Internet Architecture

ECE/CS598HPN

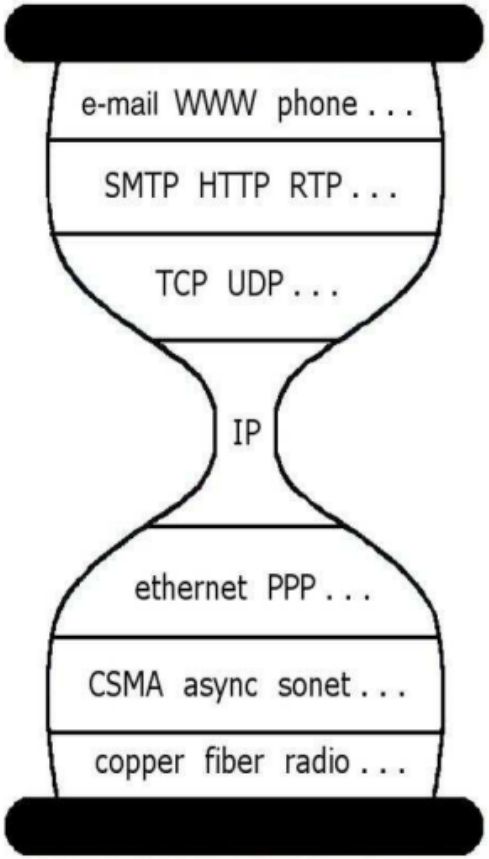
Radhika Mittal

What is Internet Architecture?

- How an endhost identifies and specifies the destination.
- How routers understand that specification to forward packets to the destination over the Internet.

- *Carried out by L3 (IP).*

IP as the narrow waist



- Facilitated a lot of innovation above and below IP.
- *Hard to change IP itself.*

Security

- Clean-slate architecture.
- Establishes *trust domains*.
- Guarantee control plane isolation for trust domains.

SCION: Scalability, Control, and Isolation On Next-Generation Networks

Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig and David G. Andersen
CyLab / Carnegie Mellon University

***Abstract*—We present the first Internet architecture designed to provide route control, failure isolation, and explicit trust information for end-to-end communications. SCION separates ASes into groups of independent routing sub-planes, called *trust domains*, which then interconnect to form complete routes. Trust domains provide natural isolation of routing failures and human misconfiguration, give endpoints strong control for both inbound and outbound traffic, provide meaningful and enforceable trust, and enable scalable routing updates with high path freshness. As a result, our architecture provides strong resilience and security properties as an intrinsic consequence of good design principles, avoiding piecemeal add-on protocols as security patches. Meanwhile, SCION only assumes that a few top-tier ISPs in the trust domain are trusted for providing reliable end-to-end communications, thus achieving a small Trusted Computing Base. Both our security analysis and evaluation results show that SCION naturally prevents numerous attacks and provides a high level of resilience, scalability, control, and isolation.**

I. Introduction

The Internet is the most geographically, administratively, and socially diverse distributed system ever invented. While today's Internet architecture admits some administrative diversity, such as by separating routing inside a domain (intra-AS routing) from global inter-domain routing, it falls short in handling the key challenges of security and isolation that arise in this intensely heterogeneous setting. As a result, we see surprisingly frequent incidents in which communication is interrupted by actions or actors far from the communicating entities. In addition to classical examples such as YouTube being globally disrupted by routing announcements from Pakistan [1], other issues surrounding the lack of resource control and isolation are not solved by existing proposals such as S-BGP [2]: the introduction of excessive routing churn [3]; traffic flooding; and even issues of global conflicts over naming and name resolution.

This paper proposes a clean-slate Internet architecture, SCION, that provides strong guarantees for failure isolation and route control in ways that map well to existing geographic, political, and legal boundaries. We show that strong control and isolation naturally leads to security and reliability without the use of high-overhead security mechanisms, while exposing to the endpoints diverse communication path sets that can support a wide spectrum of routing policies and path preferences (*path expressiveness*).

We introduce the notion of a hierarchy of *trust domains* whose members all share a common contractual, legal, cultural, geographical, or other basis for extending limited trust among each other. Examples may be a domain of U.S. commercial and educational institutions, ISPs that participate in the same peering point who share a common, binding legal contract on their behavior, or ISPs in the same state or country who are subject to the same laws and regulations. Using this abstraction, we provide the machinery to guarantee control-plane isolation: *Entities outside a trust domain cannot affect control-plane computation and communication within that trust domain.* For communication that must span trust domains, we provide the property that *the entities who can affect the communication are limited to a necessary and explicitly identified set of other trust domains.* We leave data-plane security as future work and thus do not consider denial of service attacks. In addition, the introduction of trust domains enables sources, transit ISPs, and destinations in SCION to agree *jointly* on which path to use. The architecture naturally controls routing information flow, and provides for explicit trust in path selection.

Through isolation and control, SCION enables expressive trust, i.e., *all the communicating endpoints can decide and control explicitly and precisely whom they need to trust for providing reliable communications*. Exposing such explicit trust information for end-to-end communication can eventually benefit network availability, because the endpoints can select more "trusted" communication paths with presumably more reliable data delivery; or at least, SCION holds the parties involved in the communications accountable for their misbehavior and failures.

Contributions. We design and analyze SCION, an Internet architecture emphasizing the principles of control, isolation and explicit trust. SCION enables route control for ISPs, senders and receivers at an appropriate level of granularity, balancing efficiency, expressiveness, policy compliance, and security. The isolation properties dramatically shrink the TCB and make explicit which entities communication relies upon. SCION offers strong security properties and demonstrates that the resulting routes widely mirror those in place under BGP today. We anticipate that the proposed architecture offers a useful design point for a next-generation Internet.

This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389, W911NF-09-1-0273 and W911NF-0710287 from the Army Research Office, and by NSF under awards CNS-1040801, CNS-1050224, ANI-0331653, and CNS-0520187. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, CyLab, or the U.S. Government or any of its agencies.

Security

- Architecture to limit DoS attacks.
- Receivers grant sending *capabilities* to senders.
- Routers check for this capability to determine if the packet is wanted.

TVA: a DoS-limiting Network Architecture

Xiaowei Yang, *Member,* David Wetherall, *Member,* Thomas Anderson, *Member*

***Abstract*—We motivate the capability approach to network denial-of-service (DoS) attacks, and evaluate the TVA architecture which builds on capabilities. With our approach, rather than send packets to any destination at any time, senders must first obtain "permission to send" from the receiver, which provides the permission in the form of capabilities to those senders whose traffic it agrees to accept. The senders then include these capabilities in packets. This enables verification points distributed around the network to check that traffic has been authorized by the receiver and the path in between, and hence to cleanly discard unauthorized traffic. To evaluate this approach, and to understand the detailed operation of capabilities, we developed a network architecture called TVA. TVA addresses a wide range of possible attacks against communication between pairs of hosts, including spoofed packet floods, network and host bottlenecks, and router state exhaustion. We use simulations to show the effectiveness of TVA at limiting DoS floods, and an implementation on Click router to evaluate the computational costs of TVA. We also discuss how to incrementally deploy TVA into practice.**

I. Introduction

The Internet owes much of its historic success and growth to its openness to new applications. A key design feature of the Internet is that any application can send anything to anyone at any time, without needing to obtain advance permission from network administrators. New applications can be designed, implemented and come into widespread use much more quickly, if they do not need to wait for key features to be added to the underlying network.

Quietly, however, the Internet has become much less open to new applications over the past few years. Perversely, this has happened as a rational response of network and system administrators needing to cope with the consequences of the Internet's openness. The Internet architecture is vulnerable to denial-of-service (DoS) attacks, where any collection of hosts with enough bandwidth (e.g., using machines taken over by a virus attack) can disrupt legitimate communication between any pair of other parties, simply by flooding one end or the other with unwanted traffic. These attacks are widespread, increasing, and have proven resistant to all attempts to stop them [26].

Operationally, to deal with persistent and repeated DoS and virus attacks, network and system administrators have begun to deploy automated response systems to look for anomalous behavior that might be an attack. When alarms are triggered, often by legitimate traffic, the operational response is typically to "stop everything and ask questions later." Unfortunately, any new application is likely to appear to be anomalous! Our experience with this comes from operating and using the PlanetLab testbed, which is designed to make it easy to develop new, geographically distributed, Internet applications [27]. On several occasions, we have observed innocuous, low-rate traffic from a single application trigger alarms that completely disconnected entire universities from the Internet. Since alarm rules are by nature secret, the only way to guarantee that a new application does not trigger an alarm (and the resulting disproportionate response) is to make its traffic look identical to some existing application. In other words, the only safe thing to do is to precisely mimic an old protocol.

The openness of the Internet is likely to erode if there is no effective solution to eliminate large scale DoS attacks. Attackers are winning the arms race with anomaly detection by making their traffic look increasingly like normal traffic. The CodeRed and follow-on viruses have demonstrated repeatedly that it is possible to recruit millions of machines to the task of sending normal HTTP requests to a single destination [24], [25]. This problem is fundamental to the Internet architecture: no matter how over-provisioned you are, if everyone in the world sends you a single packet, legitimate traffic will not get through.

We argue for taking a step back, to ask how, at an architectural level, we can address the DoS problem in its entirety while still allowing new applications to be deployed. Our goal, in essence, is to let any two nodes exchange whatever traffic they like (subject to bandwidth constraints of intermediate links), such that no set of third parties can disrupt that traffic exchange.

Our approach is based on the notion of capabilities, which are short-term authorizations that senders obtain from receivers and stamp on their packets. This allows senders to control the traffic that they receive. Our attraction to capabilities is that they cut to the heart of the DoS problem by allowing unwanted traffic to be removed in the network, but do so in an open manner by providing destinations with the control over which traffic is filtered. However, while capabilities may be an appealing approach, they leave many questions unanswered, such as how capabilities are granted without being vulnerable to attack.

To answer these questions and help evaluate the capability approach, we have designed and prototyped the Traffic Validation Architecture (TVA[1]). TVA is a DoS-limiting network architecture that details the operation of capabilities and combines mechanisms that counter a broad set of possible denial-of-service attacks, including those that flood the setup channel, that exhaust router state, that consume network bandwidth, and so forth. The design that we present in this paper is a revision of our earlier work [35] that pays greater attention to protecting the capability request channel.

