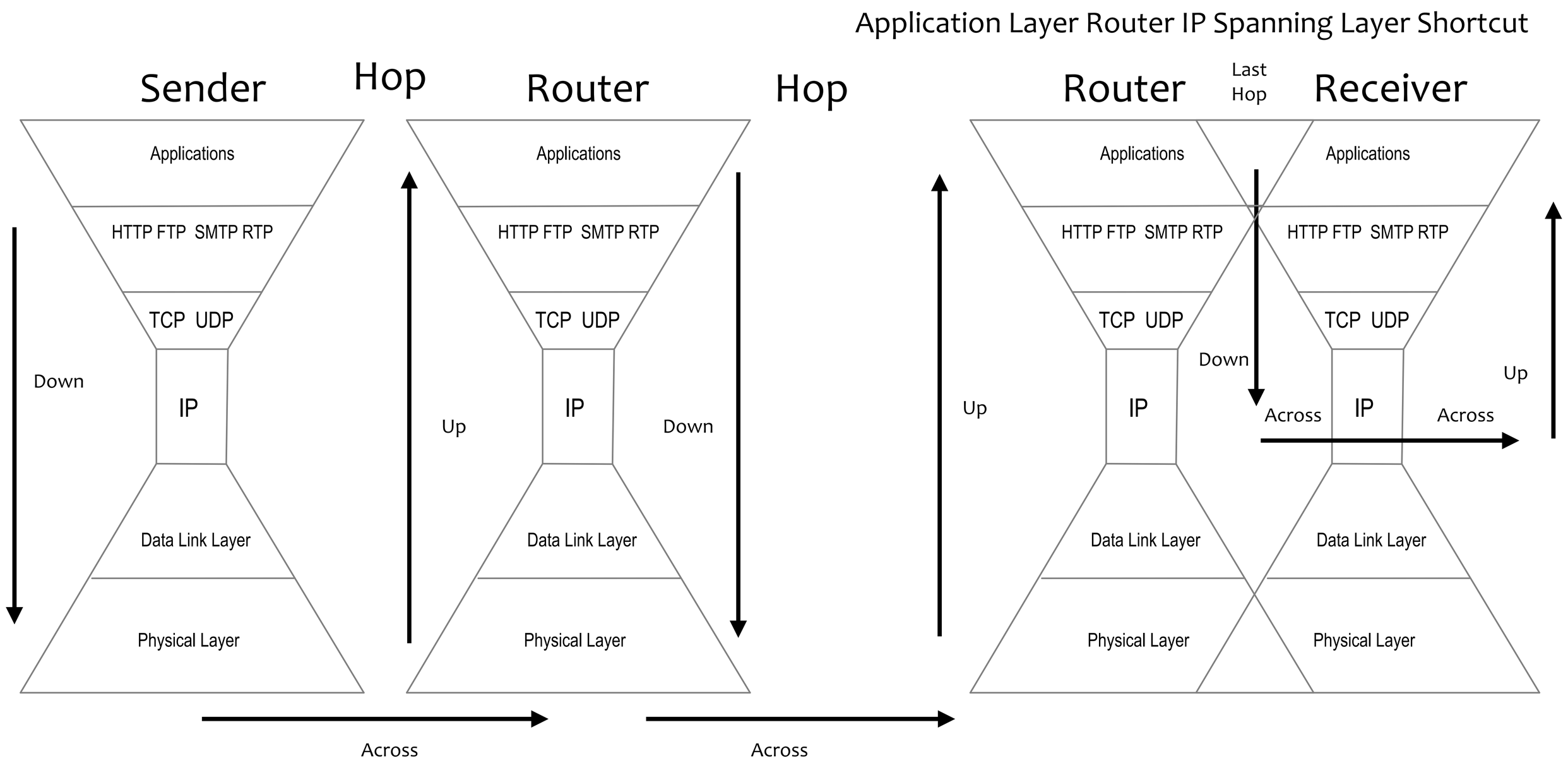


Application Layer Router Intermediation

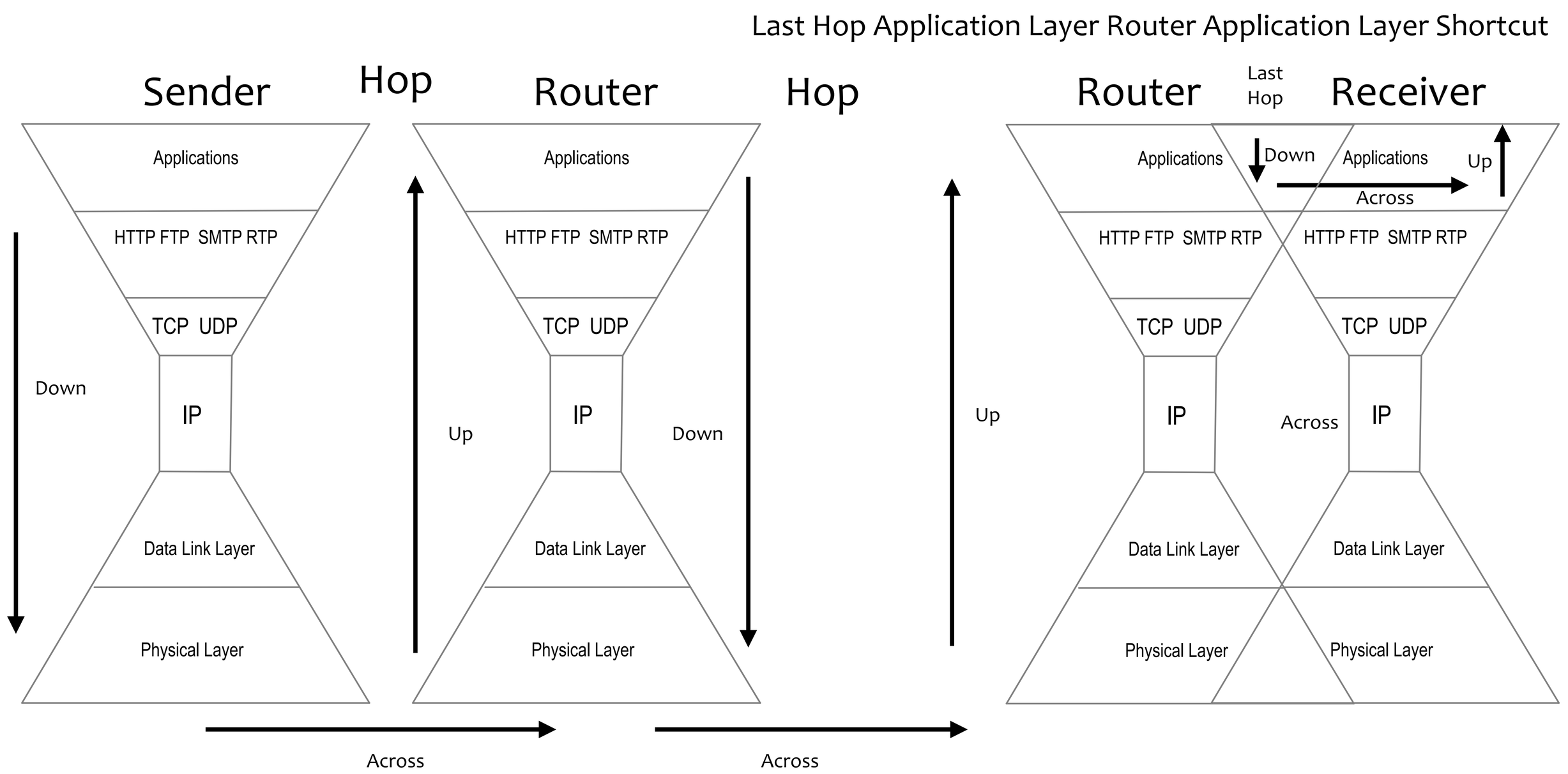
Application Layer Router
Routing & Discovery



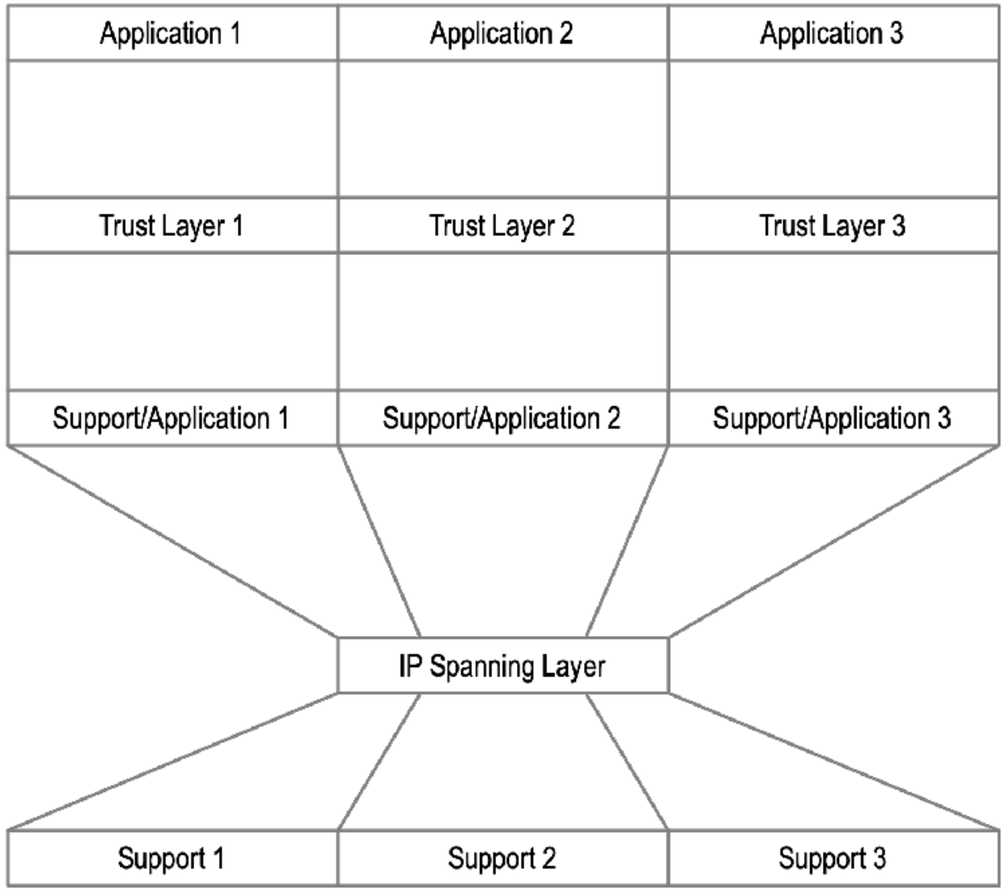
Last Hop Application Layer Routing Spanning Layer Shortcut



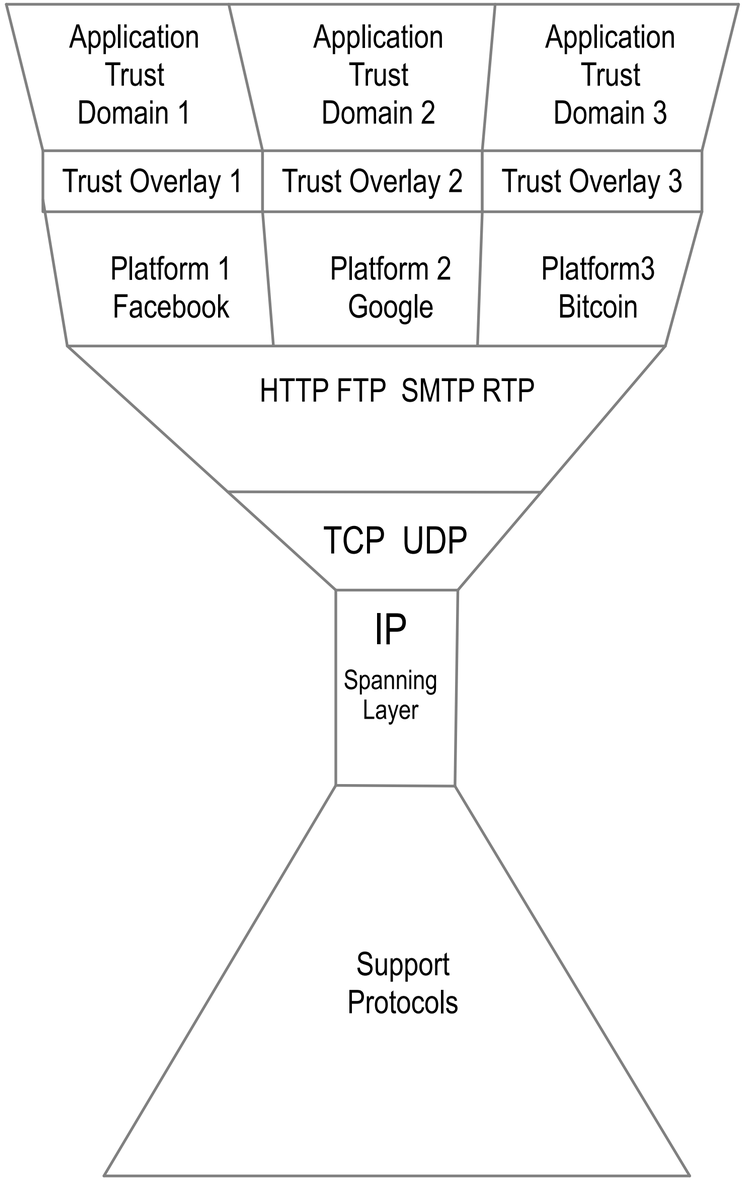
Last Hop Application Layer Routing Application Layer Shortcut