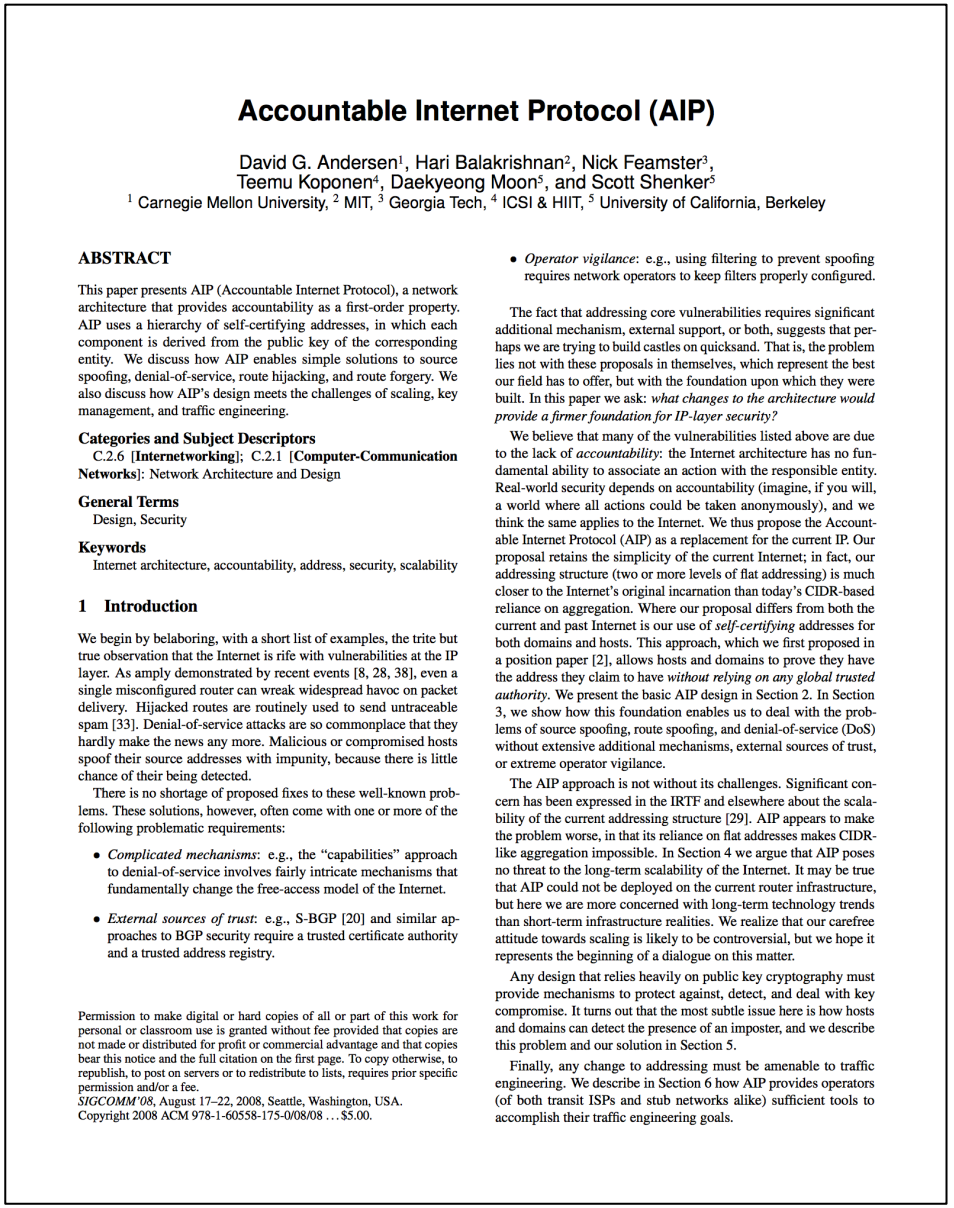
We have designed TVA to be practical in three key respects. First, we bound both the computation and state needed to process capabilities. Second, we have designed our system to be incrementally deployable in the current Internet. This can be done by placing inline packet processing boxes at trust boundaries and points of congestion, and upgrading collections of hosts to take advantage of them. No changes to Internet

Xiaowei Yang is with University of California at Irvine; David Wetherall is with both University of Washington and Intel Research Seattle. Thomas Anderson is with University of Washington. This work was supported in part by the NSF (Grant CNS-0430304 and Grant CNS-0627787).

[1]The name TVA is inspired by the Tennessee Valley Authority, which operates a large-scale network of dams to control flood damage, saving more than $200 million annually.

Accountability

- Make Internet addressing more accountable.
- Use *self-certifying* host addresses.



Accountable Internet Protocol (AIP)

David G. Andersen[1], Hari Balakrishnan[2], Nick Feamster[3],
Teemu Koponen[4], Daekyeong Moon[5], and Scott Shenker[5]
[1] Carnegie Mellon University, [2] MIT, [3] Georgia Tech, [4] ICSI & HIIT, [5] University of California, Berkeley

ABSTRACT

This paper presents AIP (Accountable Internet Protocol), a network architecture that provides accountability as a first-order property. AIP uses a hierarchy of self-certifying addresses, in which each component is derived from the public key of the corresponding entity. We discuss how AIP enables simple solutions to source spoofing, denial-of-service, route hijacking, and route forgery. We also discuss how AIP's design meets the challenges of scaling, key management, and traffic engineering.

Categories and Subject Descriptors
C.2.6 [**Internetworking**]; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms
Design, Security

Keywords
Internet architecture, accountability, address, security, scalability

1 Introduction

We begin by belaboring, with a short list of examples, the trite but true observation that the Internet is rife with vulnerabilities at the IP layer. As amply demonstrated by recent events [8, 28, 38], even a single misconfigured router can wreak widespread havoc on packet delivery. Hijacked routes are routinely used to send untraceable spam [33]. Denial-of-service attacks are so commonplace that they hardly make the news any more. Malicious or compromised hosts spoof their source addresses with impunity, because there is little chance of their being detected.

There is no shortage of proposed fixes to these well-known problems. These solutions, however, often come with one or more of the following problematic requirements:

- *Complicated mechanisms*: e.g., the "capabilities" approach to denial-of-service involves fairly intricate mechanisms that fundamentally change the free-access model of the Internet.
- *External sources of trust*: e.g., S-BGP [20] and similar approaches to BGP security require a trusted certificate authority and a trusted address registry.
- *Operator vigilance*: e.g., using filtering to prevent spoofing requires network operators to keep filters properly configured.

The fact that addressing core vulnerabilities requires significant additional mechanism, external support, or both, suggests that perhaps we are trying to build castles on quicksand. That is, the problem lies not with these proposals in themselves, which represent the best our field has to offer, but with the foundation upon which they were built. In this paper we ask: *what changes to the architecture would provide a firmer foundation for IP-layer security?*

We believe that many of the vulnerabilities listed above are due to the lack of *accountability*: the Internet architecture has no fundamental ability to associate an action with the responsible entity. Real-world security depends on accountability (imagine, if you will, a world where all actions could be taken anonymously), and we think the same applies to the Internet. We thus propose the Accountable Internet Protocol (AIP) as a replacement for the current IP. Our proposal retains the simplicity of the current Internet; in fact, our addressing structure (two or more levels of flat addressing) is much closer to the Internet's original incarnation than today's CIDR-based reliance on aggregation. Where our proposal differs from both the current and past Internet is our use of *self-certifying* addresses for both domains and hosts. This approach, which we first proposed in a position paper [2], allows hosts and domains to prove they have the address they claim to have *without relying on any global trusted authority*. We present the basic AIP design in Section 2. In Section 3, we show how this foundation enables us to deal with the problems of source spoofing, route spoofing, and denial-of-service (DoS) without extensive additional mechanisms, external sources of trust, or extreme operator vigilance.

The AIP approach is not without its challenges. Significant concern has been expressed in the IRTF and elsewhere about the scalability of the current addressing structure [29]. AIP appears to make the problem worse, in that its reliance on flat addresses makes CIDR-like aggregation impossible. In Section 4 we argue that AIP poses no threat to the long-term scalability of the Internet. It may be true that AIP could not be deployed on the current router infrastructure, but here we are more concerned with long-term technology trends than short-term infrastructure realities. We realize that our carefree attitude towards scaling is likely to be controversial, but we hope it represents the beginning of a dialogue on this matter.

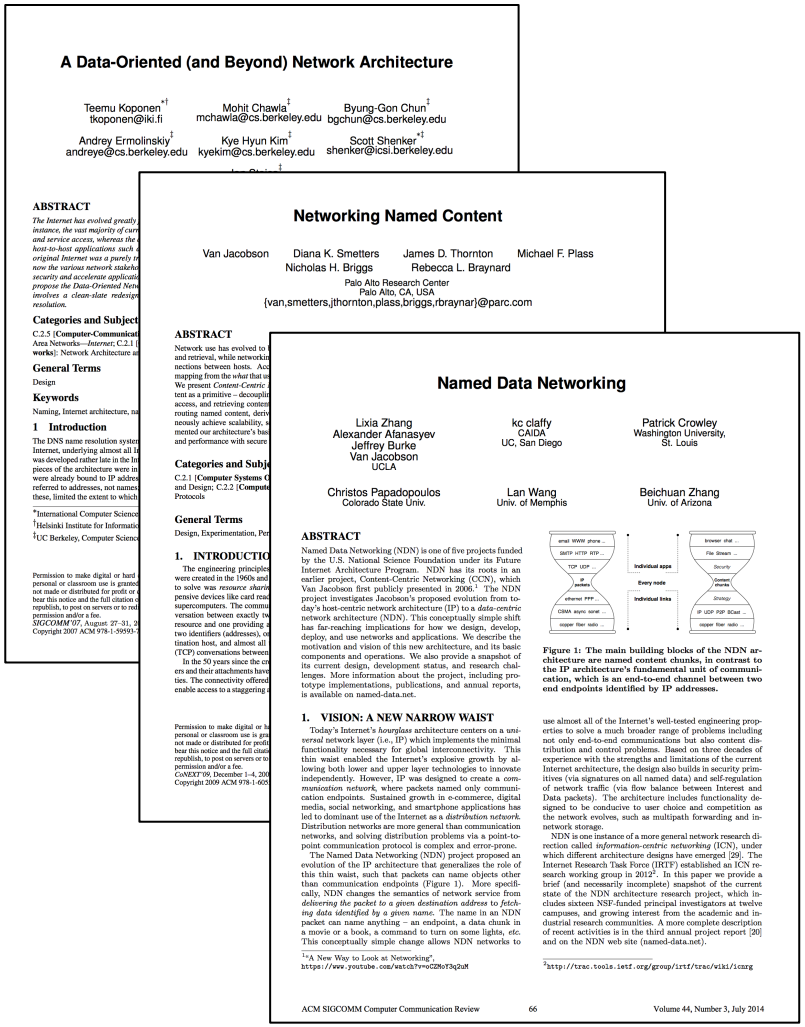
Any design that relies heavily on public key cryptography must provide mechanisms to protect against, detect, and deal with key compromise. It turns out that the most subtle issue here is how hosts and domains can detect the presence of an imposter, and we describe this problem and our solution in Section 5.

Finally, any change to addressing must be amenable to traffic engineering. We describe in Section 6 how AIP provides operators (of both transit ISPs and stub networks alike) sufficient tools to accomplish their traffic engineering goals.

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

Information-centric Networking

- Name *bits* (data or content) instead of *locations*.
- Self-certified or signed in some manner.



A Data-Oriented (and Beyond) Network Architecture

Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker

Networking Named Content

Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, Rebecca L. Braynard

Palo Alto Research Center
Palo Alto, CA, USA
{van,smetters,jthornton,plass,briggs,rbraynar}@parc.com

Named Data Networking

Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson
UCLA

kc claffy
CAIDA
UC, San Diego

Patrick Crowley
Washington University,
St. Louis

Christos Papadopoulos
Colorado State Univ.

Lan Wang
Univ. of Memphis

Beichuan Zhang
Univ. of Arizona

ABSTRACT

Named Data Networking (NDN) is one of five projects funded by the U.S. National Science Foundation under its Future Internet Architecture Program. NDN has its roots in an earlier project, Content-Centric Networking (CCN), which Van Jacobson first publicly presented in 2006.[1] The NDN project investigates Jacobson's proposed evolution from today's host-centric network architecture (IP) to a *data-centric* network architecture (NDN). This conceptually simple shift has far-reaching implications for how we design, develop, deploy, and use networks and applications. We describe the motivation and vision of this new architecture, and its basic components and operations. We also provide a snapshot of its current design, development status, and research challenges. More information about the project, including prototype implementations, publications, and annual reports, is available on named-data.net.

Figure 1: The main building blocks of the NDN architecture are named content chunks, in contrast to the IP architecture's fundamental unit of communication, which is an end-to-end channel between two end endpoints identified by IP addresses.

1. VISION: A NEW NARROW WAIST

Today's Internet's *hourglass* architecture centers on a *universal* network layer (i.e., IP) which implements the minimal functionality necessary for global interconnectivity. This thin waist enabled the Internet's explosive growth by allowing both lower and upper layer technologies to innovate independently. However, IP was designed to create a *communication network*, where packets named only communication endpoints. Sustained growth in e-commerce, digital media, social networking, and smartphone applications has led to dominant use of the Internet as a *distribution network*. Distribution networks are more general than communication networks, and solving distribution problems via a point-to-point communication protocol is complex and error-prone.

The Named Data Networking (NDN) project proposed an evolution of the IP architecture that generalizes the role of this thin waist, such that packets can name objects other than communication endpoints (Figure 1). More specifically, NDN changes the semantics of network service from *delivering the packet to a given destination address* to *fetching data identified by a given name*. The name in an NDN packet can name anything – an endpoint, a data chunk in a movie or a book, a command to turn on some lights, *etc.* This conceptually simple change allows NDN networks to use almost all of the Internet's well-tested engineering properties to solve a much broader range of problems including not only end-to-end communications but also content distribution and control problems. Based on three decades of experience with the strengths and limitations of the current Internet architecture, the design also builds in security primitives (via signatures on all named data) and self-regulation of network traffic (via flow balance between Interest and Data packets). The architecture includes functionality designed to be conducive to user choice and competition as the network evolves, such as multipath forwarding and in-network storage.

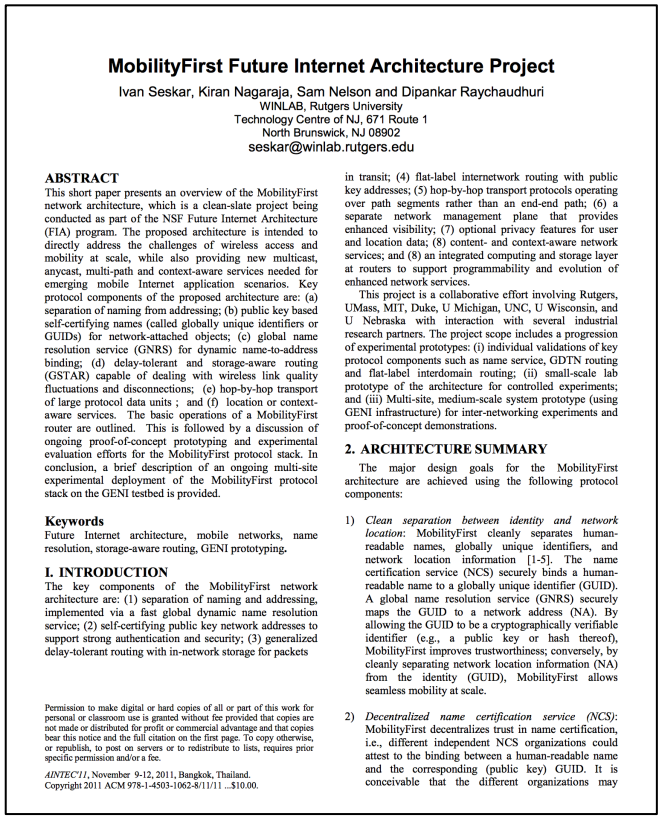
NDN is one instance of a more general network research direction called *information-centric networking* (ICN), under which different architecture designs have emerged [29]. The Internet Research Task Force (IRTF) established an ICN research working group in 2012[2]. In this paper we provide a brief (and necessarily incomplete) snapshot of the current state of the NDN architecture research project, which includes sixteen NSF-funded principal investigators at twelve campuses, and growing interest from the academic and industrial research communities. A more complete description of recent activities is in the third annual project report [20] and on the NDN web site (named-data.net).

[1] "A New Way to Look at Networking", https://www.youtube.com/watch?v=oCZMoY3q2uM

[2] http://trac.tools.ietf.org/group/irtf/trac/wiki/icnrg

ACM SIGCOMM Computer Communication Review 66 Volume 44, Number 3, July 2014

Other forms of addressing



MobilityFirst Future Internet Architecture Project

Ivan Seskar, Kiran Nagaraja, Sam Nelson and Dipankar Raychaudhuri
WINLAB, Rutgers University
Technology Centre of NJ, 671 Route 1
North Brunswick, NJ 08902
seskar@winlab.rutgers.edu

ABSTRACT
This short paper presents an overview of the MobilityFirst network architecture, which is a clean-slate project being conducted as part of the NSF Future Internet Architecture (FIA) program. The proposed architecture is intended to directly address the challenges of wireless access and mobility at scale, while also providing new multicast, anycast, multi-path and context-aware services needed for emerging mobile Internet application scenarios. Key protocol components of the proposed architecture are: (a) separation of naming from addressing; (b) public key based self-certifying names (called globally unique identifiers or GUIDs) for network-attached objects; (c) global name resolution service (GNRS) for dynamic name-to-address binding; (d) delay-tolerant and storage-aware routing (GSTAR) capable of dealing with wireless link quality fluctuations and disconnections; (e) hop-by-hop transport of large protocol data units ; and (f) location or context-aware services. The basic operations of a MobilityFirst router are outlined. This is followed by a discussion of ongoing proof-of-concept prototyping and experimental evaluation efforts for the MobilityFirst protocol stack. In conclusion, a brief description of an ongoing multi-site experimental deployment of the MobilityFirst protocol stack on the GENI testbed is provided.

Keywords
Future Internet architecture, mobile networks, name resolution, storage-aware routing, GENI prototyping.

I. INTRODUCTION
The key components of the MobilityFirst network architecture are: (1) separation of naming and addressing, implemented via a fast global dynamic name resolution service; (2) self-certifying public key network addresses to support strong authentication and security; (3) generalized delay-tolerant routing with in-network storage for packets in transit; (4) flat-label internetwork routing with public key addresses; (5) hop-by-hop transport protocols operating over path segments rather than an end-end path; (6) a separate network management plane that provides enhanced visibility; (7) optional privacy features for user and location data; (8) content- and context-aware network services; and (8) an integrated computing and storage layer at routers to support programmability and evolution of enhanced network services.

This project is a collaborative effort involving Rutgers, UMass, MIT, Duke, U Michigan, UNC, U Wisconsin, and U Nebraska with interaction with several industrial research partners. The project scope includes a progression of experimental prototypes: (i) individual validations of key protocol components such as name service, GDTN routing and flat-label interdomain routing; (ii) small-scale lab prototype of the architecture for controlled experiments; and (iii) Multi-site, medium-scale system prototype (using GENI infrastructure) for inter-networking experiments and proof-of-concept demonstrations.

2. ARCHITECTURE SUMMARY
The major design goals for the MobilityFirst architecture are achieved using the following protocol components:

1) *Clean separation between identity and network location*: MobilityFirst cleanly separates human-readable names, globally unique identifiers, and network location information [1-5]. The name certification service (NCS) securely binds a human-readable name to a globally unique identifier (GUID). A global name resolution service (GNRS) securely maps the GUID to a network address (NA). By allowing the GUID to be a cryptographically verifiable identifier (e.g., a public key or hash thereof), MobilityFirst improves trustworthiness; conversely, by cleanly separating network location information (NA) from the identity (GUID), MobilityFirst allows seamless mobility at scale.

2) *Decentralized name certification service (NCS)*: MobilityFirst decentralizes trust in name certification, i.e., different independent NCS organizations could attest to the binding between a human-readable name and the corresponding (public key) GUID. It is conceivable that the different organizations may

AINTECT11, November 9-12, 2011, Bangkok, Thailand.

Replace IP addresses with location-independent, flat, globally unique IDs.

Serval: An End-Host Stack for Service-Centric Networking

Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer
Matvey Arye, Steven Y. Ko, Jennifer Rexford, Michael J. Freedman
Princeton University

Abstract
Internet services run on multiple servers in different locations, serving clients that are often mobile and multi-homed. This does not match well with today's network stack, designed for communication between fixed hosts with topology-dependent addresses. As a result, online service providers resort to clumsy and management-intensive work-arounds—forfeiting the scalability of hierarchical addressing to support virtual server migration, directing all client traffic through dedicated load balancers, restarting connections when hosts move, and so on.

In this paper, we revisit the design of the network stack to meet the needs of online services. The centerpiece of our Serval architecture is a new Service Access Layer (SAL) that sits above an unmodified network layer, and enables applications to communicate directly on service names. The SAL provides a clean *service-level* control/data plane split, enabling policy, control, and in-stack name-based routing that connects clients to services via diverse discovery techniques. By tying *active sockets* to the control plane, applications trigger updates to service routing state upon invoking socket calls, ensuring up-to-date service resolution. With Serval, end-points can seamlessly change network addresses, migrate flows across interfaces, or establish additional flows for efficient and uninterrupted service access. Experiments with our high-performance in-kernel prototype, and several example applications, demonstrate the value of a unified networking solution for online services.

1. Introduction
The Internet is increasingly a platform for accessing services that run anywhere, from servers in the datacenter and computers at home, to the mobile phone in one's pocket and a sensor in the field. An application can run on multiple servers at different locations, and can launch at or migrate to a new machine at any time. In addition, user devices are often multi-homed (*e.g.*, WiFi and 4G) and mobile. In short, modern services operate under unprecedented *multiplicity* (in service replicas, host interfaces, and network paths) and *dynamism* (due to replica failure and recovery, service migration, and client mobility).

Yet, multiplicity and dynamism match poorly with today's host-centric TCP/IP-stack that binds connections to fixed attachment points with topology-dependent addresses and conflates service, flow, and network identifiers. This forces online services to rely on clumsy and restrictive techniques that manipulate the network layer and constrain how services are composed, managed, and controlled. For example, today's load balancers repurpose IP addresses to refer to a group of (possibly changing) service instances; unfortunately, this requires all client traffic to traverse the load balancer. Techniques for handling mobility and migration are either limited to a single layer-2 domain or introduce "triangle routing." Hosts typically cannot spread a connection over multiple interfaces or paths, and changing interfaces requires the initiation of new connections. The list goes on and on.

To address these problems, we present the Serval architecture that runs on top of an unmodified network layer. Serval provides a service-aware network stack, where applications communicate directly on *service names* instead of addresses and ports. A service name corresponds to a group of (possibly changing) processes offering the same service. This elevates services to first-class network entities (distinct from hosts or interfaces), and decouples services from network and flow identifiers. Hence, service names identify *who* one communicates with, flow names identify *what* communication context to use, while addresses tell *where* to direct the communication.

At the core of Serval is a new Service Access Layer (SAL) that sits between the transport and network layers. The SAL maps service names in packets to network addresses, based on rules in its *service table* (analogous to how the network layer uses a forwarding table). Unlike traditional "service layers," which sit *above* the transport layer, the SAL's position *below* transport provides a programmable *service-level* data plane that can adopt diverse service discovery techniques. The SAL can be programmed through a user-space control plane, acting on service-level events triggered by *active sockets* (*e.g.*, a service instance automatically registers on `binding` a socket). This gives network programmers hooks for ensuring service-resolution systems are up-to-date.

As such, Serval gives service providers more control over service access, and clients more flexibility in resolving services. For instance, by forwarding the first packet of a connection based on service name, the SAL can defer binding a service until the packet reaches the part of the network with fine-grain, up-to-date information. This

Replace IP addresses (and ports) with *service names*.

Source routing

Pathlet Routing

P. Brighten Godfrey[†], Igor Ganichev[‡], Scott Shenker[‡§], and Ion Stoica[‡*]
[†]University of Illinois at Urbana-Champaign [‡]UC Berkeley [§]ICSI
pbg@illinois.edu, {igor,shenker,istoica}@cs.berkeley.edu

ABSTRACT

We present a new routing protocol, pathlet routing, in which networks advertise fragments of paths, called pathlets, that sources concatenate into end-to-end source routes. Intuitively, the pathlet is a highly flexible building block, capturing policy constraints as well as enabling an exponentially large number of path choices. In particular, we show that pathlet routing can emulate the policies of BGP, source routing, and several recent multipath proposals.