Platform Locked Trust



Trust Domain Based Segmentation

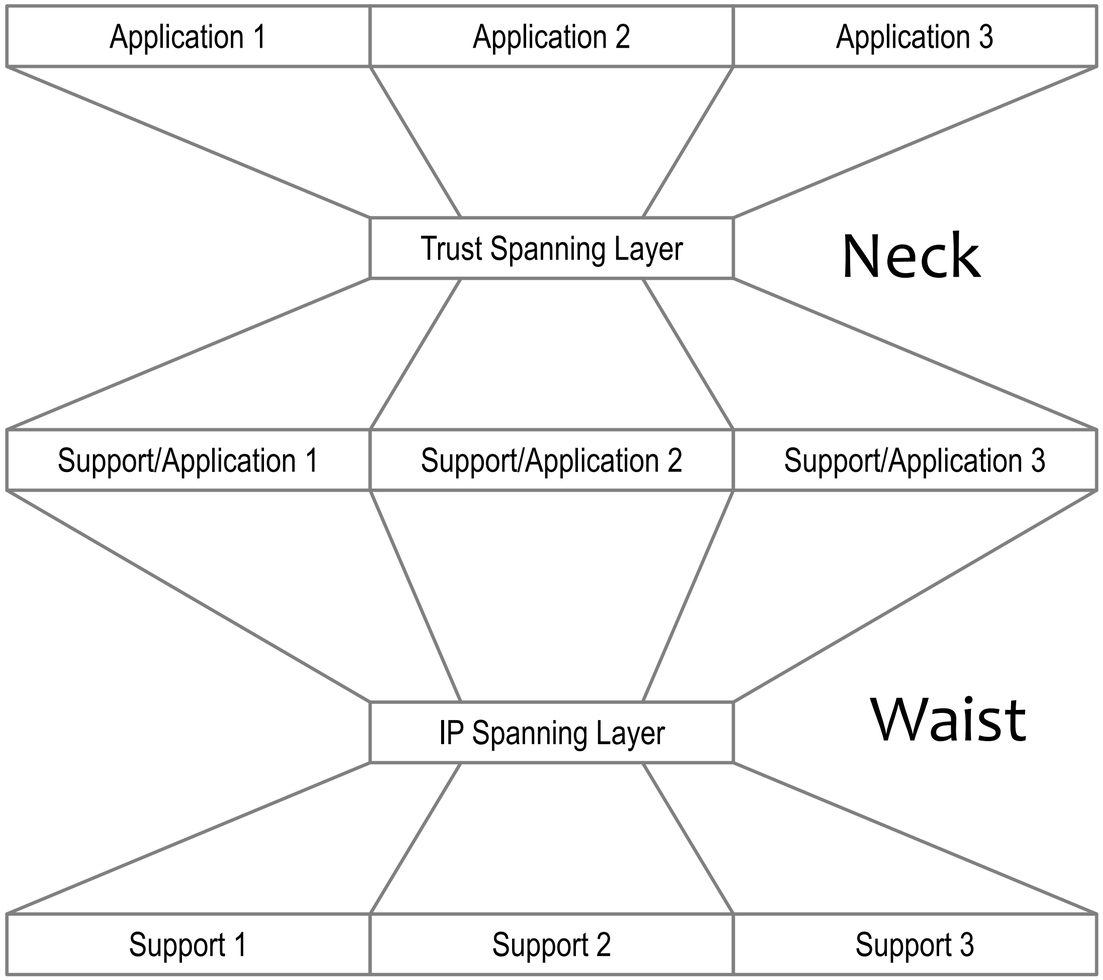


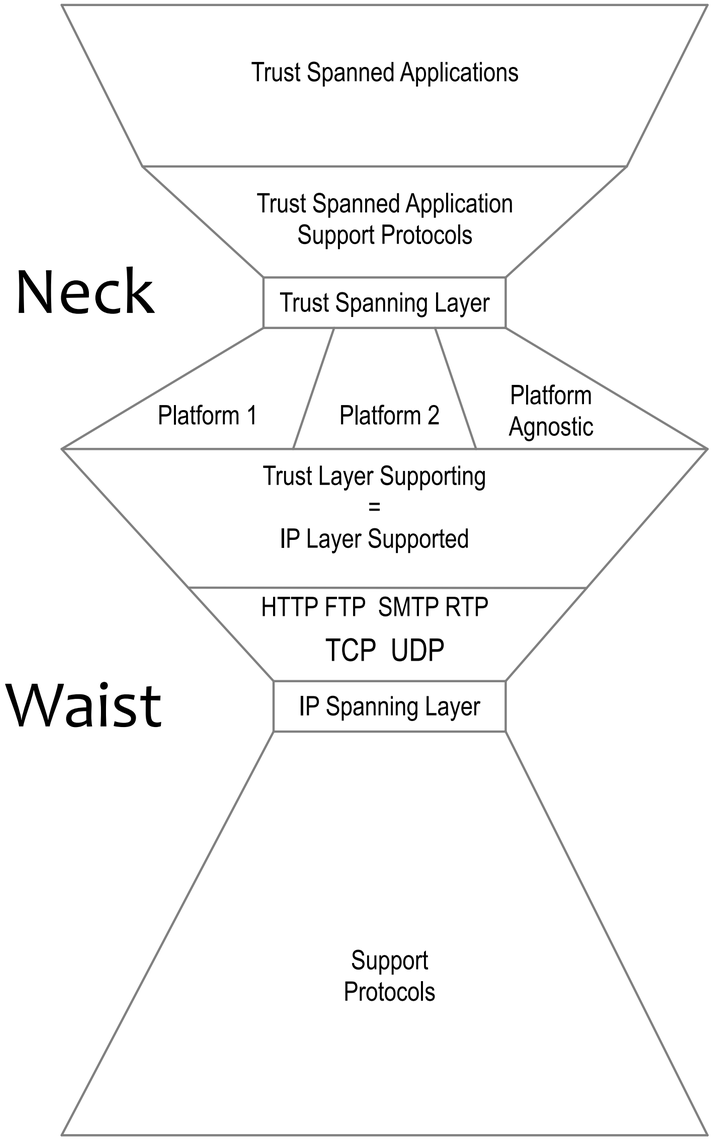
Each trust layer only spans platform specific applications

Bifurcated internet trust map

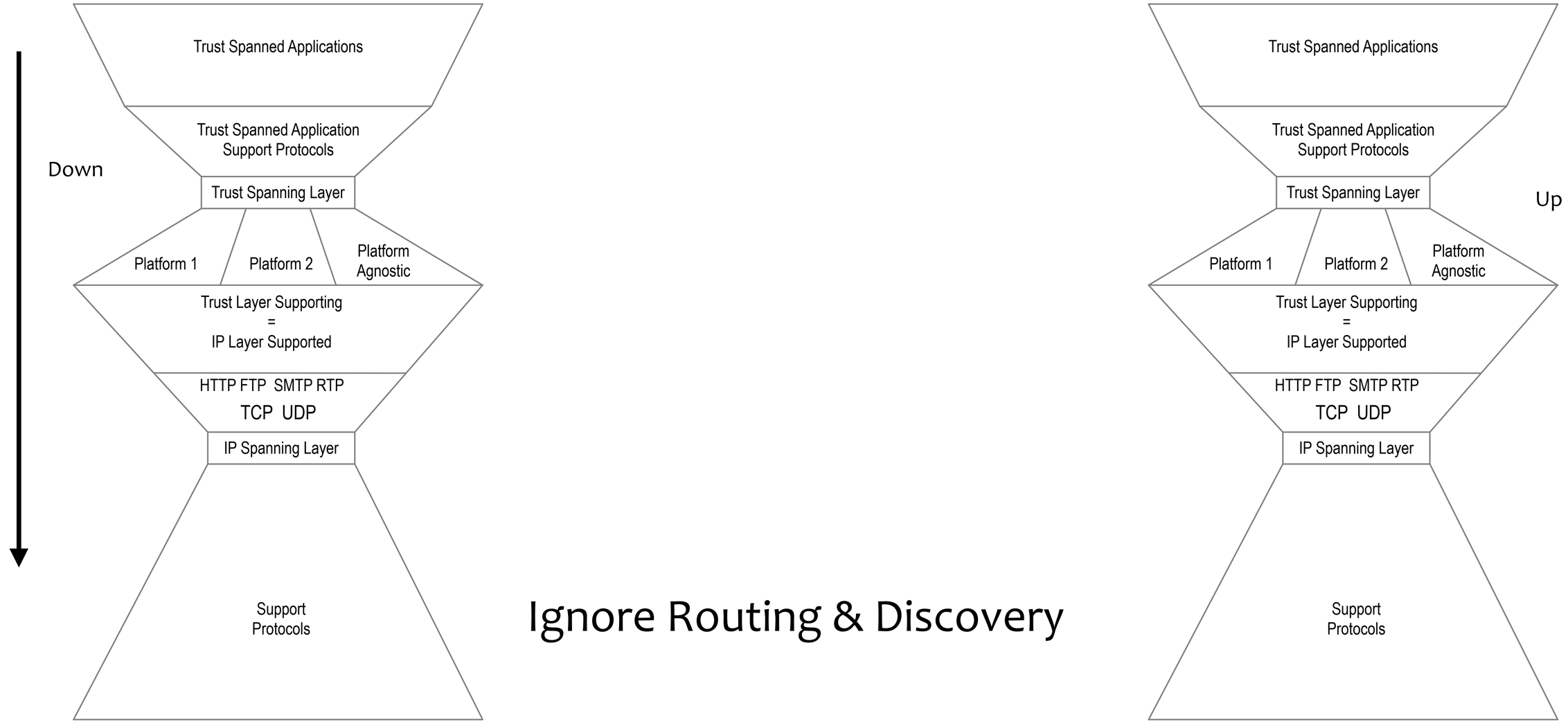
No *spanning* trust layer

Solution: Trust Neck and IP Waist

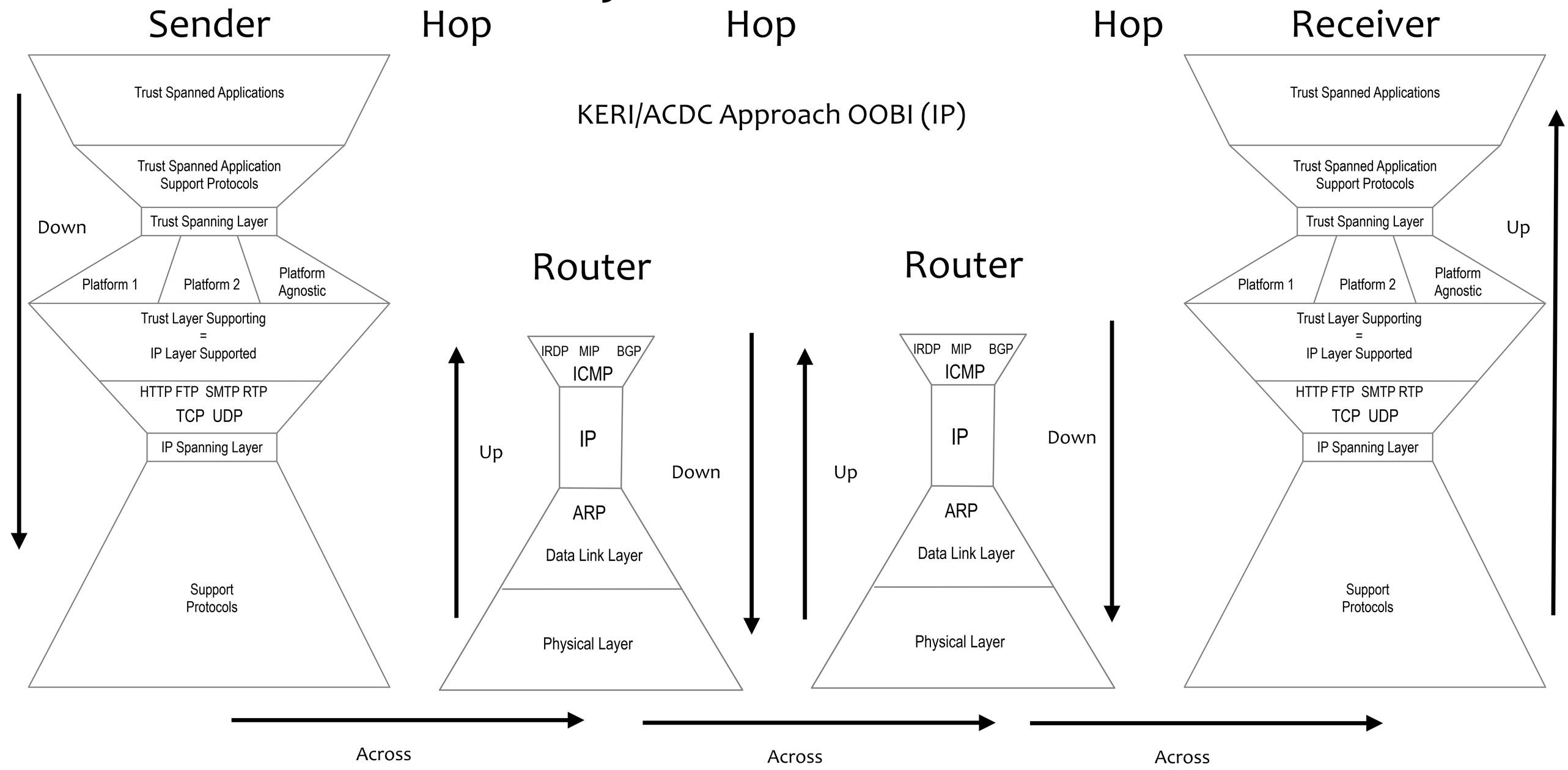




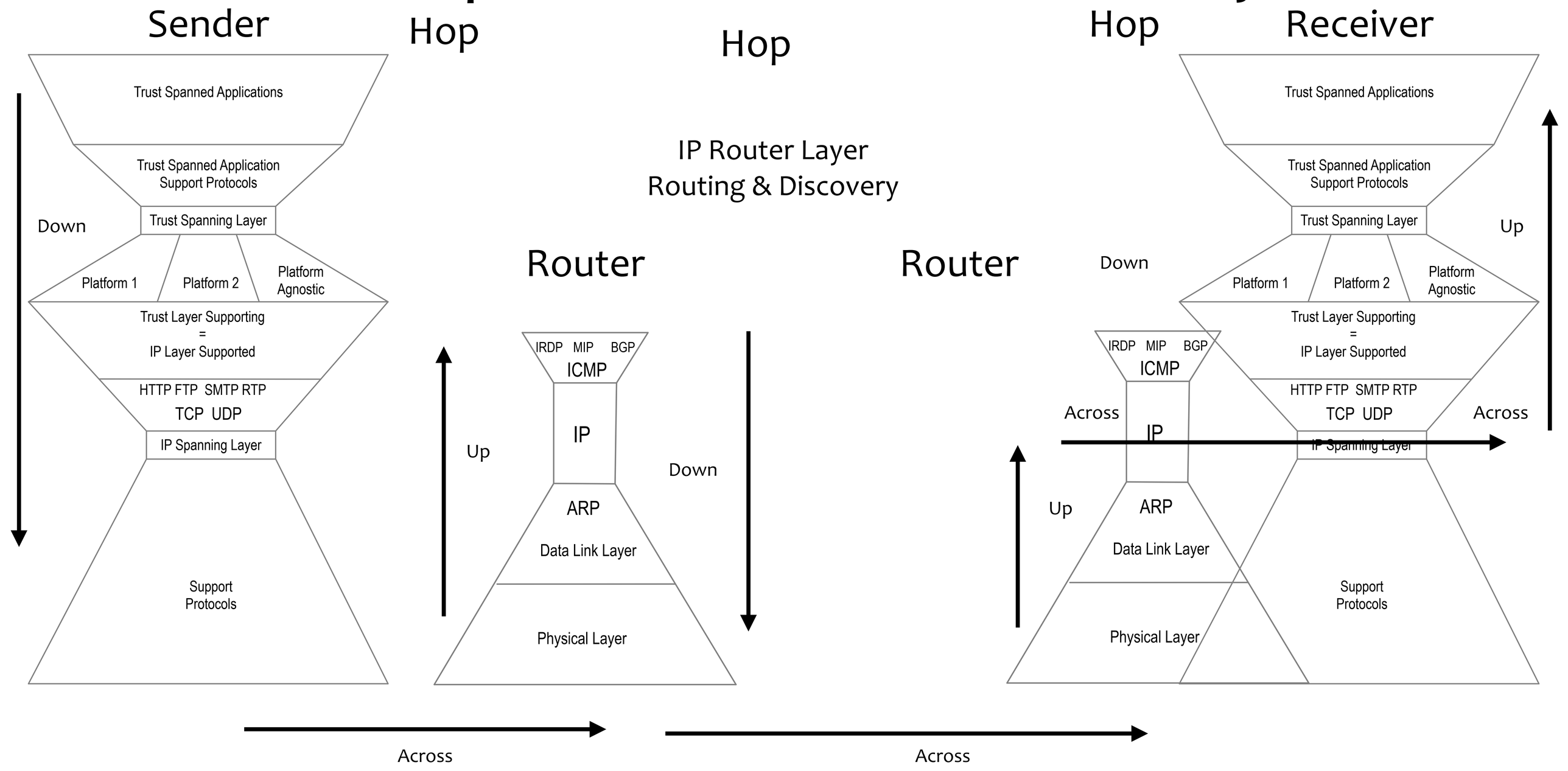
Trust Application Layer Endpoint Spanning