This flexibility lets us address two major challenges for Internet routing: scalability and source-controlled routing. When a router's routing policy has only "local" constraints, it can be represented using a small number of pathlets, leading to very small forwarding tables and many choices of routes for senders. Crucially, pathlet routing does not impose a global requirement on what style of policy is used, but rather allows multiple styles to coexist. The protocol thus supports complex routing policies while enabling and incentivizing the adoption of policies that yield small forwarding plane state and a high degree of path choice.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet-switching networks; C.2.2 [**Network Protocols**]: Routing Protocols; C.2.6 [**Internetworking**]: Routers

General Terms

Design, Experimentation, Performance, Reliability

1. INTRODUCTION

Challenges for interdomain routing. Interdomain routing faces several fundamental challenges. One is *scalability*: routers running the Internet's interdomain routing protocol, Border Gateway Protocol (BGP) [25], require state that scales linearly in the number of IP prefixes advertised in the Internet. This is particularly a concern in the data plane where the router stores the routing table, or forwarding information base (FIB). Because it has to operate at high speeds and often uses SRAM rather than commodity DRAM, FIB memory is arguably more constrained and expensive than other resources in a router [22]. Moreover, the number of IP prefixes is increasing at an increasing rate [15], leading to the need for expensive hardware and upgrades. The Internet Architecture Board Workshop on Routing and Addressing recently identified FIB growth as one of the key concerns for future scalability of the routing system [22].

A second challenge for interdomain routing is to provide *multipath routing*, in which a packet's source (an end host or edge router) selects its path from among multiple options. For network users, multipath routing is a solution to two important deficiencies of BGP: poor reliability [1, 14, 17] and suboptimal path quality, in terms of metrics such as latency, throughput, or loss rate [1, 27]. Sources can observe end-to-end failures and path quality and their effect on the particular application in use. If multiple paths are exposed, the end-hosts could react to these observations by switching paths much more quickly and in a more informed way than BGP's control plane, which takes minutes or tens of minutes to converge [19, 21]. For network providers, multipath routing represents a new service that can be sold. In fact, route control products exist today which dynamically select paths based on availability, performance, and cost for multi-homed edge networks [3]; exposing more flexibility in route selection could improve their effectiveness. Greater choice in routes may bring other benefits as well, such as enabling competition and encouraging "tussles" between different parties to be resolved within the protocol [6].

But providing multiple paths while respecting network owners' policies is nontrivial. BGP provides no multipath service; it selects a single path for each destination, which it installs in its FIB and advertises to its neighbors. Several multipath routing protocols have been proposed, but these have tradeoffs such as not supporting all of BGP's routing policies [32, 30], exposing only a limited number of additional paths [28], making it difficult to know which paths will be used [31, 23], or increase the size of the FIB [28, 31, 23], which would exacerbate the scalability challenge.

Our contributions. This paper addresses the challenges of scalability and multipath routing with a novel protocol called *pathlet routing*. In pathlet routing, each network advertises *pathlets*—fragments of paths represented as sequences of virtual nodes (vnodes) along which the network

*The first and fourth authors were supported in part by a Cisco Collaborative Research Initiative grant.

SIGCOMM'09, August 17–21, 2009, Barcelona, Spain.

NIRA: A New Inter-Domain Routing Architecture

Xiaowei Yang, *Member, IEEE*, David Clark, *Fellow, IEEE*, and Arthur W. Berger

***Abstract*—In today's Internet, users can choose their local Internet service providers (ISPs), but once their packets have entered the network, they have little control over the overall routes their packets take. Giving a user the ability to choose between provider-level routes has the potential of fostering ISP competition to offer enhanced service and improving end-to-end performance and reliability. This paper presents the design and evaluation of a new Internet routing architecture (NIRA) that gives a user the ability to choose the sequence of providers his packets take. NIRA addresses a broad range of issues, including practical provider compensation, scalable route discovery, efficient route representation, fast route fail-over, and security. NIRA supports user choice without running a global link-state routing protocol. It breaks an end-to-end route into a sender part and a receiver part and uses address assignment to represent each part. A user can specify a route with only a source and a destination address, and switch routes by switching addresses. We evaluate NIRA using a combination of network measurement, simulation, and analysis. Our evaluation shows that NIRA supports user choice with low overhead.**

***Index Terms*—Inter-domain routing, Internet architecture, routing, source routing, user-controlled routing.**

I. Introduction

THIS paper is concerned with the question of how Internet traffic is routed at the domain level [at the level of the autonomous system (AS)][1] as it travels from source to destination. Today, users can pick their own Internet service providers (ISPs), but once their packets have entered the network, the users have no control over the overall routes their packets take. ISPs make business decisions to interconnect, and technically the BGP routing protocol [48] is used to select the specific route a packet follows. Each domain makes local decisions that determine what the next hop (at the domain level) will be, but the user cannot exercise any control at this level. In this context, a *user* could be a human user, or a program.

In [12], Clark *et al.* argued that a better alternative would be to give the user more control over routing at the domain level. User-controlled routes are a key factor in maintaining the competitiveness of the ISP marketplace [12]. An analogy can be seen in the telephone system, which allows the user to pick his long-distance provider separately from his (usually monopolist) local provider. Allowing the user to select his long-distance provider has created the market for competitive long-distance service, and driven the price to a small fraction of its pre-competition starting point. For the consumer, especially the residential broadband consumer, there is likely to be a very small number of competitive local ISPs offering service [8], [12], [15].[2] With cable competing only with DSL, the market is a duopoly at best (at the facilities level) and often a monopoly in practice. If users cannot choose their backbone ISPs separately from their local ISPs, local providers can then control the selection of backbone providers and capture the market power of consumers. This will reduce the competitive pressures on the backbone providers and lead to a vertically integrated ISP market. The recent merger of SBC and AT&T (to use the old names) [5], and Verizon and MCI [63] only add to this concern. In the worst case, SBC sends all its traffic to the AT&T backbone, and Verizon sends its traffic to MCI. We have the re-emergence of market power in the backbone market. Conversely, when users can control the sequence of providers his packets take, the power of user choice fosters competition. In a competitive market, ISPs that are more efficient attract more users, which creates a positive loop for them to further advance technologies and to improve efficiency. In the long term, competition disciplines the market, drives innovation, and lowers the costs to provide services [12], [66].

Moreover, recent studies on overlay networks and multihoming show that letting the user choose routes also brings technical benefits. The default routing path chosen by BGP [48] may not be the best in terms of performance, reliability, or cost. End hosts on an overlay network often find alternative Internet paths with lower latencies, lower losses, or higher reliability than the default routes chosen by BGP [3], [27], [51]. For instance, Detour found that for almost 80% of the default paths, an alternative route offers a lower loss rate [51]. Similarly, recent studies also show that multihomed sites can improve path quality and reduce monetary cost by intelligently choosing their upstream providers [2], [25].

The prevalence of these alternative paths suggests that giving the user the ability to choose routes can lead to improved performance, reliability, or user satisfaction. Only users know whether a path works for their applications or not. A user may choose a path that has a high throughput and a low latency for playing online games, even if the path may cost more. In contrast, a user may prefer a low cost path for peer-to-peer file downloads. Furthermore, letting the user choose routes also improves reliability. A user can use multipath routing to improve the reliability for mission-critical applications, such as 911 calls, or quickly switch to an alternative route if the default routing path fails.

Manuscript received October 24, 2005; revised April 4, 2006; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor O. Bonaventure.
X. Yang is with the University of California at Irvine, Irvine, CA 92697 USA (e-mail: xwy@uci.edu).
D. Clark is with MIT CSAIL, Cambridge, MA 02139 USA.
A. W. Berger is with MIT CSAIL and Akamai Technologies, Cambridge, MA 02139 USA.
Digital Object Identifier 10.1109/TNET.2007.893888

[1]In this paper, an AS is also referred to as a *domain*. An AS that provides transit service is sometimes called a *provider*.

[2]Although the Telecommunications Act of 1996 requires that incumbent local exchange carriers (iLEC, the local phone companies that own the wires) to provide open access of their networks to competitors at reasonable costs, the implementation of the Act has been difficult [15]. The iLECs have disincentives to open up the local market. To block market entries, they may create obstacles such as significant nonrecurring costs in the leasing of their network elements. As a result, the DSL market shares of the competitive local exchange carriers keep decreasing [8]. In the mean time, facility-level competition has not taken place widely, due to the high capital requirements for market entry [8], [15].

Lots of proposals

Accountable Internet Protocol (AIP)

David G. Andersen[1], Hari Balakrishnan[2], Nick Feamster[3],

TVA: a DoS-limiting Network Architecture

SCION: Scalability, Control, and Isolation On Next-Generation Networks

Pathlet Routing

P. Brighten Godfrey[†], Igor Ganichev[‡], Scott Shenker[‡§], and Ion Sto
[†]University of Illinois at Urbana-Champaign [‡]UC Berkeley [§]ICS

NIRA: A New Inter-Domain Routing Architecture

IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 15, NO. 4, AUGUST 2007 775

Xiaowei Yang, *Member, IEEE*, David Clark, *Fellow, IEEE*, and Arthur W. Berger

Abstract—**In today's Internet, users can choose their lo Internet service providers (ISPs), but once their packets ha r the overall rout to choose betwe ng ISP competiti o-end performan gn and evaluati) that gives a us**

Serval: An End-Host Stack for Service-Centric Networking

Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer
Matvey Arye, Steven Y. Ko, Jennifer Rexford, Michael J. Freedman

MobilityFirst Future Internet Architecture Project

Ivan Seskar, Kiran Nagaraja, Sam Nelson and Dipankar Ra
WINLAB, Rutgers University

A Data-Oriented (and Beyond) Network Architecture

Teemu Koponen[*†] tkoponen@iki.fi
Mohit Chawla[‡] mchawla@cs.berkeley.edu
Byung-Gon Chun[‡] bgchun@cs.berkeley.edu
Andrey Ermolinskiy[‡] andreye@cs.berkeley.edu
Kye Hyun Kim[‡] kyekim@cs.berkeley.edu
Scott Shenker[*‡] shenker@icsi.berkeley.edu
Ion Stoica[‡] istoica@cs.berkeley.edu

ABSTRACT

The Internet has evolved greatly from its original incarnation. For instance, the vast majority of current Internet usage is data retrieval and service access, whereas the architecture was designed around host-to-host applications such as telnet and ftp. Moreover, the original Internet was a purely transparent carrier of packets, but now the various network stakeholders use middleboxes to improve security and accelerate applications. To adapt to these changes, we propose the Data-Oriented Network Architecture (DONA), which involves a clean-slate redesign of Internet naming and name resolution.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design

Keywords

Naming, Internet architecture, name resolution, data, middleboxes

1 Introduction

The DNS name resolution system is a fundamental part of today's Internet, underlying almost all Internet usage. However, the DNS was developed rather late in the Internet's evolution, after many basic pieces of the architecture were in place. For instance, TCP sessions were already bound to IP addresses and the Berkeley Socket API referred to addresses, not names; frozen design decisions, such as these, limited the extent to which DNS names (or any other naming system) could permeate the architecture. As a result, the current role of naming in the architecture is more an accident of history than the result of principled architectural design. In this proposal, we take a "clean-slate" look at naming and name resolution.

The test of any architecture is whether it gracefully accommodates a wide spectrum of potential uses (and can withstand potential abuses), both those we encounter in the present and those we anticipate for the future. However, to motivate our design, we first focus more narrowly on one particular issue; the shift in usage from host-centric to data-centric applications.

The first Internet applications, such as file transfer and remote login, focused strictly on host-to-host communication: The user explicitly directed the source to communicate with another host, and the network's only role was to carry packets to the destination address in the packet header. The Internet architecture was built around this host-to-host model and, as a result, the architecture is well-suited for communication between pairs of stationary hosts.

Today, however, the vast majority of Internet usage is data retrieval and service access, where the user cares about content and is oblivious to location. That is, the user knows that she wants headlines from CNN, or videos from YouTube, or access to her bank account, but does not know or care on which machine the desired data or service resides. The current architecture can support data/service access, as is obvious from its prevalence on the Internet, but it does not fit comfortably within the host-to-host model. For instance, consider the following three user-relevant issues:

- **Persistence:** once given a name for some data or service, the user would like that name to remain valid as long as the underlying data or service is available. There should be no equivalent of today's "broken links" when data is moved to another site. Today, HTTP redirect and dynamic DNS are used to minimize this problem, but they are not sufficient answers. For instance, neither works if the data changes administrative domains, unless the operator of the previous domain provides perpetual support.
- **Availability:** data and services should have high availability, in terms of both reliability and low-latency. Availability is usually provided by replication at endpoints, and the network's role is to allow user requests to find nearby copies. The first large-scale solution to this was deployed by Akamai, using intelligent DNS servers and URL rewriting.

[*]International Computer Science Institute (ICSI)
[†]Helsinki Institute for Information Technology (HIIT)
[‡]UC Berkeley, Computer Science Division

Networking Named Content

Van Jacobson Diana K. Smetters James D. Thornton Michael F. Plass
Nicholas H. Briggs Rebecca L. Braynard
Palo Alto Research Center
Palo Alto, CA, USA
{van,smetters,jthornton,plass,briggs,rbraynar}@parc.com

ABSTRACT

Network use has evolved to be dominated by content distribution and retrieval, while networking technology still speaks only of connections between hosts. Accessing content and services requires mapping from the *what* that users care about to the network's *where*. We present *Content-Centric Networking* (CCN) which treats content as a primitive – decoupling location from identity, security and access, and retrieving content by name. Using new approaches to routing named content, derived heavily from IP, we can simultaneously achieve scalability, security and performance. We implemented our architecture's basic features and demonstrate resilience and performance with secure file downloads and VoIP calls.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Network Architecture and Design; C.2.2 [**Computer Systems Organization**]: Network Protocols

General Terms

Design, Experimentation, Performance, Security

1. INTRODUCTION

The engineering principles and architecture of today's Internet were created in the 1960s and '70s. The problem networking aimed to solve was *resource sharing* — remotely using scarce and expensive devices like card readers or high-speed tape drives or even supercomputers. The communication model that resulted is a conversation between exactly two machines, one wishing to use the resource and one providing access to it. Thus IP packets contain two identifiers (addresses), one for the source and one for the destination host, and almost all the traffic on the Internet consists of (TCP) conversations between pairs of hosts.

In the 50 years since the creation of packet networking, computers and their attachments have become cheap, ubiquitous commodities. The connectivity offered by the Internet and low storage costs enable access to a staggering amount of new content – 500 exabytes created in 2008 alone [13]. People value the Internet for w tent it contains, but communication is still in terms of *whe*

We see a number of issues that affect users arising from compatibility between models.

- **Availability**: Fast, reliable content access requires a pre-planned, application-specific mechanisms like C P2P networks, and/or imposes excessive bandwidth
- **Security**: Trust in content is easily misplaced, re untrustworthy location and connection information.
- **Location-dependence**: Mapping content to host l complicates configuration as well as implementatio work services.

The direct, unified way to solve these problems is to *where* with *what*. Host-to-host conversations are a networ *straction* chosen to fit the problems of the '60s. We a *named data* is a better abstraction for today's communicati lems than *named hosts*. We introduce *Content-Centric Net* (CCN), a communications architecture built on named da has no notion of host at its lowest level – a packet "address content, not location. However, we preserve the design d that make TCP/IP simple, robust and scalable.

Figure 1 compares the IP and CCN protocol stacks. Mo of the stack reflect bilateral agreements; *e.g.*, a layer 2 fram tocol is an agreement between the two ends of a physical li layer 4 transport protocol is an agreement between some p and consumer. The only layer that requires universal agre layer 3, the network layer. Much of IP's success is due to plicity of its network layer (the IP packet - the thin 'wais stack) and the weak demands it makes on layer 2, namely: unreliable, unordered, best-effort delivery. CCN's netwo (Section 3) is similar to IP's and makes fewer demands on giving it many of the same attractive properties. Additional can be layered over anything, including IP itself.

CCN departs from IP in a number of critical ways. these, *strategy* and *security*, are shown as new layers in tocol stack. CCN can take maximum advantage of mul multaneous connectivities (*e.g.*, ethernet and 3G and bluet 802.11) due to its simpler relationship with layer 2. The layer (Section 3.3) makes the fine-grained, dynamic optin choices needed to best exploit multiple connectivities unde ing conditions. CCN secures content itself (Section 5), rat the connections over which it travels, thereby avoiding mar host-based vulnerabilities that plague IP networking.

We describe the architecture and operation of CCN in Se through 5. In Section 6 we evaluate performance using

Named Data Networking

Lixia Zhang
Alexander Afanasyev
Jeffrey Burke
Van Jacobson
UCLA

kc claffy
CAIDA
UC, San Diego

Patrick Crowley
Washington University,
St. Louis

Christos Papadopoulos
Colorado State Univ.

Lan Wang
Univ. of Memphis

Beichuan Zhang
Univ. of Arizona

ABSTRACT

Named Data Networking (NDN) is one of five projects funded by the U.S. National Science Foundation under its Future Internet Architecture Program. NDN has its roots in an earlier project, Content-Centric Networking (CCN), which Van Jacobson first publicly presented in 2006.[1] The NDN project investigates Jacobson's proposed evolution from today's host-centric network architecture (IP) to a *data-centric* network architecture (NDN). This conceptually simple shift has far-reaching implications for how we design, develop, deploy, and use networks and applications. We describe the motivation and vision of this new architecture, and its basic components and operations. We also provide a snapshot of its current design, development status, and research challenges. More information about the project, including prototype implementations, publications, and annual reports, is available on named-data.net.

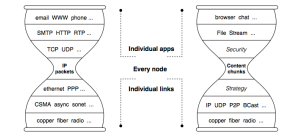


Figure 1: The main building blocks of the NDN architecture are named content chunks, in contrast to the IP architecture's fundamental unit of communication, which is an end-to-end channel between two end endpoints identified by IP addresses.

1. VISION: A NEW NARROW WAIST

Today's Internet's *hourglass* architecture centers on a *universal* network layer (i.e., IP) which implements the minimal functionality necessary for global interconnectivity. This thin waist enabled the Internet's explosive growth by allowing both lower and upper layer technologies to innovate independently. However, IP was designed to create a *communication network*, where packets named only communication endpoints. Sustained growth in e-commerce, digital media, social networking, and smartphone applications has led to dominant use of the Internet as a *distribution network*. Distribution networks are more general than communication networks, and solving distribution problems via a point-to-point communication protocol is complex and error-prone.

The Named Data Networking (NDN) project proposed an evolution of the IP architecture that generalizes the role of this thin waist, such that packets can name objects other than communication endpoints (Figure 1). More specifically, NDN changes the semantics of network service from *delivering the packet to a given destination address* to *fetching data identified by a given name*. The name in an NDN packet can name anything – an endpoint, a data chunk in a movie or a book, a command to turn on some lights, *etc.* This conceptually simple change allows NDN networks to use almost all of the Internet's well-tested engineering properties to solve a much broader range of problems including not only end-to-end communications but also content distribution and control problems. Based on three decades of experience with the strengths and limitations of the current Internet architecture, the design also builds in security primitives (via signatures on all named data) and self-regulation of network traffic (via flow balance between Interest and Data packets). The architecture includes functionality designed to be conducive to user choice and competition as the network evolves, such as multipath forwarding and in-network storage.

NDN is one instance of a more general network research direction called *information-centric networking* (ICN), under which different architecture designs have emerged [29]. The Internet Research Task Force (IRTF) established an ICN research working group in 2012[2]. In this paper we provide a brief (and necessarily incomplete) snapshot of the current state of the NDN architecture research project, which includes sixteen NSF-funded principal investigators at twelve campuses, and growing interest from the academic and industrial research communities. A more complete description of recent activities is in the third annual project report [20]

How do we enable innovation in Internet Architecture?

Trotsky: Enabling a Permanent Revolution in Internet Architecture

James McCauley, Yotam Harchol, Aurojit Panda,
Barath Raghavan, Scott Shenker

SIGCOMM'19

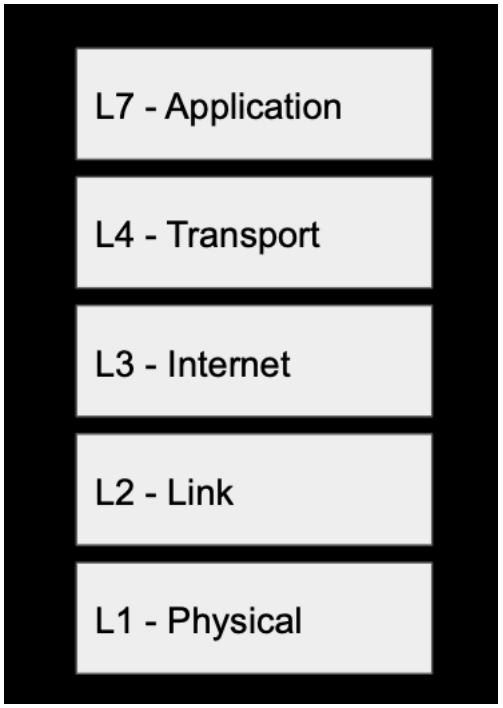
Some slide contents borrowed from McCauley's SIGCOMM'19 talk.

How do we enable innovation in Internet Architecture?

- Remove the narrow waist!
- How?
- Two steps

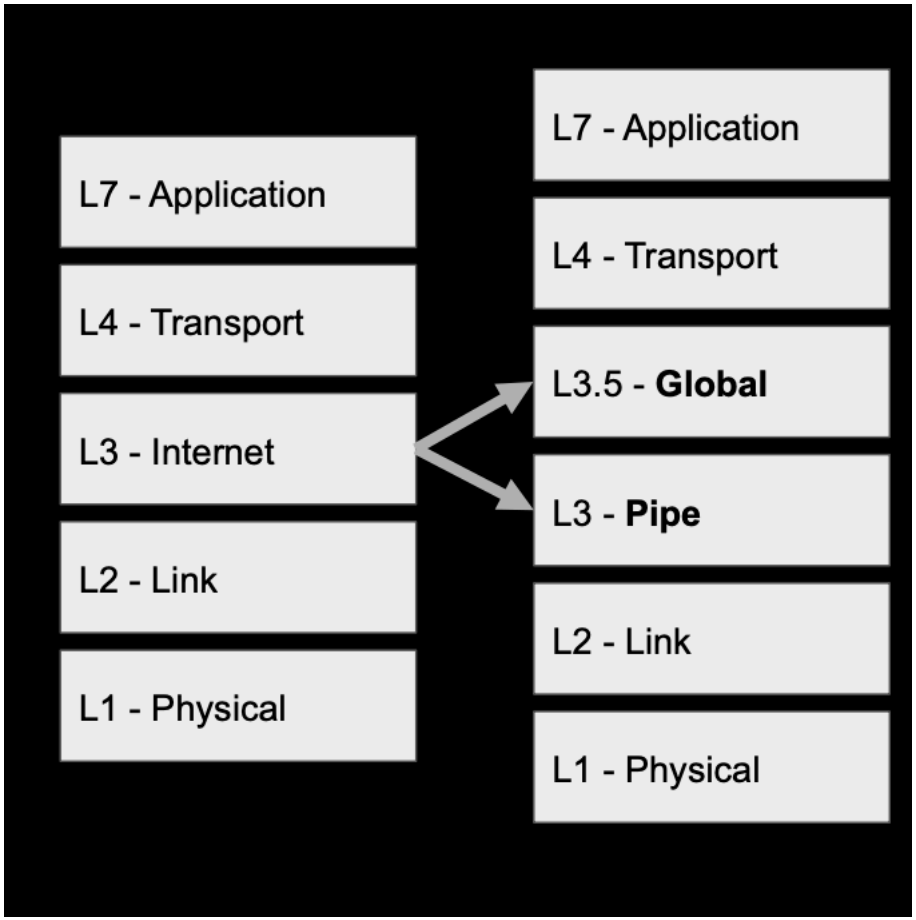
Step 1: Fix Layering

- We are missing a layer!
- Internet is not a composition of L2 networks.
- It is a composition of *domains*.