Untrusted Layer Router Intermediation



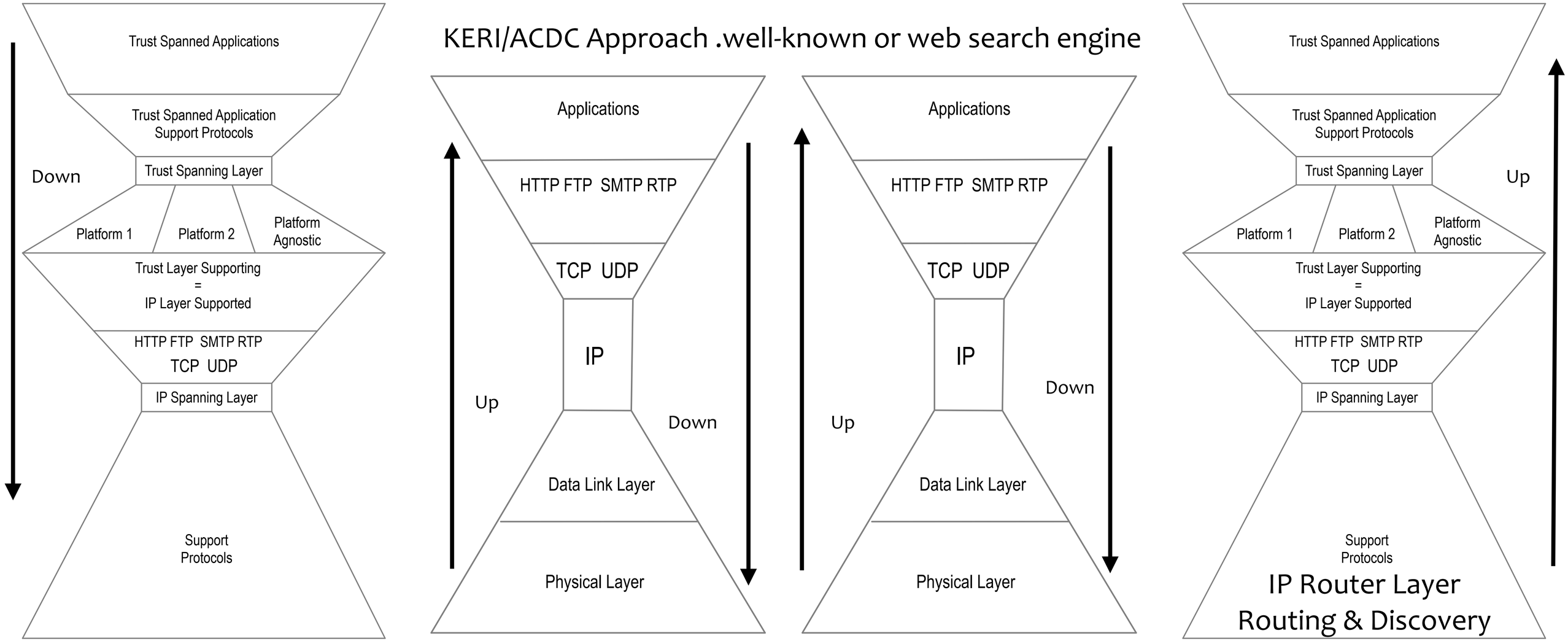
Last Hop Shortcut via Untrused Layer



Untrusted Layer Router Intermediation

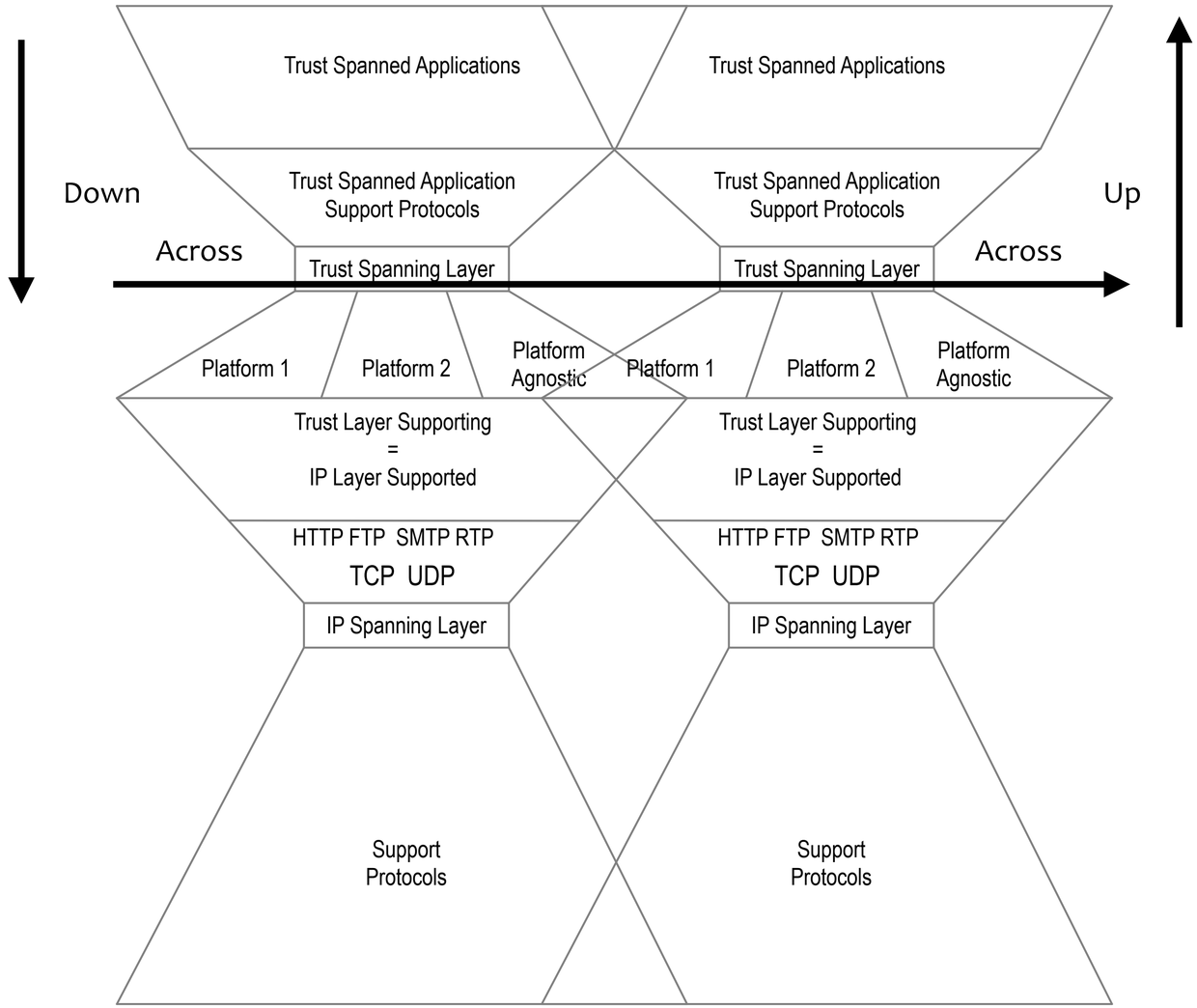
Sender Hop Router Hop Router Hop Receiver