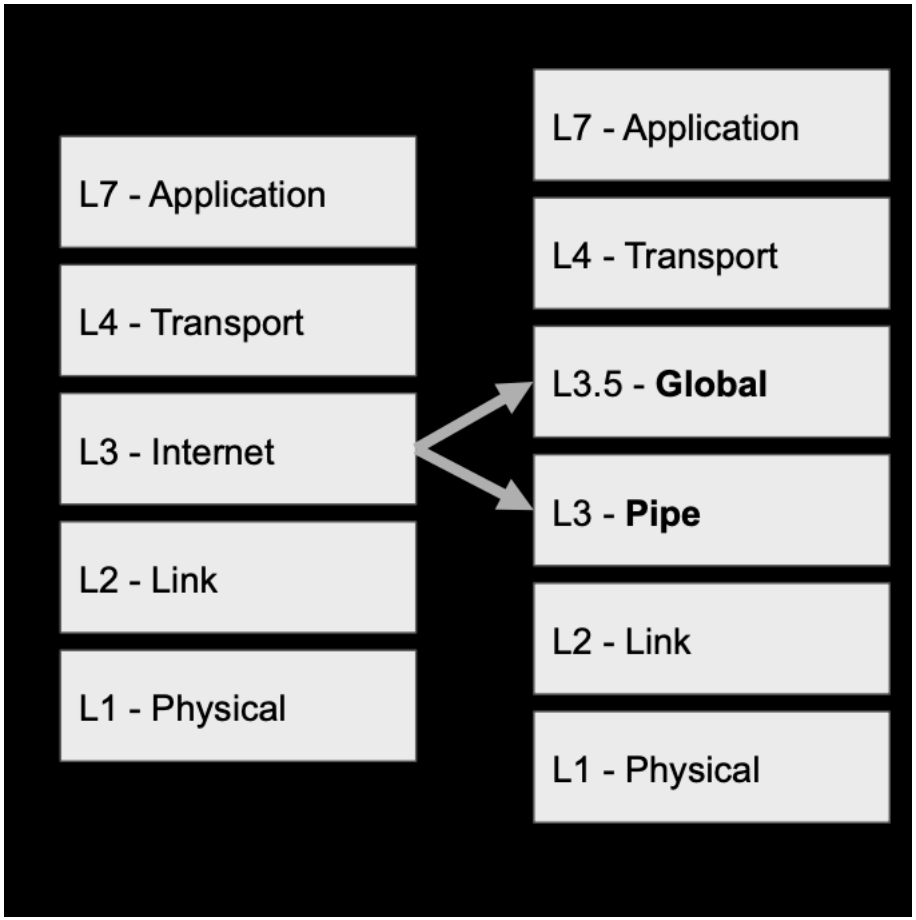
Step 1: Fix Layering

- We are missing a layer!
- Internet is not a composition of L2 networks.
- It is a composition of *domains*.



Step 1: Fix Layering

- Decouple how data is delivered:
  - within a domain (L3)
  - across domains (L3.5)
- Decouple how two domains deliver data internally.



How do we enable innovation in Internet Architecture?

- Remove the narrow waist!
- How?
- Two steps:
  - Layer 3.5: decouple intra-domain and inter-domain data delivery.

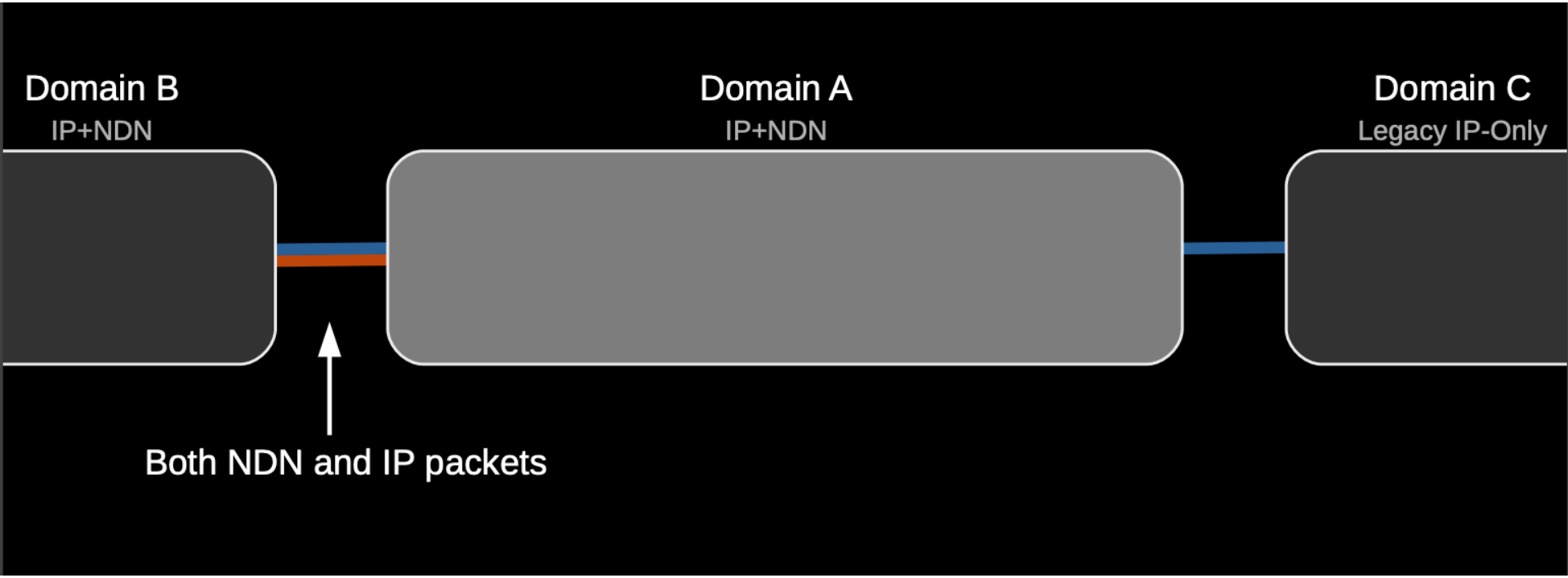
Step 2: Embrace multiple architectures

- Support multiple L3.5 protocols.
  - Up to the domain to choose which ones it wants to support.
- Trotsky Processors (TPs) deployed at domain edge (in software) responsible for implementing supported L3.5 protocols.

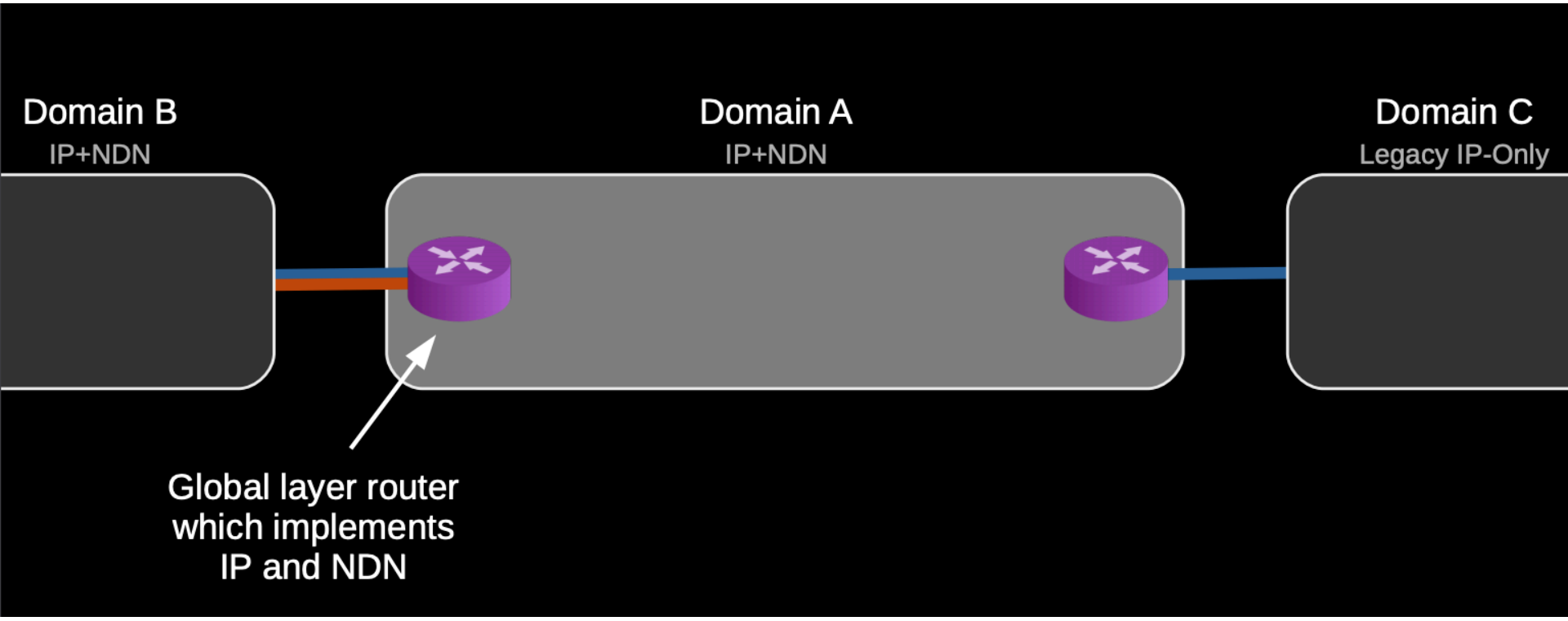
How do we enable innovation in Internet Architecture?

- Remove the narrow waist!
- How?
- Two steps:
  - Layer 3.5: decouple intra-domain and inter-domain data delivery.
  - Embrace multiple L3.5 protocols instead of upgrading to a single one.