KERI/ACDC Approach .well-known or web search engine

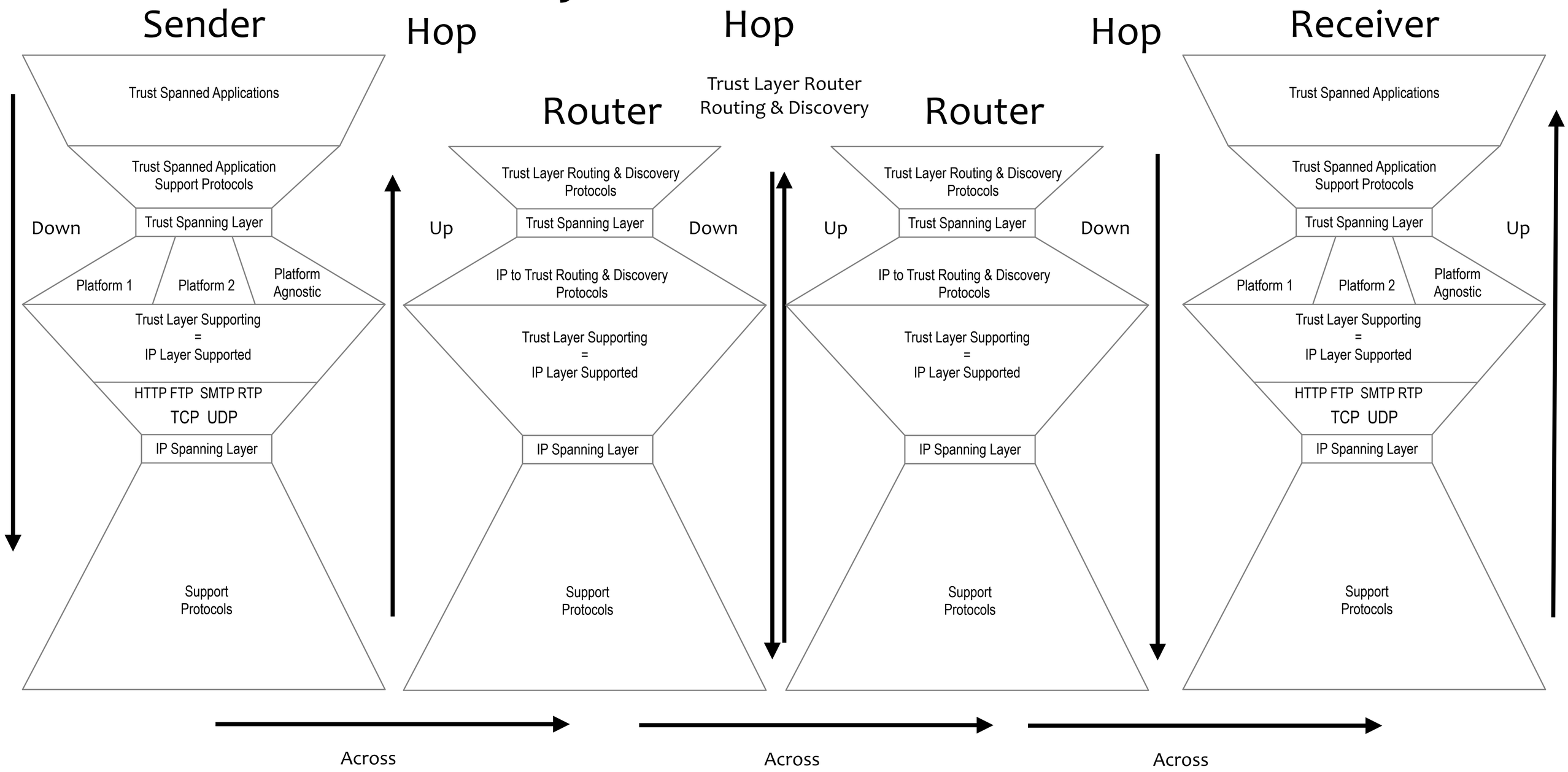


Across Across Across

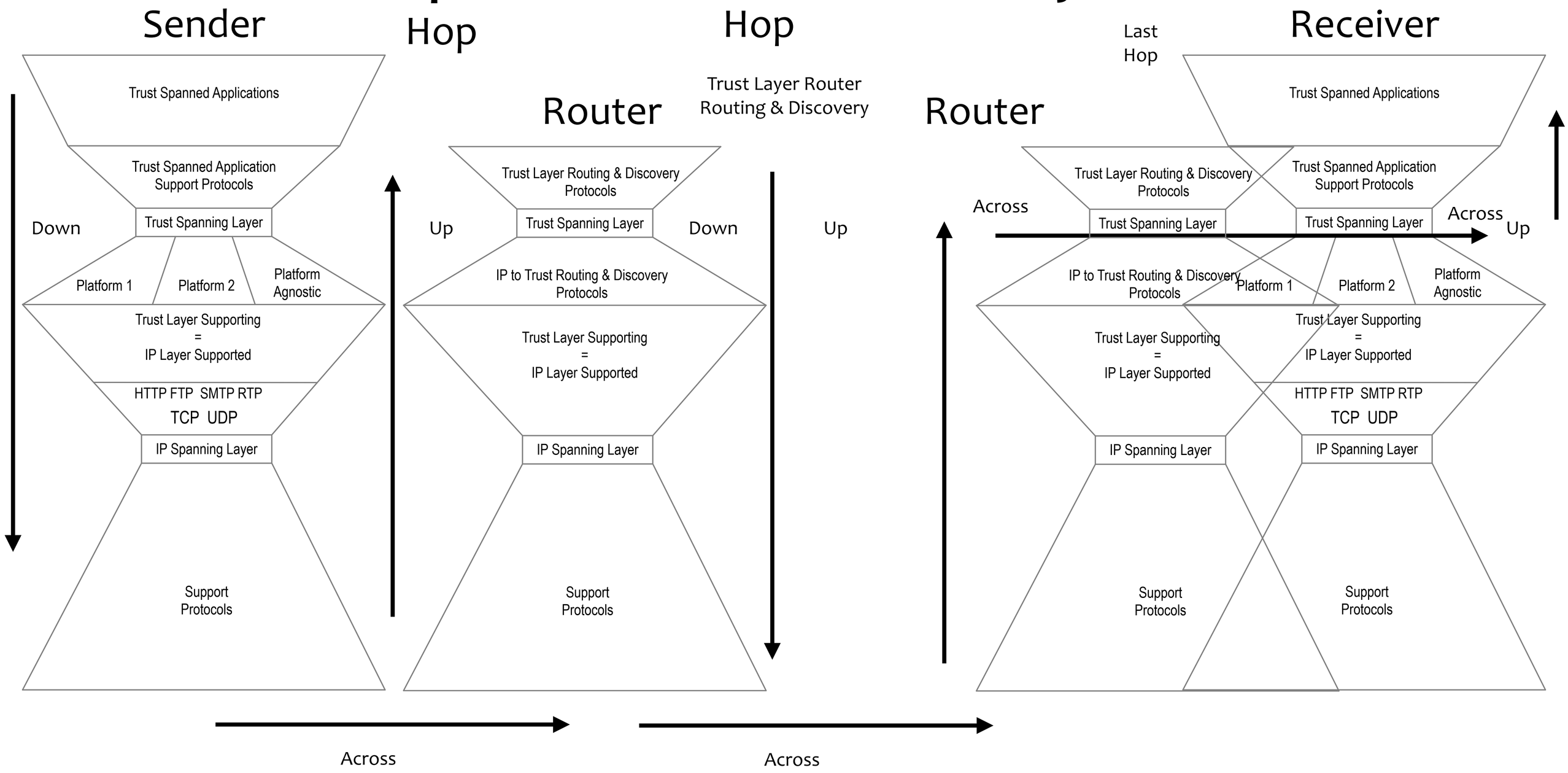
Shared Layer Routing Shortcut



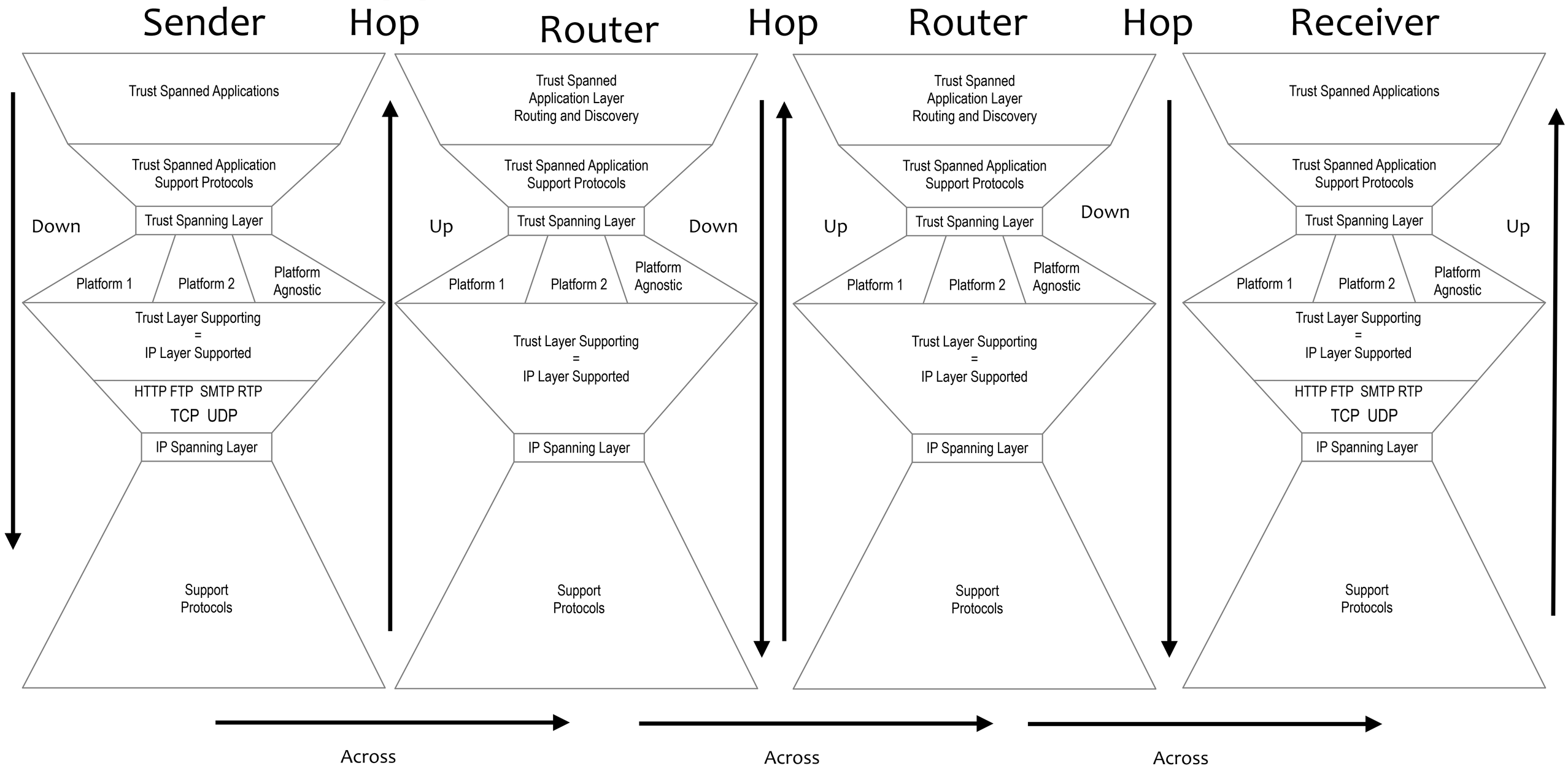
Trusted Layer Router Intermediation



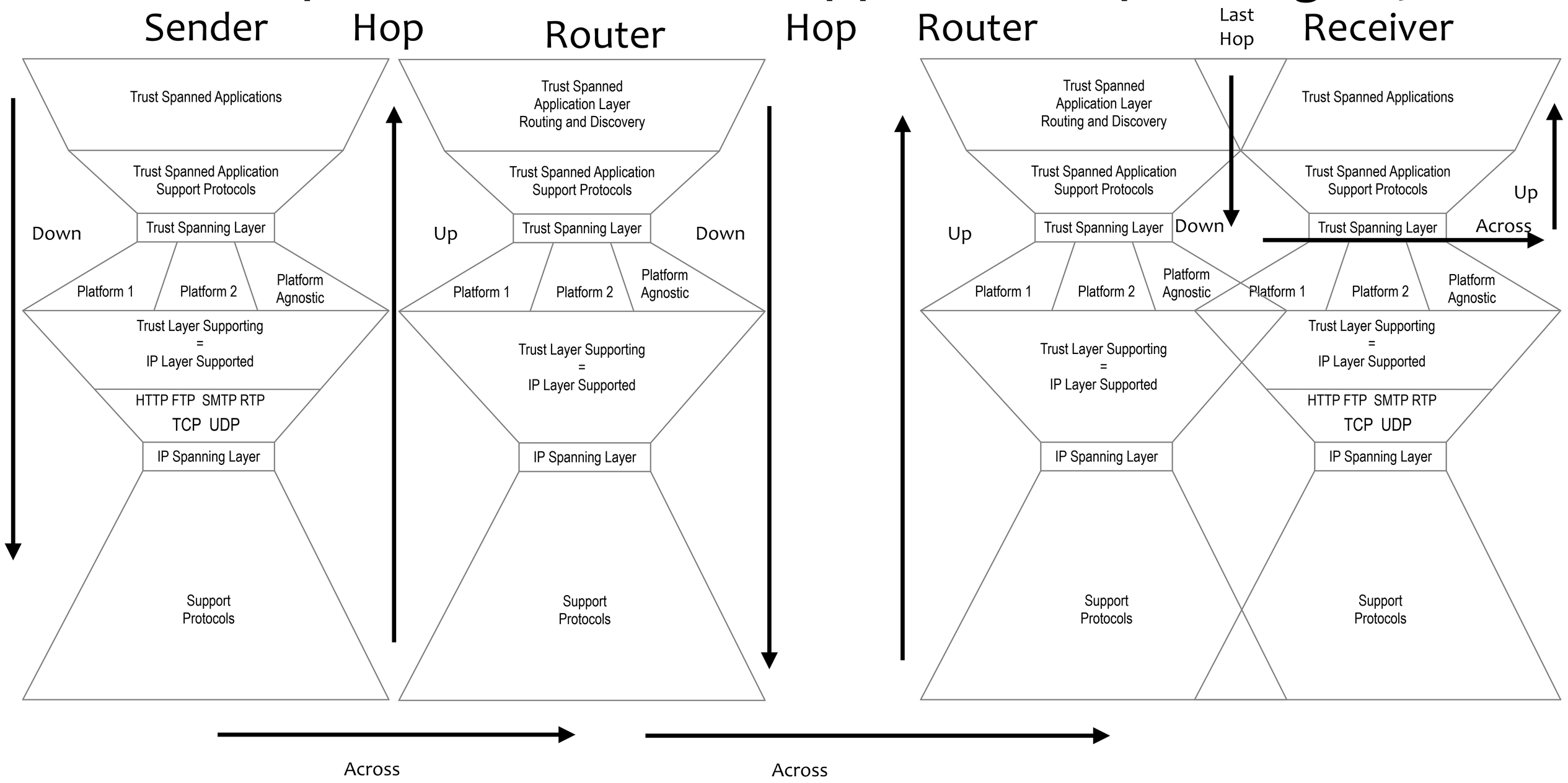
Last Hop Shortcut Trusted Layer Router



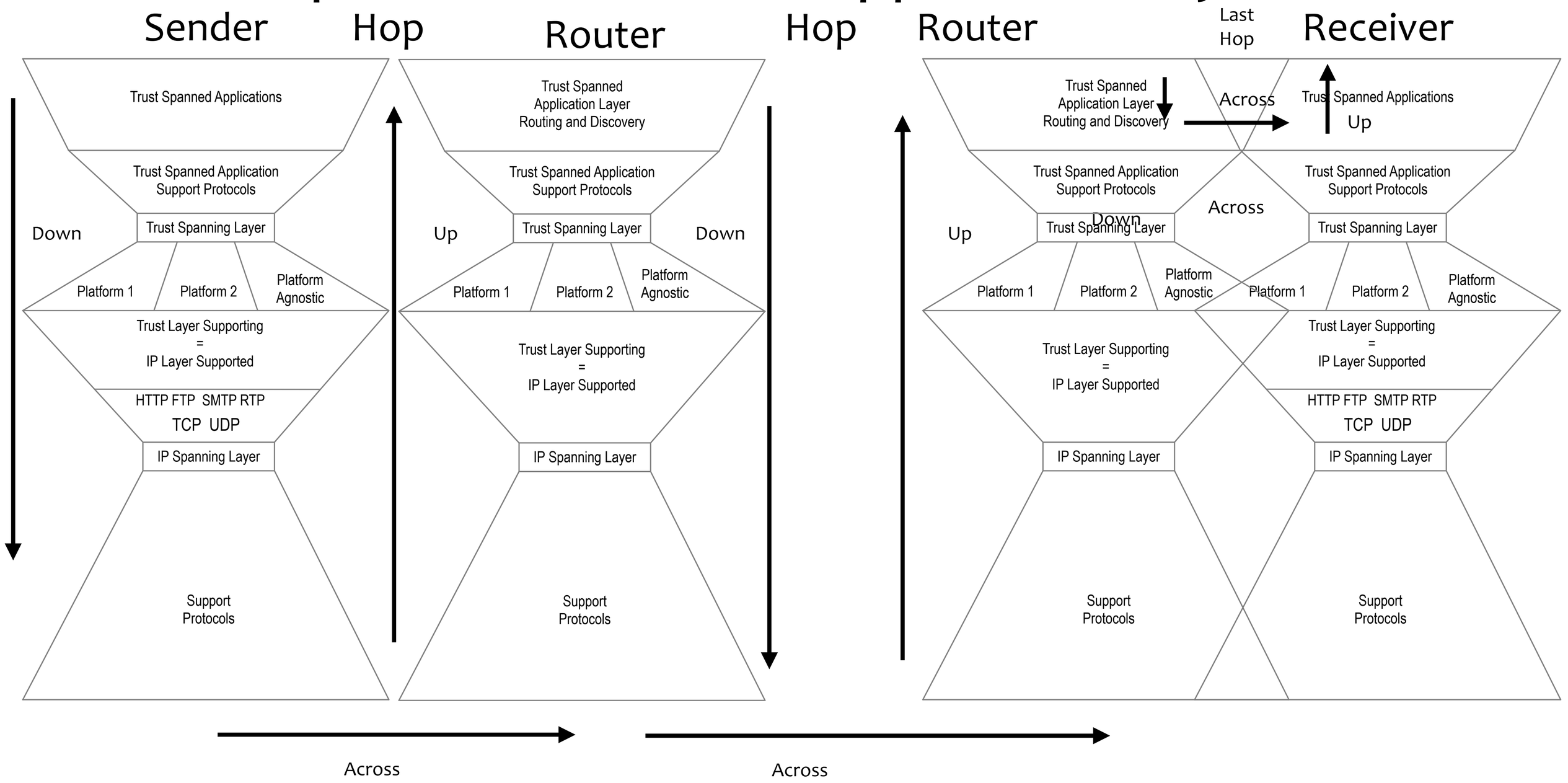
Trusted Application Layer Router Intermediation



Last Hop Shortcut Trusted Application Spanning Layer



Last Hop Shortcut Trusted Application Layer Router



Trust Spanning Layer Solution

Minimally sufficient means (satisfies hourglass theorem's weakness property) for trust spanning layer is an identity (identifier) system security overlay.

Also need routing and discovery protocols that are compatible with the trust spanning layer.

Routing and protocols are not part of the trust spanning layer but augment it. Routing and discovery protocols are essentially adjacent to the trust spanning layer protocol.

IP Analogy

A MAC Address is to an IP Address as an IP address is to a TSP identifier

A cellular SEI is to an IP address as an IP address is to a TSP identifier

Mapping to/from TSP Identifier and IP Address is like ARP/RARP which map to/from IP address and MAC address

ARP is routing and discovery below IP

Router that forwards source TSP packets to dest TSP is below TSL and is not trustable.

Router that verifies source TSP signatures on packets is below TSL and is not trustable but provides a quality of service.

Router that has TSP identifier and endorses source TSP packets for which it has verified signatures one is above TSL and is a trustable router.

TCP came before IP. TCP is a connected services protocol originally design to support remote terminals (telnet, rsh, rcp) that is inappropriate for other connectionless applications (like rpc/udp). Routing is difficult with TCP. So IP was invented to support all types of protocols that all share one interface, the IP layer to enable routing. And with IP came UDP/IP and TCP/IP.

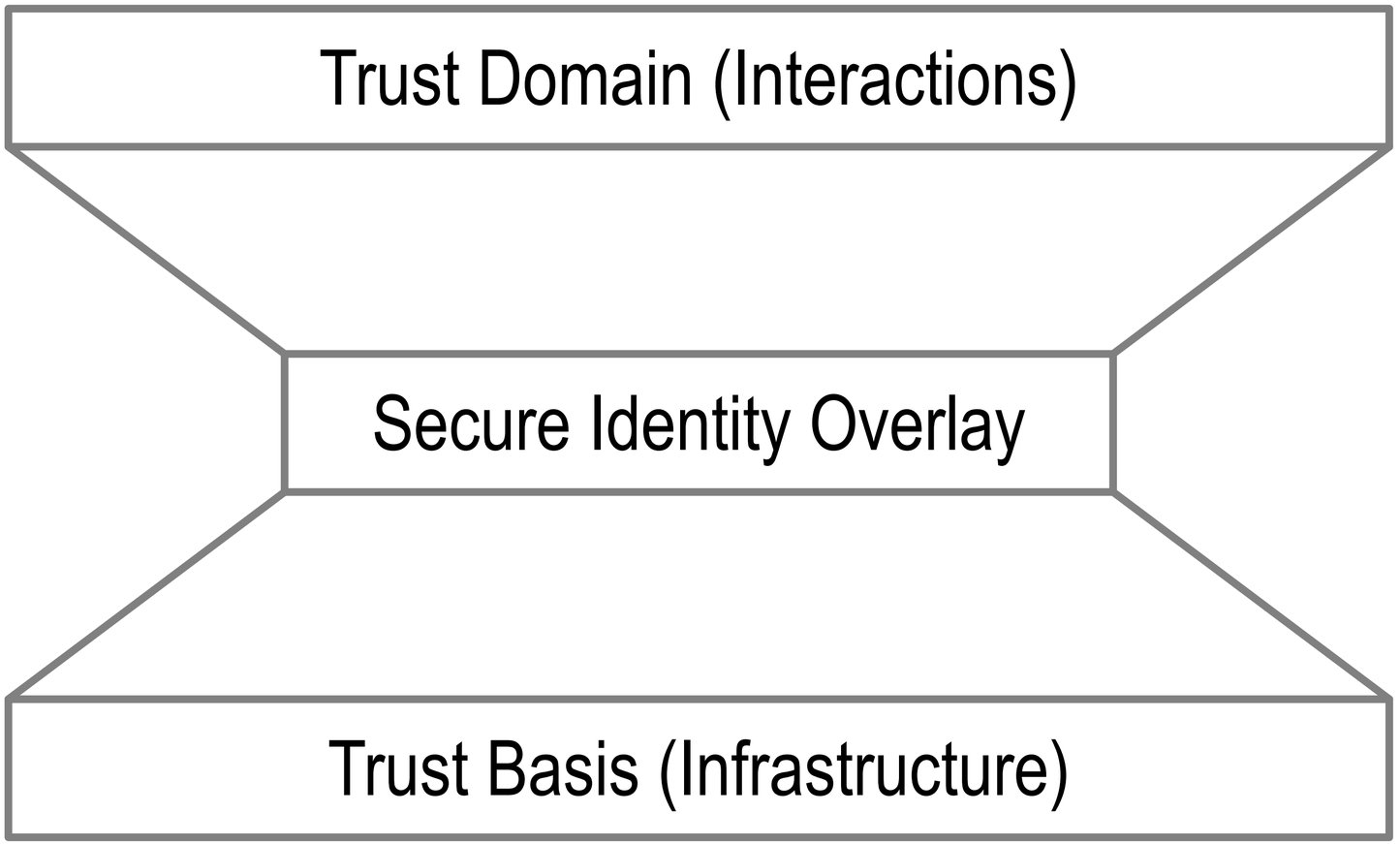
But IP is not secure which forces the need for a trust spanning layer that is secure.

Trustable Connections (signed traffic) between two endpoints are analogous to TCP above IP i.e. above TSP. Untrustable connections (not signed traffic) are below TSP.

TCP/IP != Connectioned TSP

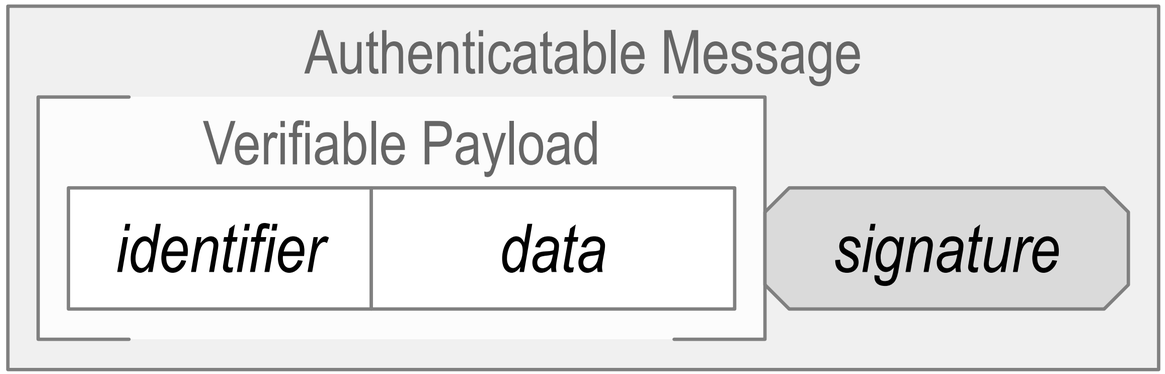
Signed statements that are not part of a dedicated connection are like UDP datagrams, they are trustable but are topology free for routing purposes.