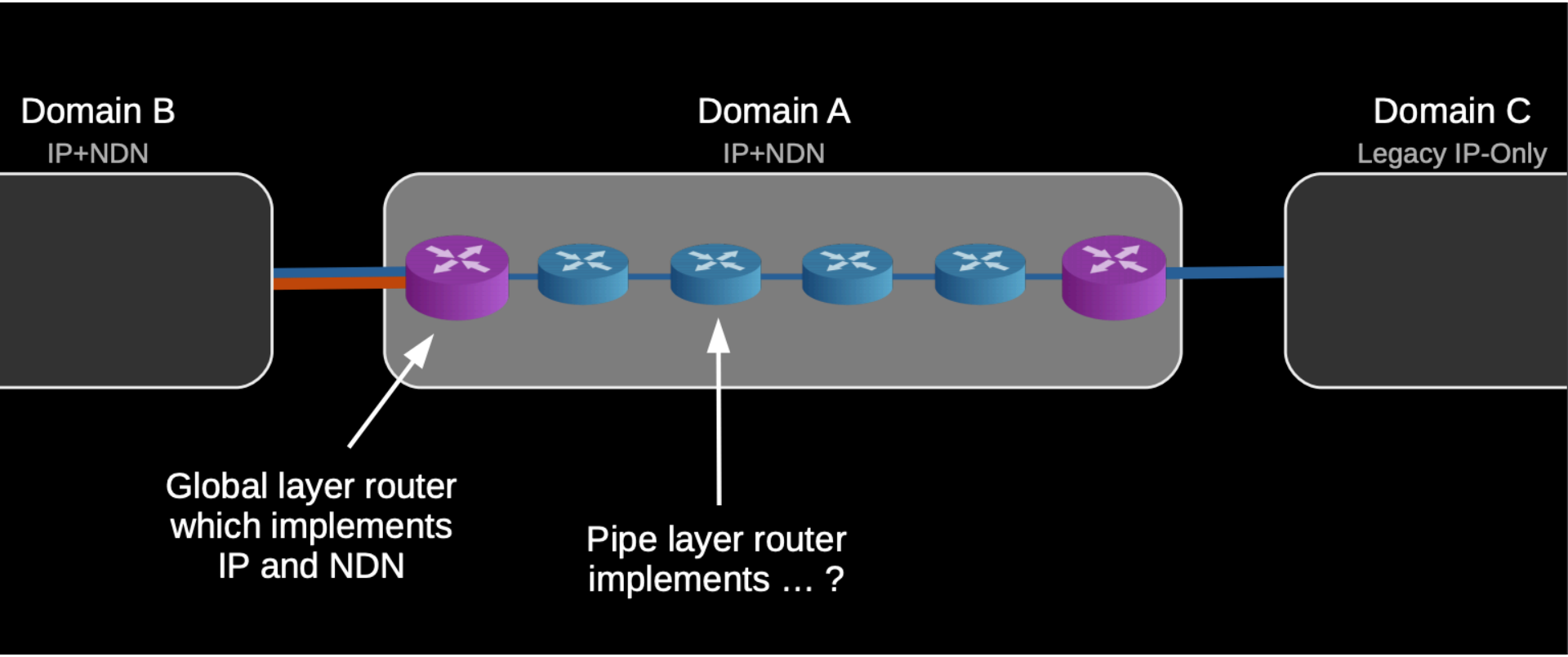
Inside a domain



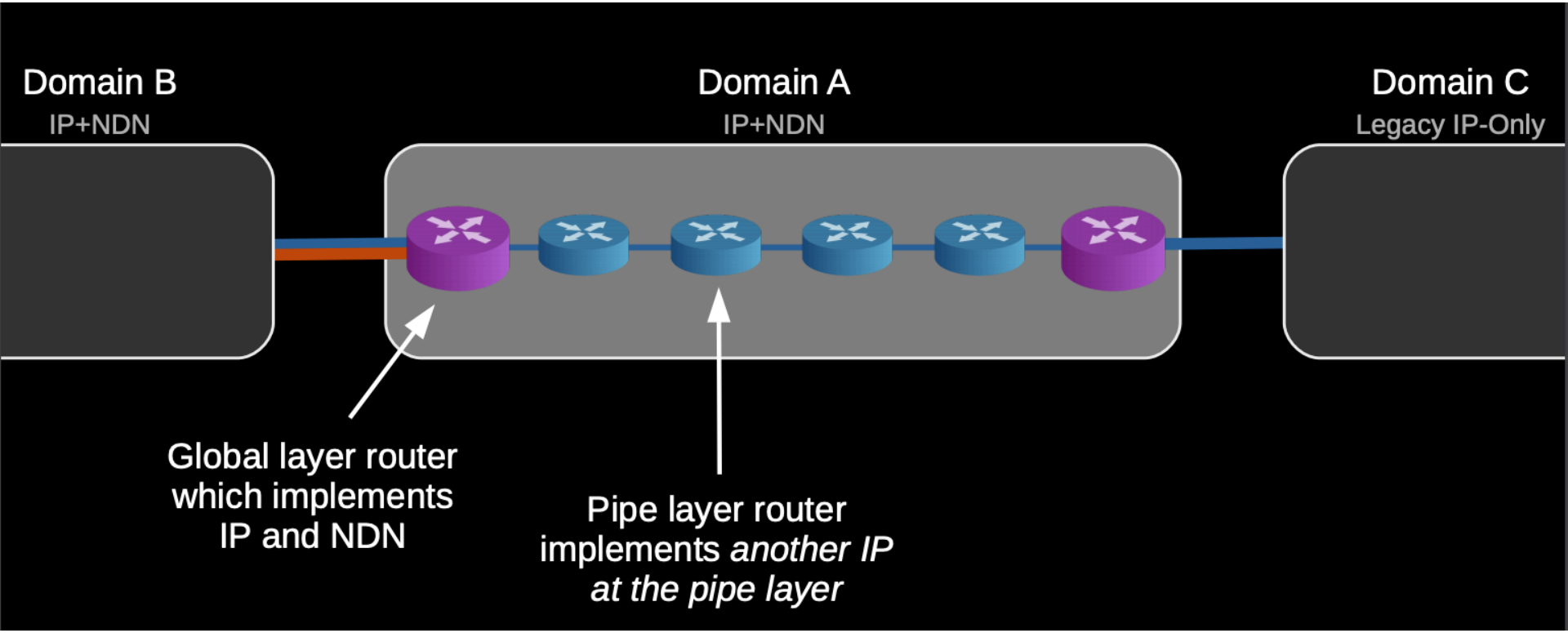
Inside a domain



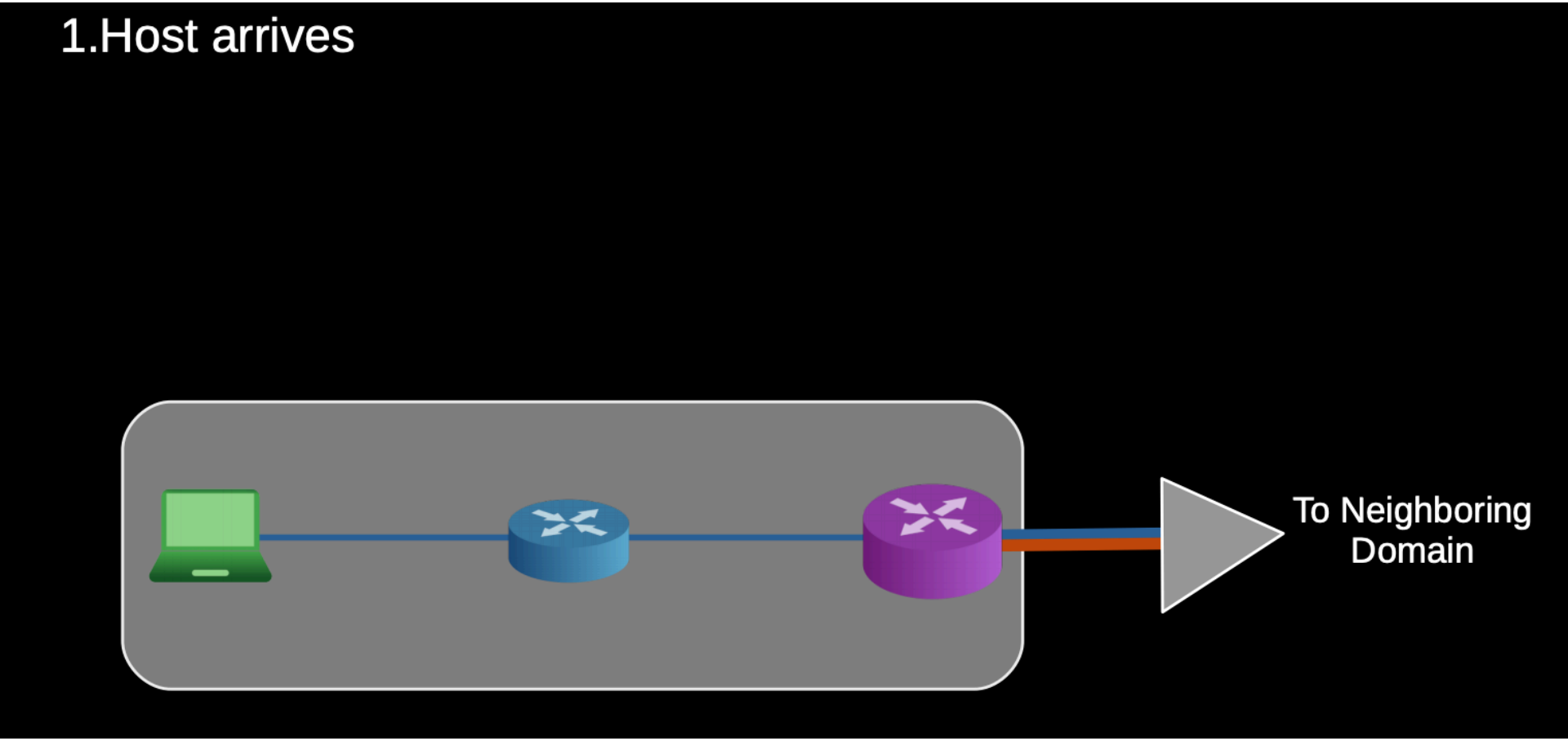
Inside a domain



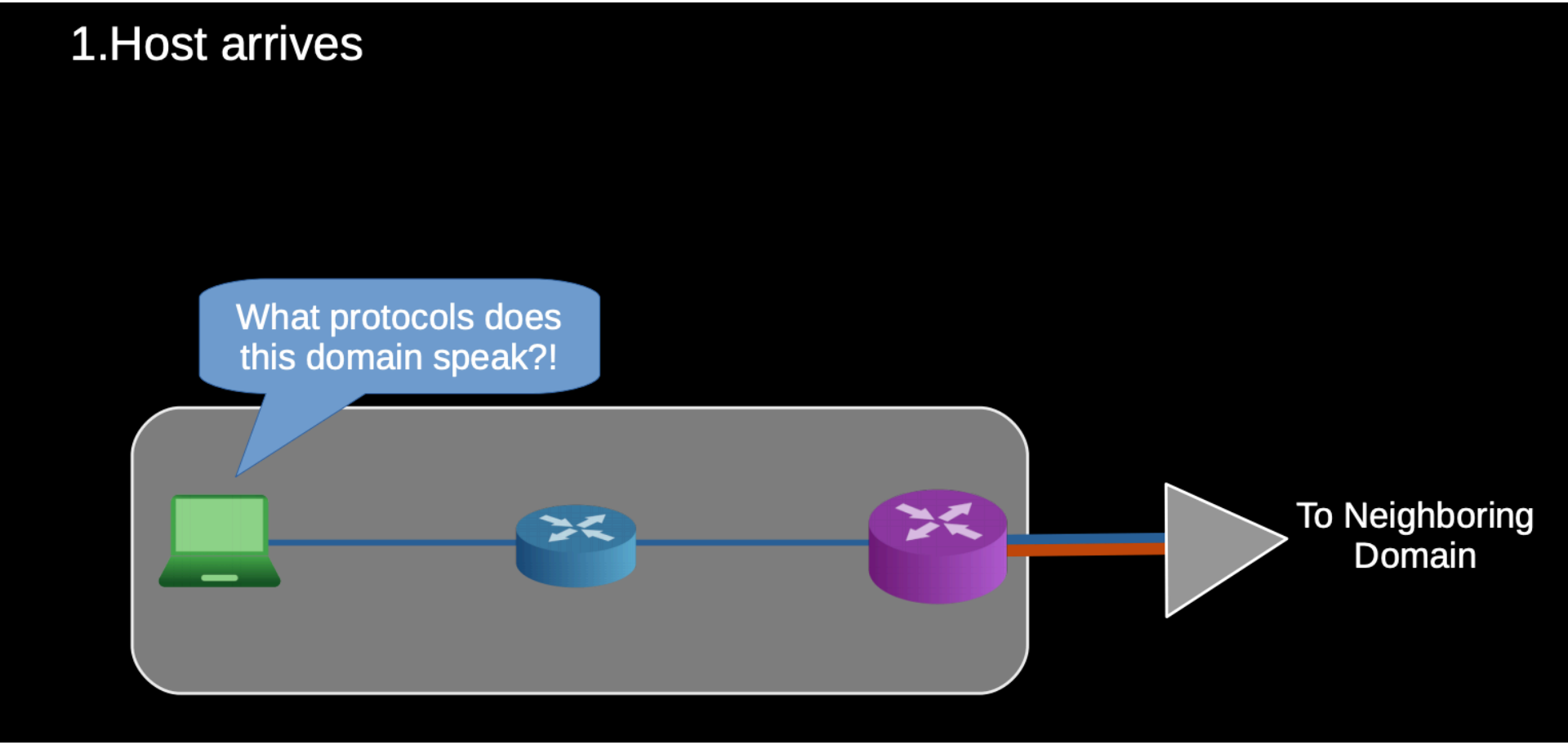
Inside a domain



Host initialization



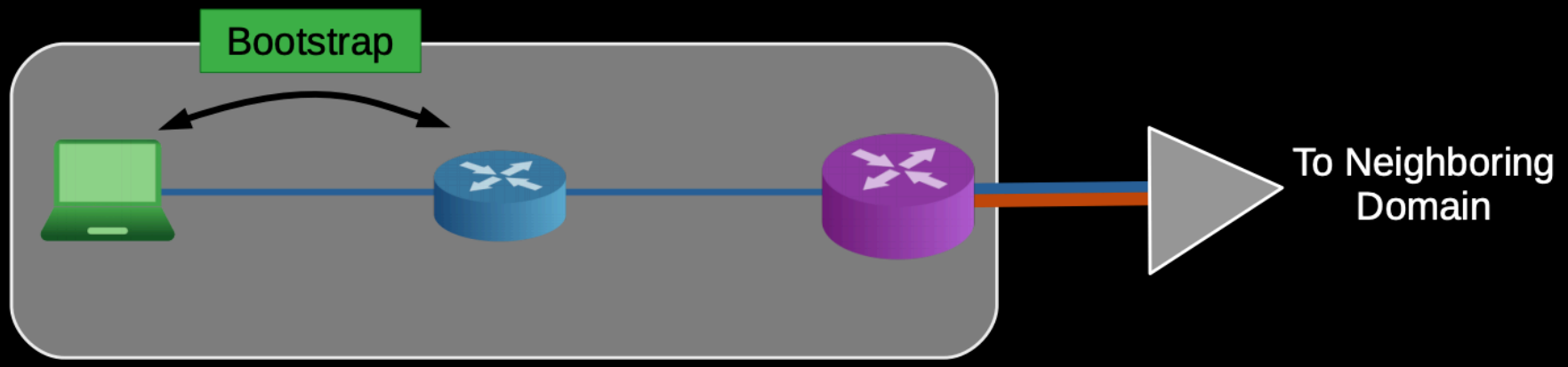
Host initialization



Host initialization

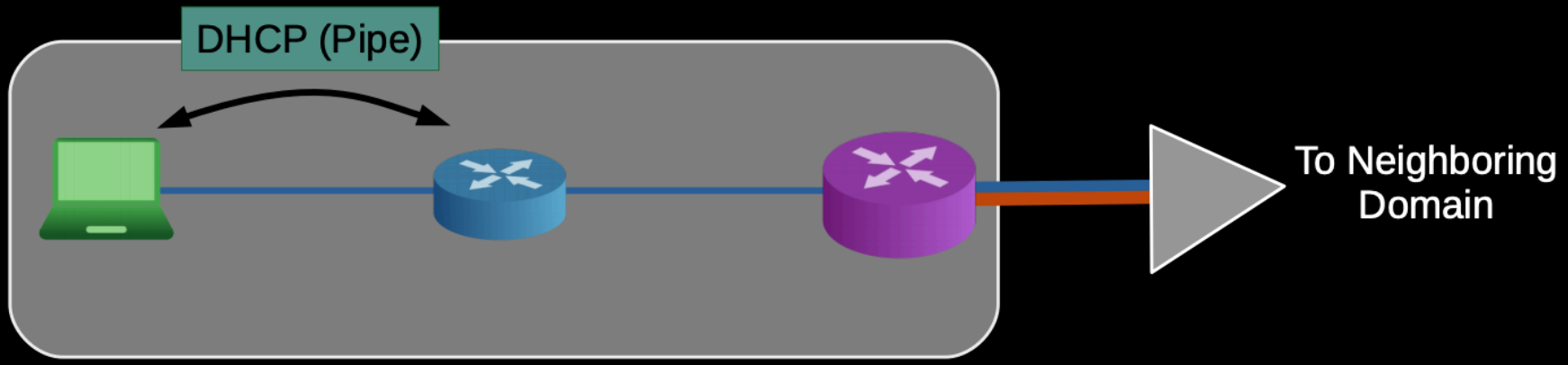
1.Host arrives

2.Bootstrap to learn which pipe and global protocols supported



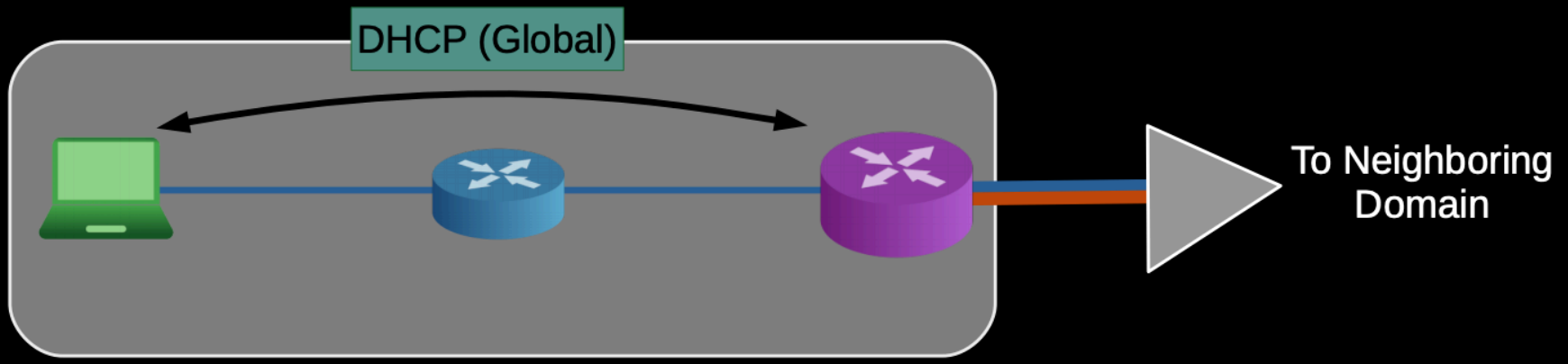