Identity (-ifier) System Security Overlay



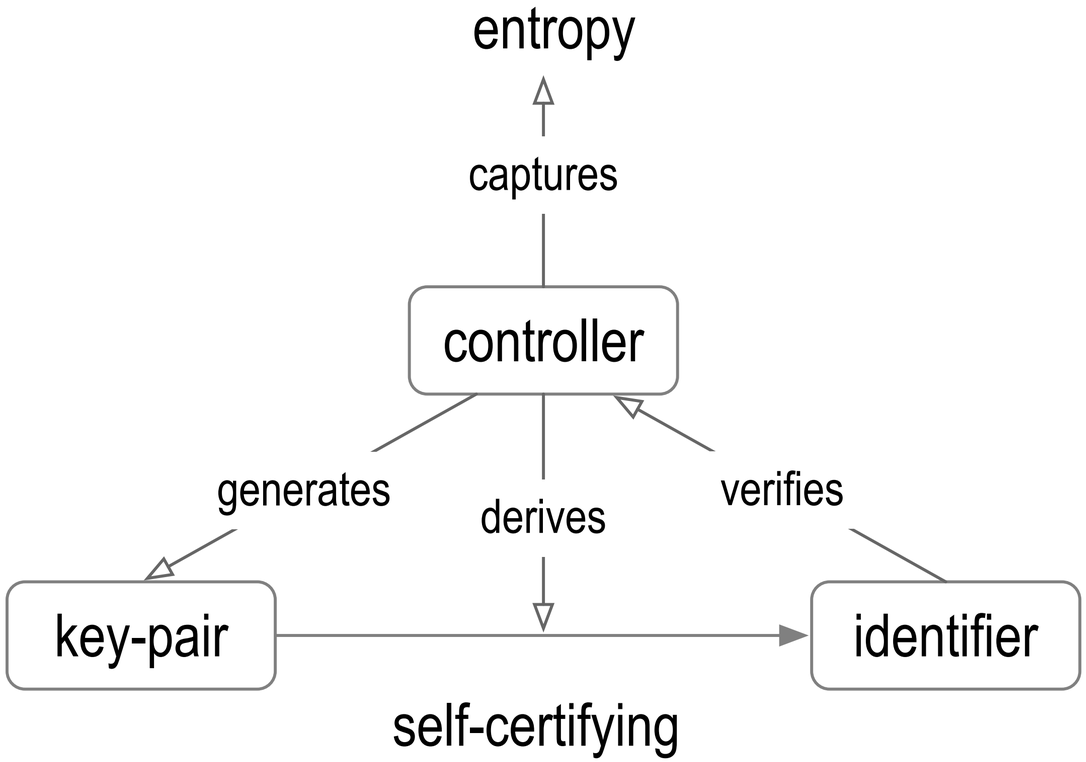
Identity (-ifier) System Security Overlay

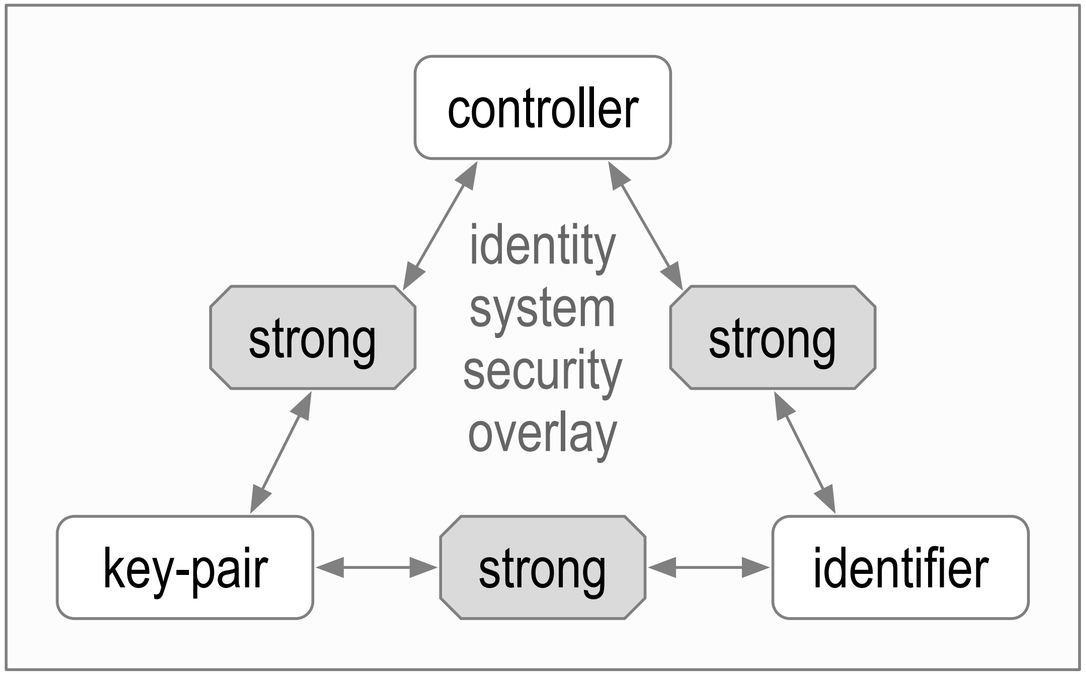
Establish authenticity of IP packet's message payload.



The overlay's security is contingent on the mapping's security.

Self-Certifying Identifier (SCID) (Ephemeral): Issuance and Binding





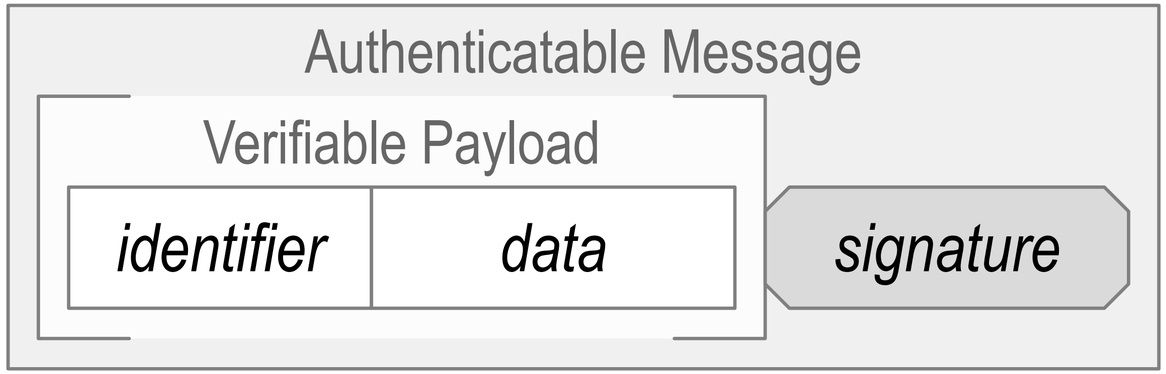
Self-Certifying Identifier Issuance Triad

cryptographic root-of-trust

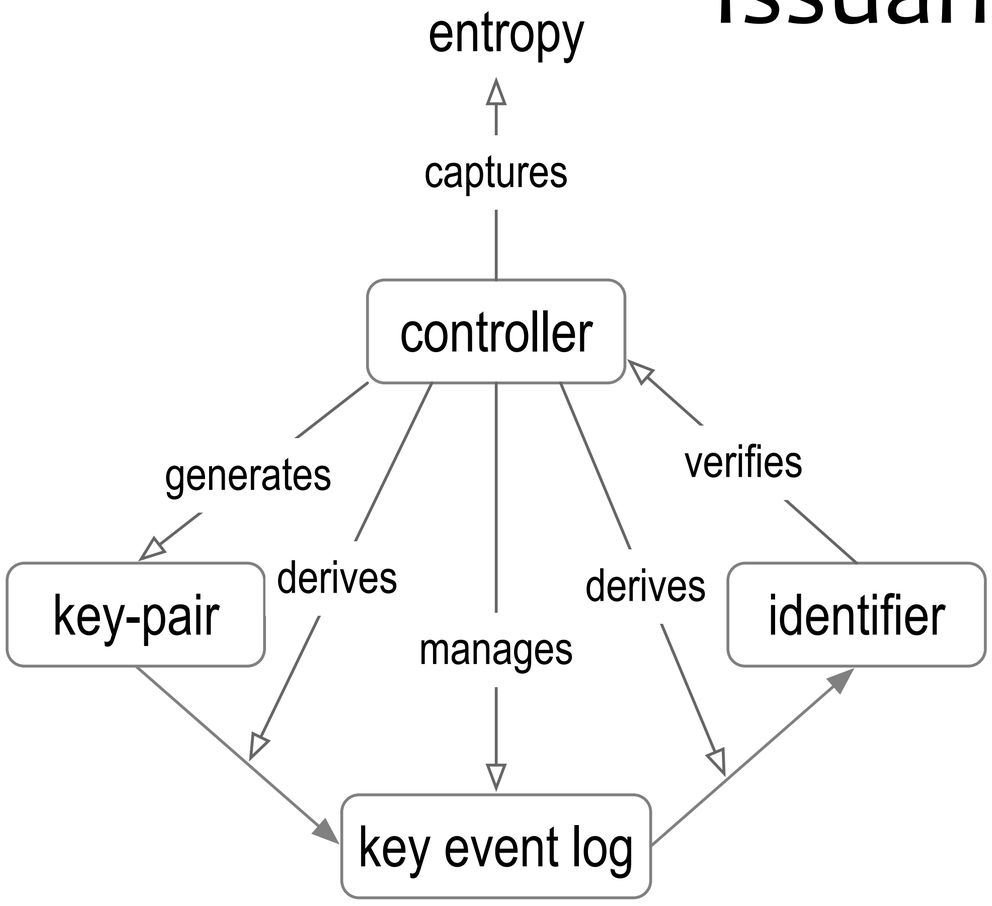
Identity (-ifier) System Security Overlay

ephemeral (non-transferable) mapping = identifier is encoded public key

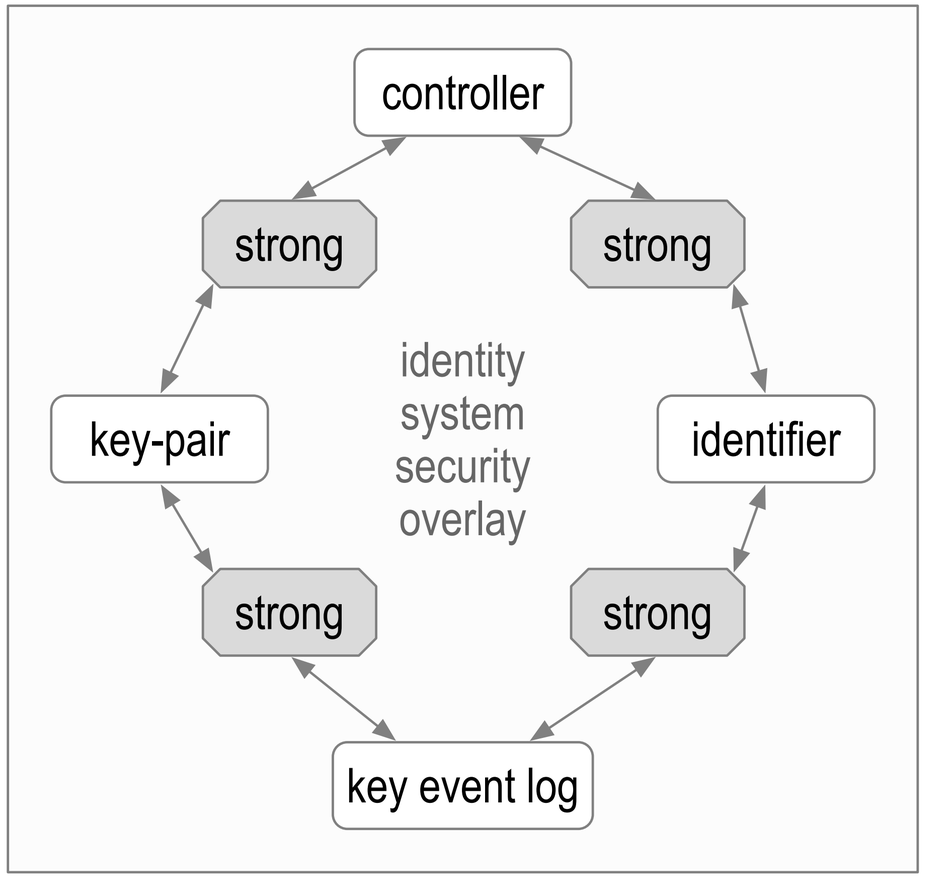
Establish authenticity of IP packet's message payload.



Autonomic Identifier (AID) (Persistent): Issuance and Binding



autonomic, self-managing



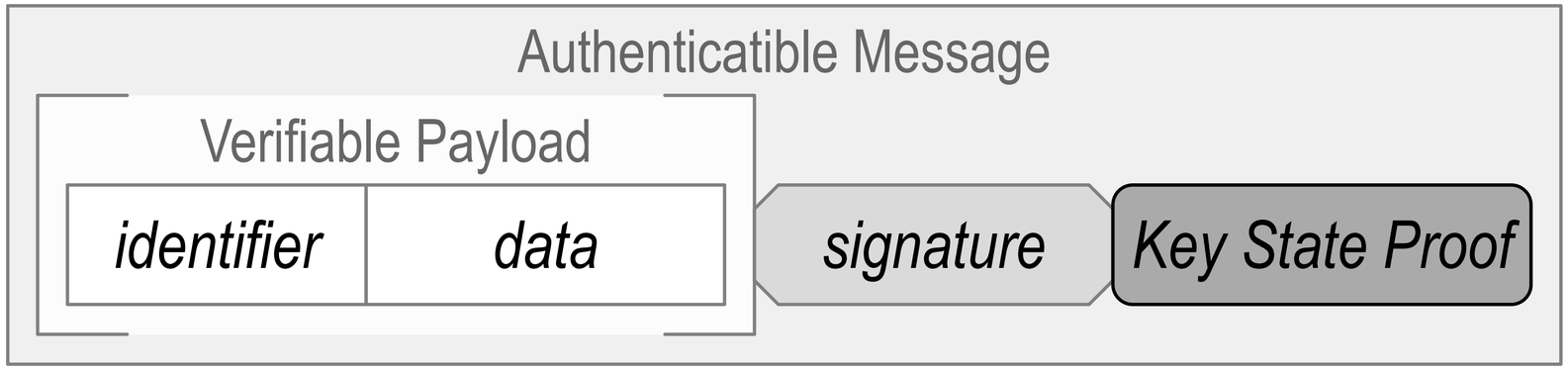
Autonomic Identifier Issuance Tetrad

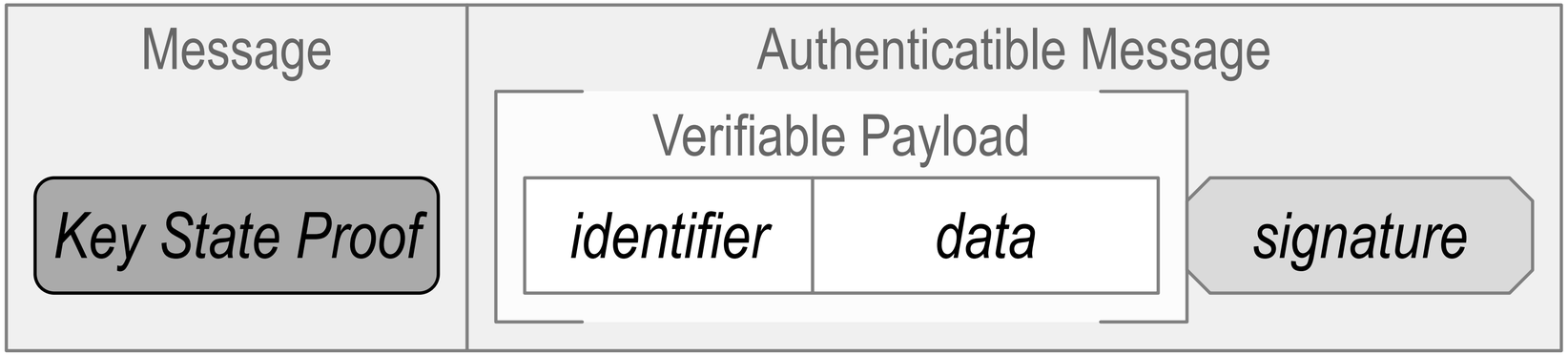
cryptographic root-of-trust & verifiable persistent control

Identity (-ifier) System Security Overlay

persistent (transferable) mapping = verifiable data structure of key state changes

Establish authenticity of IP packet's message payload.





Identifier System Security

Authentic transmission of data may be verified using an identity system security overlay.

Message authenticity is provided by verifying signatures to the authoritative keys pairs for the identifier included in the message.

This overlay maps identifiers to cryptographic key-pairs.

The security of the overlay is contingent on the security of the mapping.

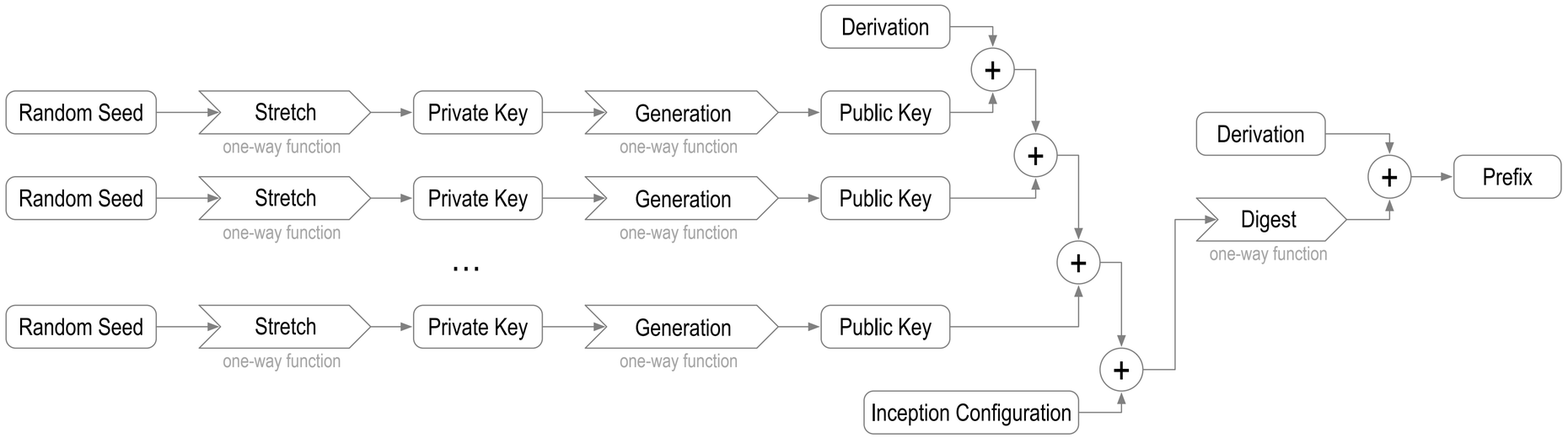
When those identifiers are self-certifying they are derived via cryptographic one-way functions from the key pairs.

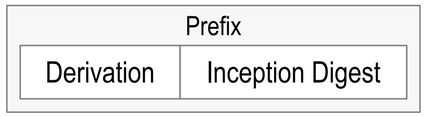
This strongly binds identifiers to key-pairs and makes control over the identifier equivalent to control over the key pairs.

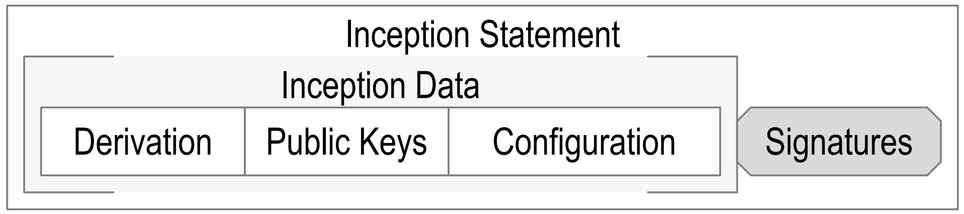
This provides a self-certifying identifier with a cryptographic root-of-trust.

A key event log (KEL) provide support for secure key rotation (transferable control) without changing the identifier.

Cryptographic Root-of-Trust & Persistent Control: Self-Certifying Identifier + Key Event Log

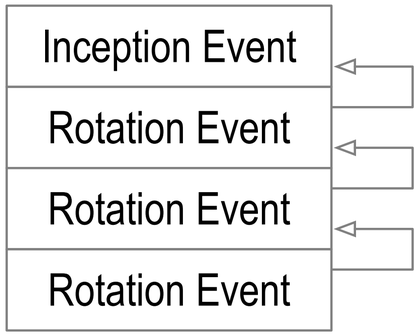






Key Event Log

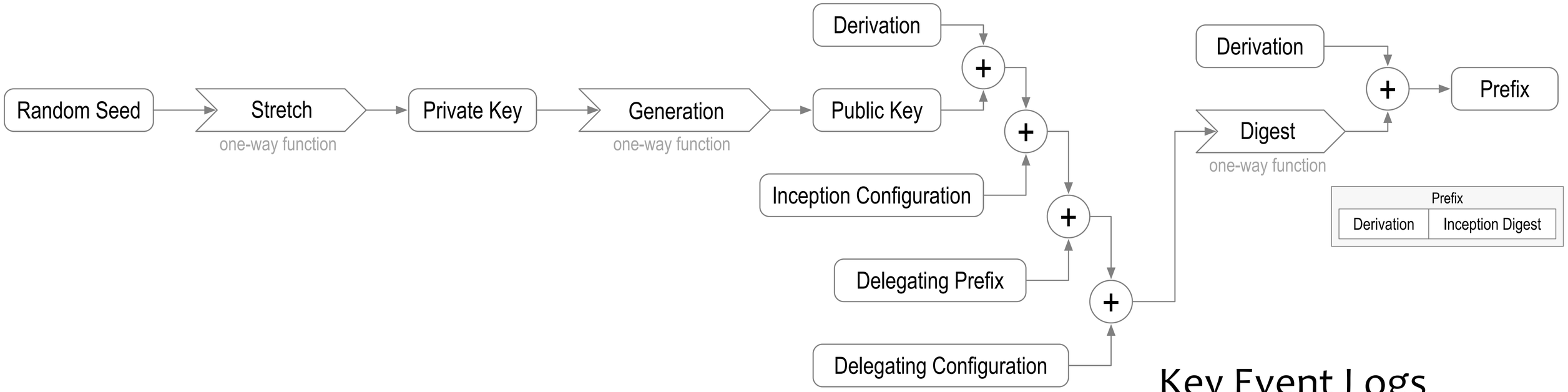
Establishment Subsequence

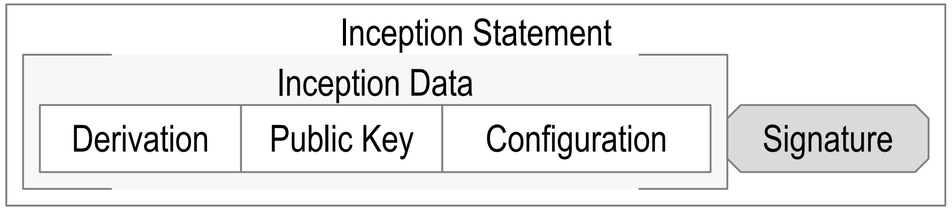


EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

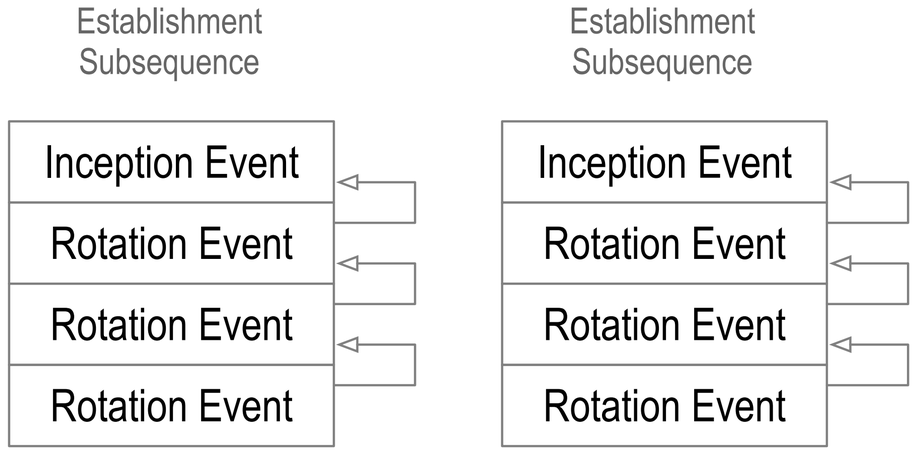
did:un:EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148/path/to/resource?name=secure#really

Delegated Identifiers





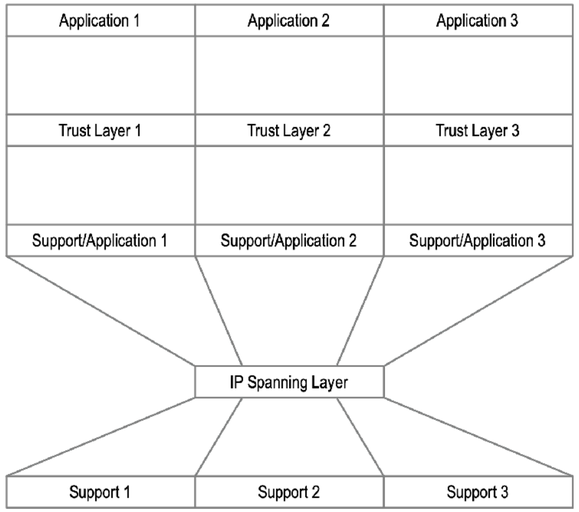
Key Event Logs



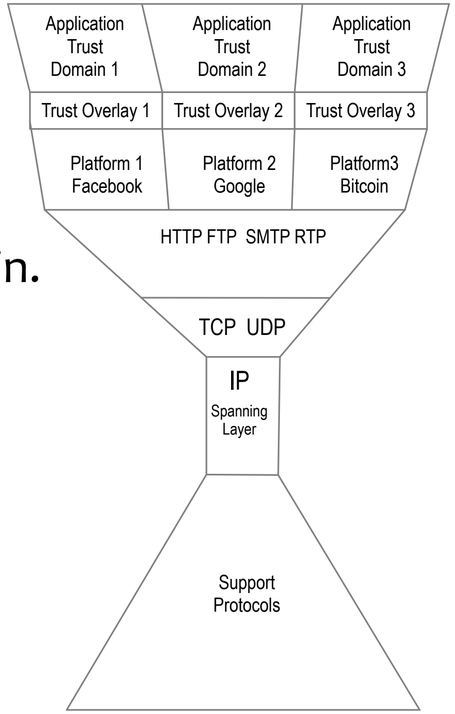
EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148

did:keri:EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148/path/to/resource?name=sec#yes

Interoperability for Segmented Trust



All interoperability mechanisms are not created equal.
Interoperability based on a spanning layer is unique.
Interoperability based on a trust-spanning layer is also unique.
A trust-spanning layer uses a common protocol to make one trust domain.
This is different from interoperability across multiple trust domains.



Examples of interoperability across multiple trust domains:

DNS/CA (trust spanning layer protocol but effective trust domain segmentation across multiple CA administrators)

URL based on DNS/CA (common name-spaced resources found across segmented trust domains)

“Login with” to a website supporting multiple OpenID providers (trust domain segmentation across multiple Identity Providers)

SSO with OpenID aggregator (Hide trust domain segmentation behind centralized Identity Provider)

Each host is its own OpenID Identity Provider

Client-side rendering of multiple language-specific backends (hide code segmentation with browser rendering)

Client-side routing of multiple language-specific backend renderers (hide code segmentation with browser routing of code

Universal DID method Resolver (trust domain method code segmentation by resolver rendering of did:docs)

DIDs as Namespace (name spacing segmented trust domains)

Multi-DID Method Wallet as Aggregator (Hide trust domain segmentation behind wallet UX, client side routing & rendering)

Flaws of DNS/CA as Trust Spanning Layer

Insecure Key Rotation

Binding between the controlling keys and the controlled identifier is asserted by one or more CAs.

Security strength or weakness derived not cryptography but from the operational processes of CAs.

DNS provides rented identifiers under centralized control. DNS protocols are insecure due to certain structural security limitations. Domain validation weakness problem: DNS is always vulnerable to attacks that allow an adversary to observe the domain validation probes that CAs send. These can include attacks against the DNS, TCP, or BGP protocols (which lack the cryptographic protections of TLS/SSL), or the compromise of routers. Such attacks are possible either on the network near a CA, or near the victim domain itself.

It is difficult to assure the correctness of the match between data and entity when the data are presented to the CA (perhaps over an electronic network), and when the credentials of the person/company/program asking for a certificate are likewise presented.

Aggregation problem: Identity claims (authenticate with an identifier), attribute claims (submit a bag of vetted attributes), and policy claims are combined in a single container. This raises privacy, policy mapping, and maintenance issues.

Delegation problem: CAs cannot technically restrict subordinate CAs from issuing certificates outside a limited namespaces or attribute set; this feature of X.509 is not in use. Therefore, a large number of CAs exist on the Internet, and classifying them and their policies is an insurmountable task. Delegation of authority within an organization cannot be handled at all, as in common business practice.

Federation problem: Certificate chains that are the result of subordinate CAs, bridge CAs, and cross-signing make validation complex and expensive in terms of processing time. Path validation semantics may be ambiguous. The hierarchy with a third-party trusted party is the only model. This is inconvenient when a bilateral trust relationship is already in place.

DNS/CA is badly broken.

Attempts to secure it without changing its fundamental design is like putting a bandage on a compound fracture.

https://en.wikipedia.org/wiki/X.509

https://en.wikipedia.org/wiki/Certificate_authority

Flaws of original PGP Web-of-Trust as Trust Spanning Layer

No in-band Key Rotation mechanism

Limited supporting protocols (non minimally sufficient support)

Limited supported protocols (all essential applications not supported)

Using the Hourglass Theorem in Protocol Stack Design

Taking a descriptive view of the hourglass allows us to use it as an analytical or predictive tool to understand the impact of a community's adopting a particular interface as a standard, be it de jure or de facto. Making the distinction between the use of the hourglass as a descriptive tool or as a means of justifying a standard also explains how different hourglasses can be examined and compared within the discussion of the same layered system. Every prospective spanning layer has an associated pre- and post-image, regardless of whether it is considered for any kind of standardization.

Pre-image of layer S is the set of services that support S

Post-image of layer S is the set of services that S supports.

(https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/abstract)

Trade-offs

The balance between more applications and more supports is achieved by first choosing the set of necessary applications N and then seeking a spanning layer sufficient for N that is as weak as possible. This scenario makes the choice of necessary applications N the most directly consequential element in the process of defining a spanning layer that meets the goals of the hourglass model.

Note the implication that the tradeoff between the weakness of the spanning layer and its sufficiency for a particular set of applications N is unavoidable. This suggests the design of a spanning layer may have a tendency to fail if it attempts to both achieve a high degree of weakness and also be sufficient to support a large set of necessary applications.

For example: the set of necessary applications may be only verifiably authentic attestations for the application stack and a minimally sufficient routing and discovery mechanism for the routing stack.

Over time broader adoption induced by the broader support likewise induced by the simpler (weaker) spanning layer drives implementers to create more applications and protocol layers that are supported by that spanning layer. The goal is adoptability via most supporting services for only the necessary or essential supported services.

(https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/abstract)

Methodology

The hourglass model can be understood as describing the general shape of the subspace that we navigate in designing layered systems. If one goal is maximizing possible supports, then the Hourglass Theorem tells us that

the slope of the subspace of feasible solutions when considering this goal as a function of the logical weakness of the spanning layer is non-negative. We have no metrics for logical strength or for the size of the space of possible solutions, only for the notions of one service description being weaker than another and one set of service descriptions being included in another.

Resist the temptation to thicken the “spanning layer” in order to support everyone’s preferred application as is. Instead intent everyone to port their application to use the thin “spanning layer” because of broader support below the spanning layer.

This thickening to support existing applications increases interoperability but is not spanning or hourglass interoperability and ultimately results in less adoption not more. Cases in point, DID method proliferation interoperability resulting in thick wallets; Shared ledger proliferation interoperability resulting in thick cross ledger transfer protocols.

From M x N to M+N with the hourglass (M applications and N supports)

(https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/abstract)

(https://www.oilshell.org/blog/2022/02/diagrams.html)

Hourglass vs. End-to-end

The countervailing theory for IP dominance is called the end-to-end theory. End-to-end theory (OSI ISO model) is also about layering in order to encapsulate functionality. Moving a function higher or lower in the protocol stack may or may not weaken or strengthen a lower layer. They are not strongly correlated, end-to-end analysis is therefore less exacting than hourglass analysis for predicting maximum adoption and scalability.

(https://cacm.acm.org/magazines/2019/7/237714-on-the-hourglass-model/abstract)

Weakness Adoption Advantage Examples

IP does not enforce the property of symmetric routable endpoints. This would make it stronger (thicker)

- This weakness enabled DHCP and NAT below to leverage TCP and UDP ports above without changing IP in the middle. Most internet access does not require symmetry. TCP ephemeral ports allows asymmetry so DHCP and NAT were invented to work around too few IPv4 addresses.

Mobile IP leverages multiple routed IP addresses to maintain a persistent connection for mobile IP devices. Cellular networks converted to IP due to the broadening of support given by mobile IP.

IPv6 has enough addresses so no need for NAT and DHCP.

- IPv6 routing includes Mobile IP.
- IPv6 could do away with MAC addresses and make the MAC address an IPv6 service address inside the IPv6 block allocated to a device.
- But IPv6 could have been thinner. Only add address space not add features hence slower IPv6 adoption. Cellular saved IPv6.

HTTP is weaker because it does not support synchronous resource state (full duplex) or peer-to-peer. This simplified HTTP and led to rapid adoption.

- HTTP cache management is a weaker optional solution to synchronizing resource state.
- Later web sockets, HSL, WebRTC were layered on top of HTTP for full duplex peer-to-peer streaming and synchronized resource state.

Internet is a binary protocol, Trust should be a ? protocol

With CESR the Trust spanning layer protocol can eat its cake and have it too.

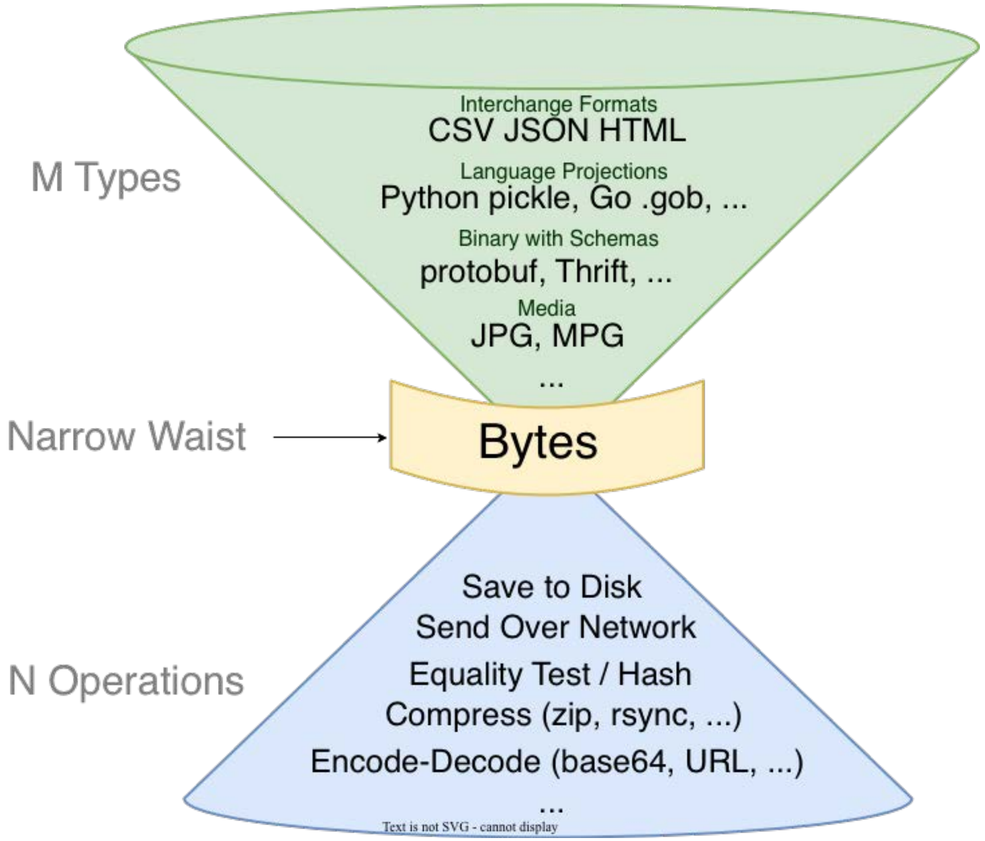
Avoid the text vs. binary protocol wars.

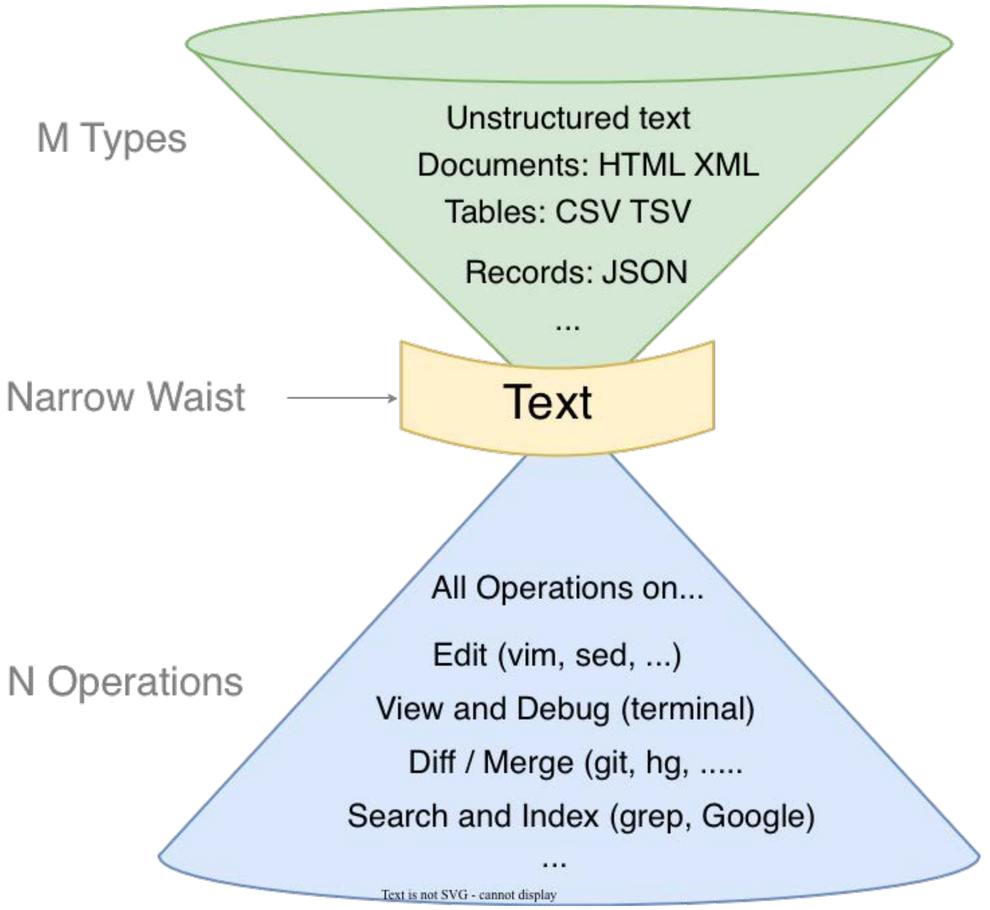
It can be text-native for adoptability but use binary for terseness when beneficial.

CESR Unifies Bytes and Text Hourglasses (Unix)

https://www.oilshell.org/blog/2022/02/diagrams.html

https://www.oilshell.org/cross-ref.html?tag=osh-language#osh-language





Questions

Two Types of Trust

Attributional Trust (“who” said it)
Reputational Trust (“what” was said)
RAP: Reputable Authenticatable Pseudonymity
Cryptographic Pseudonyms (Cryptonyms) enable both security and privacy.
Privacy to the degree unlinked with natural person i.e. a pseudonym.
Securely attributable = authenticatable and accountable
Reputable = Accountable
Identity Graph to manage mutual links between different identifiers and to derive reputation by reference.

Who Blesses Whom (or what)

- AID Controller blesses controlling keys and content addresses (attributional trust via cryptographic secure attribution to AID)
- Trust Anchor blesses other AIDs and content addresses (reputational trust via reference lends credibility and veracity to the what)
- Trust Registry blesses Trust Anchors and other content addresses. (layered reputational trust)

Human *Basis-of-Trust*

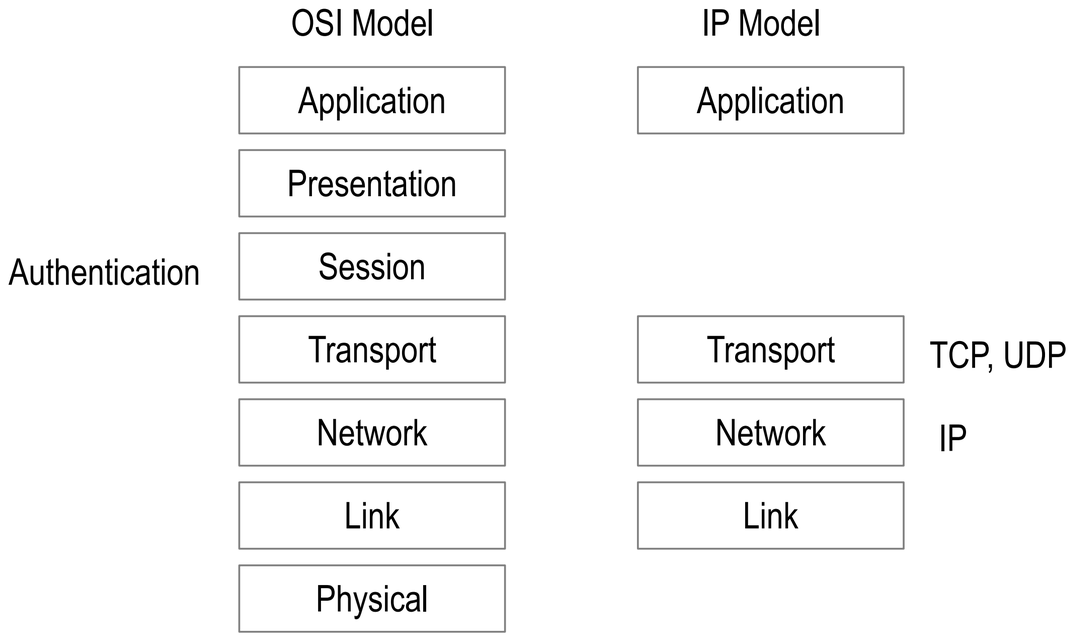
“in person”

I can know you – therefore I can trust you

“on the internet”

I can’t really know you – therefore I can’t really trust you

The Internet Protocol (IP) is *bro-ken* because it has no *security (trust)* layer.



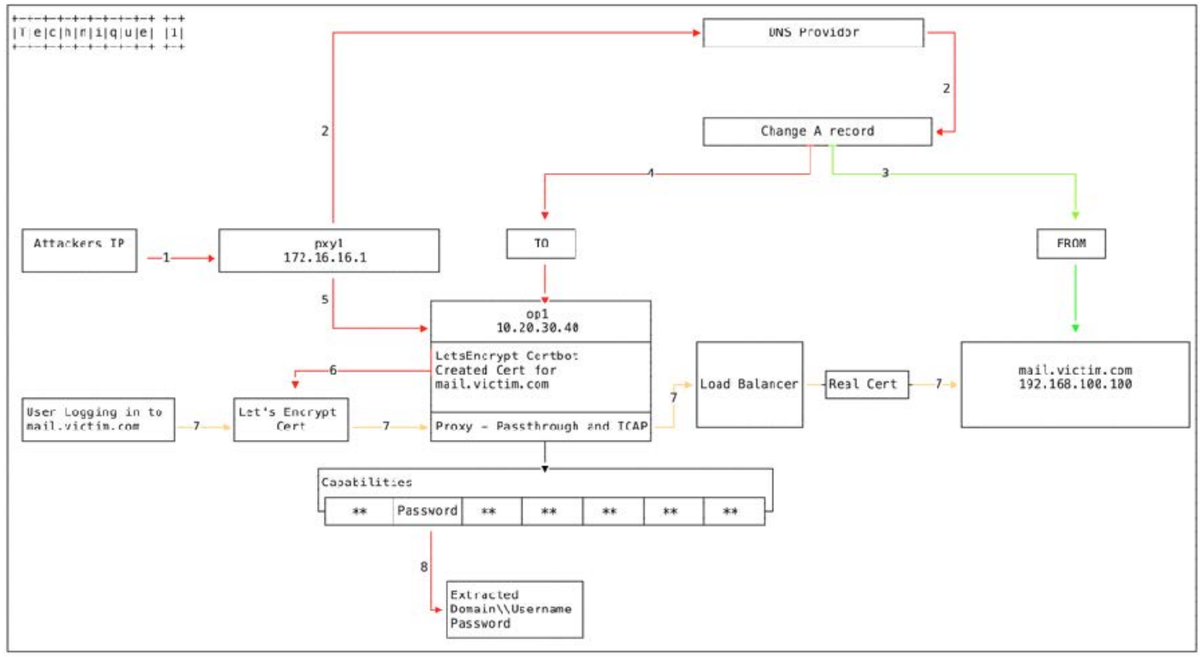
Instead …

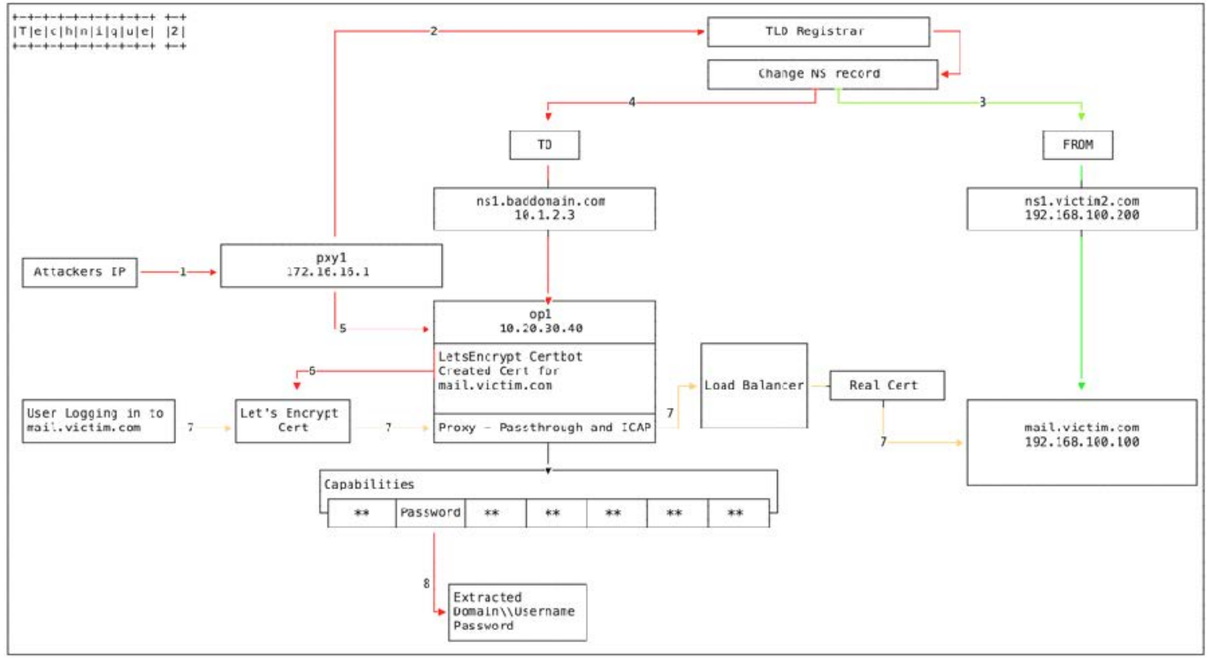
We use *bolt-on* identity system security overlays. (DNS-CA …)

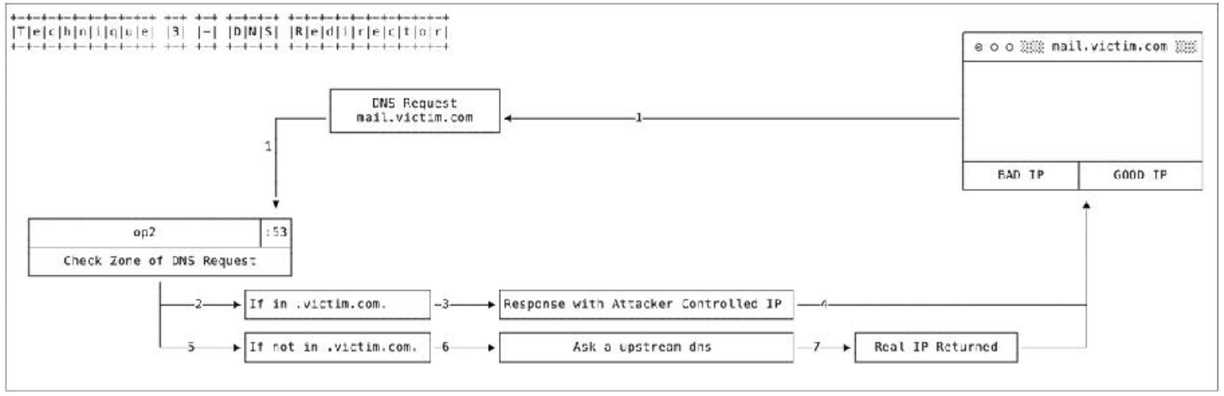
DNS Hijacking

DNS hijacking uses clever tricks that enable attackers to obtain valid TLS certificate for hijacked domains.

https://arstechnica.com/information-technology/2019/01/a-dns-hijacking-wave-is-targeting-companies-at-an-almost-unprecedented-scale/







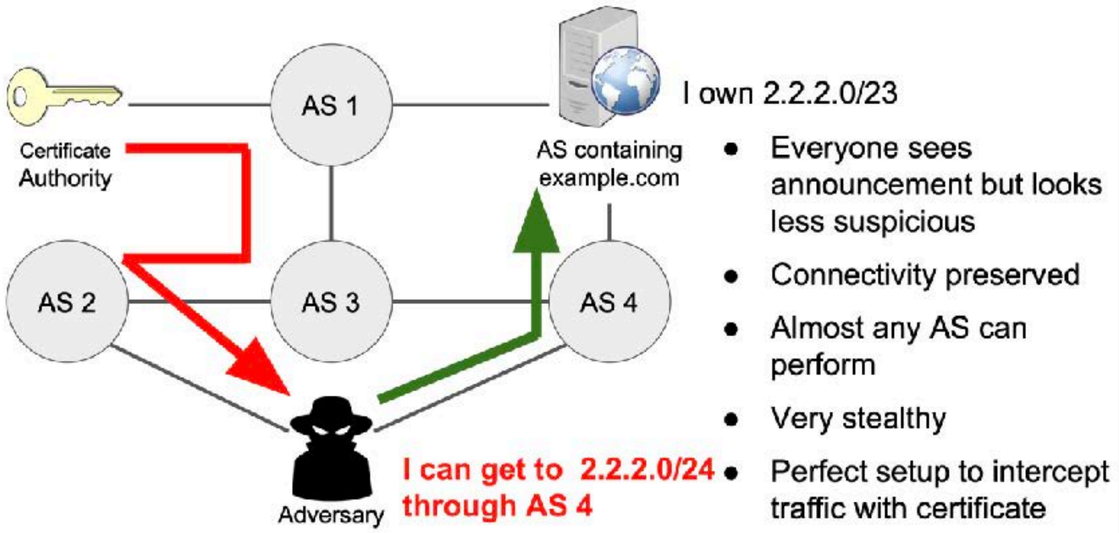
BGP Hijacking: AS Path Poisoning

Spoofing domain verification process from CA enables attackers to obtain valid TLS certificate for hijacked domains.

Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J. and Mittal, P., "Bamboozling certificate authorities with {BGP}," vol. 27th {USENIX} Security Symposium, no. {USENIX} Security 18, pp. 833-849, 2018 https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee

Gavrichenkov, A., "Breaking HTTPS with BGP Hijacking," BlackHat, 2015 https://www.blackhat.com/docs/us-15/materials/us-15-Gavrichenkov-Breaking-HTTPS-With-BGP-Hijacking-wp.pdf

AS path poisoning

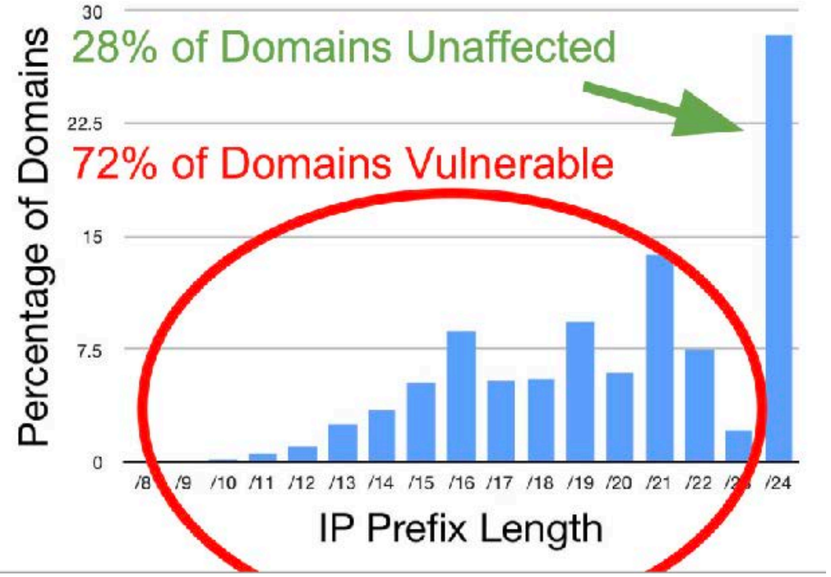


I own 2.2.2.0/23

- Everyone sees announcement but looks less suspicious
- Connectivity preserved
- Almost any AS can perform
- Very stealthy
- Perfect setup to intercept traffic with certificate

Vulnerability of domains: sub-prefix attacks

- Any AS can launch
- Only prefix lengths less than /24 vulnerable (filtering)



Zero-Trust Architecture

Never Trust, Always Verify

Perimeter-less Security Model

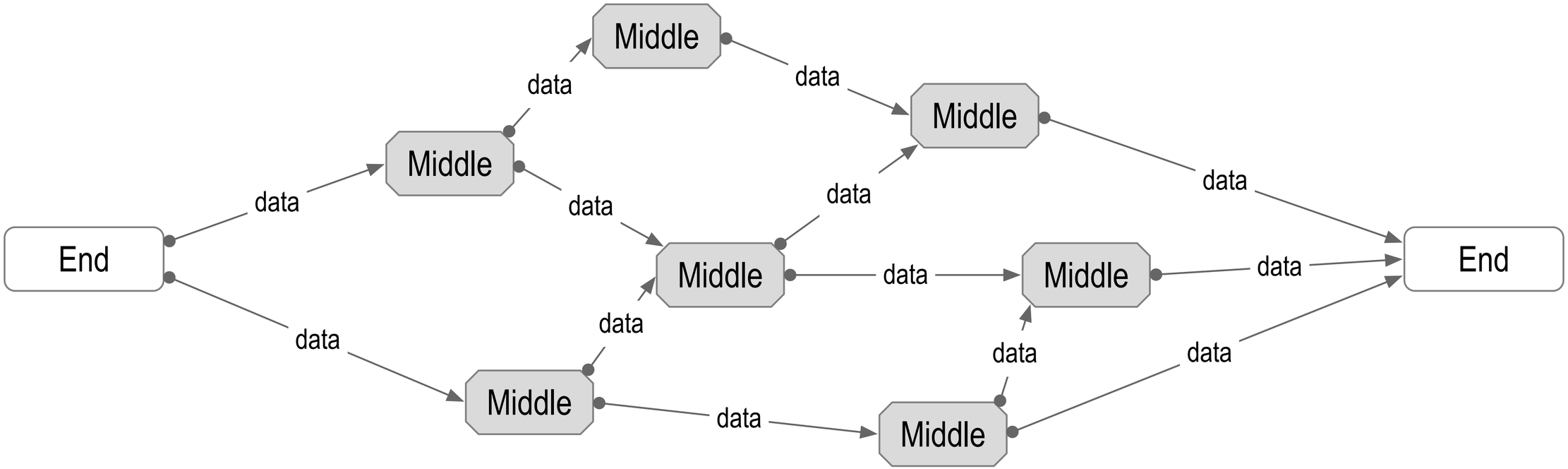
Data is signed and/or encrypted both in motion and at rest.

Zero-Trust Spectrum: ratio of trusted surface to verifiable surface

End goal = all data has end-to-end verifiable authenticity

End Verifiability

End-to-End Verifiability



If the edges are secure, the security of the middle doesn't matter.

Ambient Verifiability: any-data, any-where, any-time by any-body

Zero-Trust-Computing

Its much easier to protect one's private keys than to protect all internet infrastructure

PKI Then and Now

Who uses a password manager?

Who uses an authenticator app?

Who uses password-less login?

Then: Managing private keys impossible for users, federated identity.

Now: Mobile Devices with MFA & secure boot, password-less login.

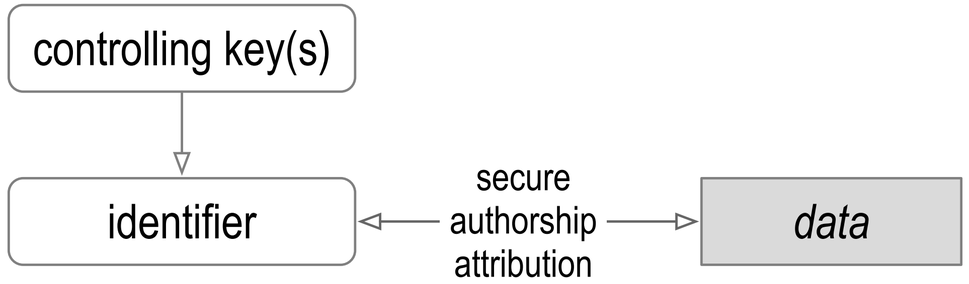
Then: Weak Crypto

Now: Strong crypto: ECC signing & asymmetric encryption.

Then: Perimeter security, no persistence of control over identifiers.

Now: authentic web and zero-trust architecture for identity.

Flaw of PKI (DNS/CA)



Use of private keys exposes them to side-channel attack.

Over-time, exposure makes private keys weak.

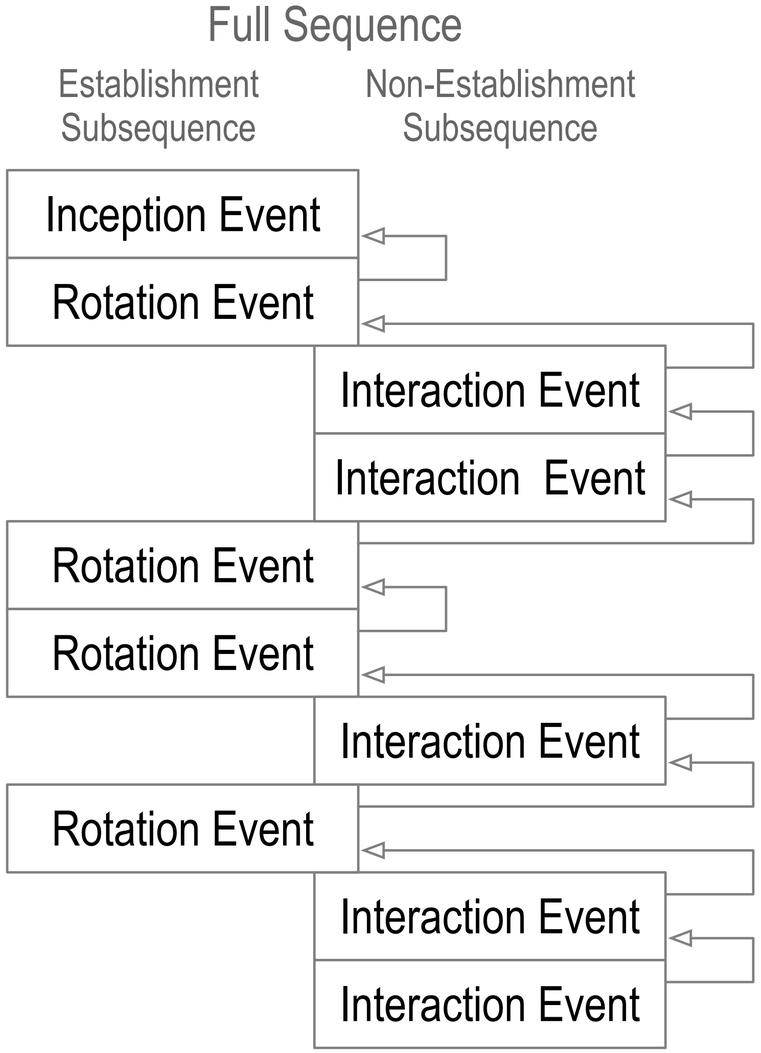
Thus, from time-to-time one must revoke and replace the controlling private keys for a given identifier

Hence key rotation

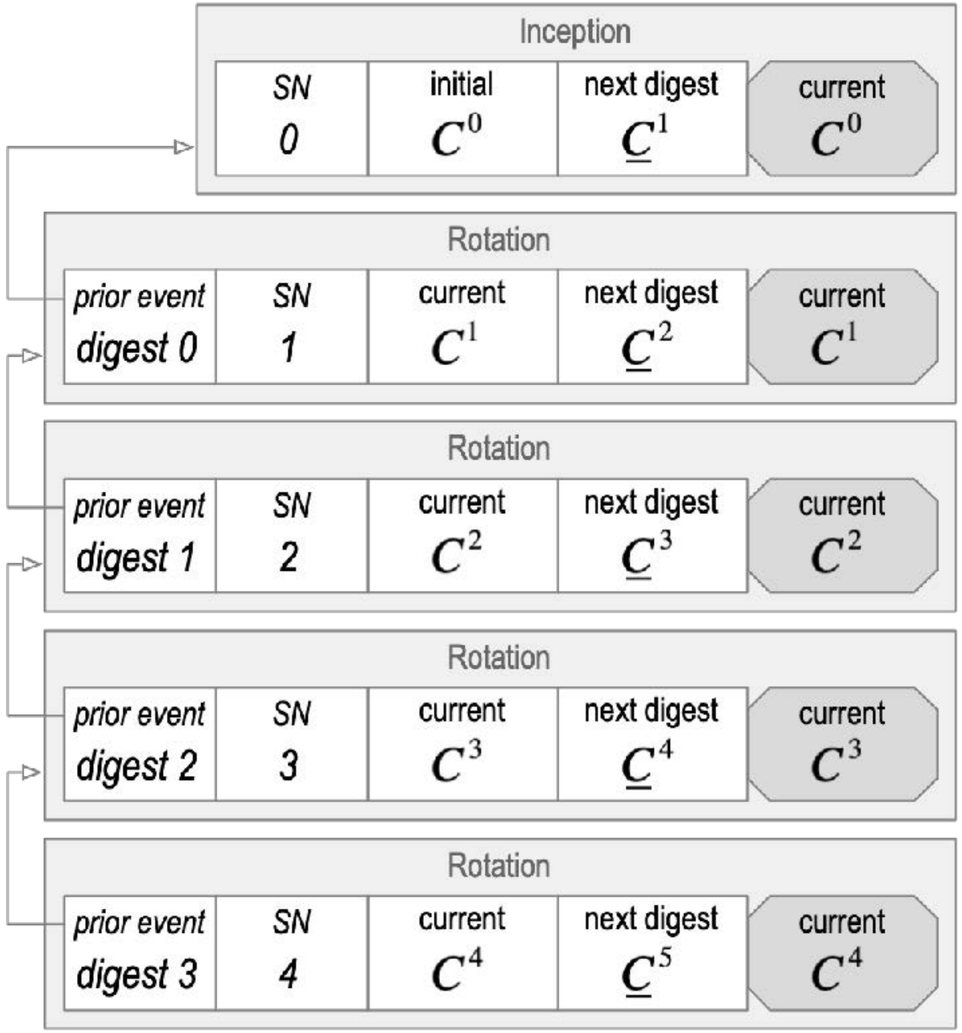
Existing PKI must re-establish the root-of-trust with each rotation thereby making it vulnerable to attack

Breaks the chain-of-trust-of-control over the identifier

Solution: Key Pre-Rotation



duplicity evident
verifiable data structure



Digest of *next* key(s) makes pre-rotation post-quantum secure

D.J. Bernstein: https://cr.yp.to/hash/collisioncost-20090517.pdf