Host initialization

1. Host arrives
2. Bootstrap to learn which pipe and global protocols supported
3. Configure pipe layer (e.g. DHCP for pipe layer IP address)



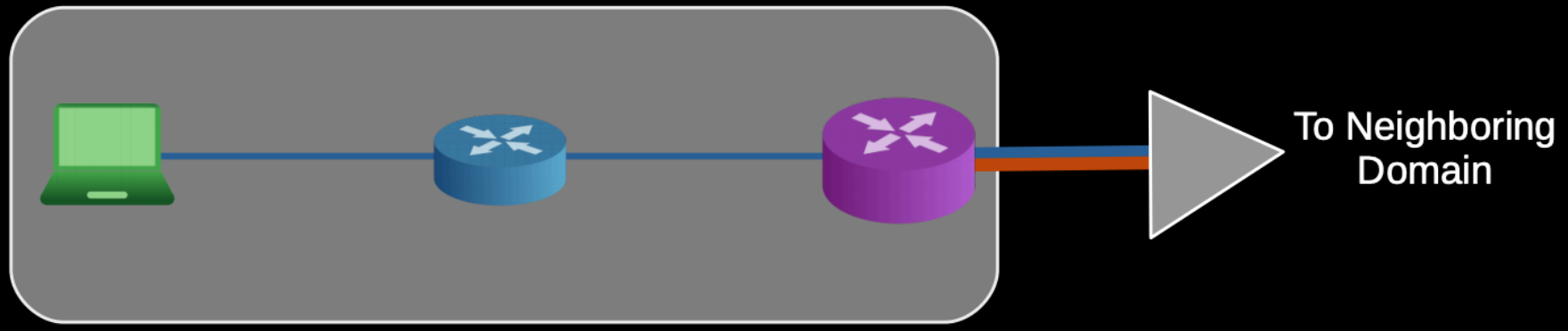
Host initialization

1. Host arrives
2. Bootstrap to learn which pipe and global protocols supported
3. Configure pipe layer (e.g. DHCP for pipe layer IP address)
4. Configure global layer (e.g., DHCP for global layer IP address)

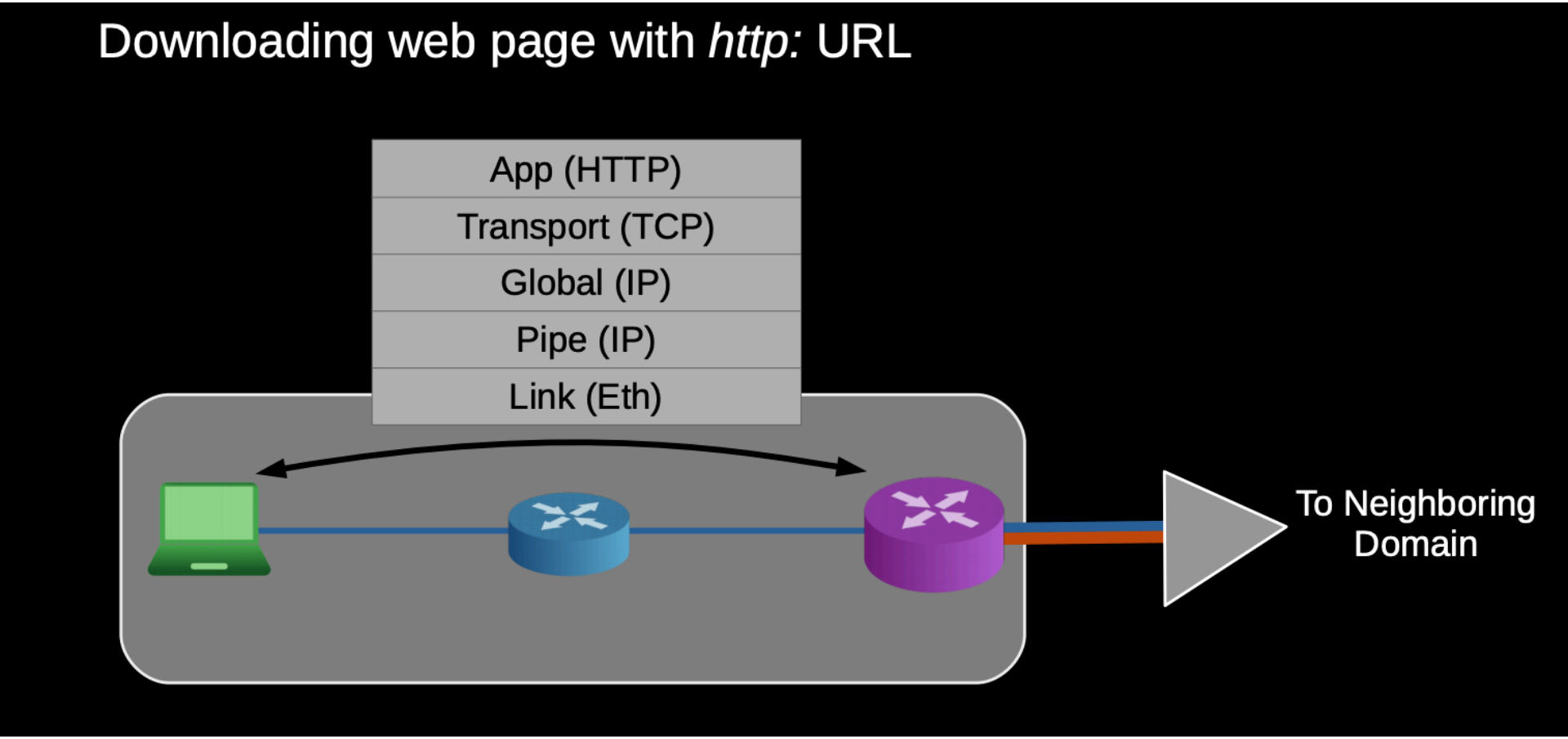


Host initialization

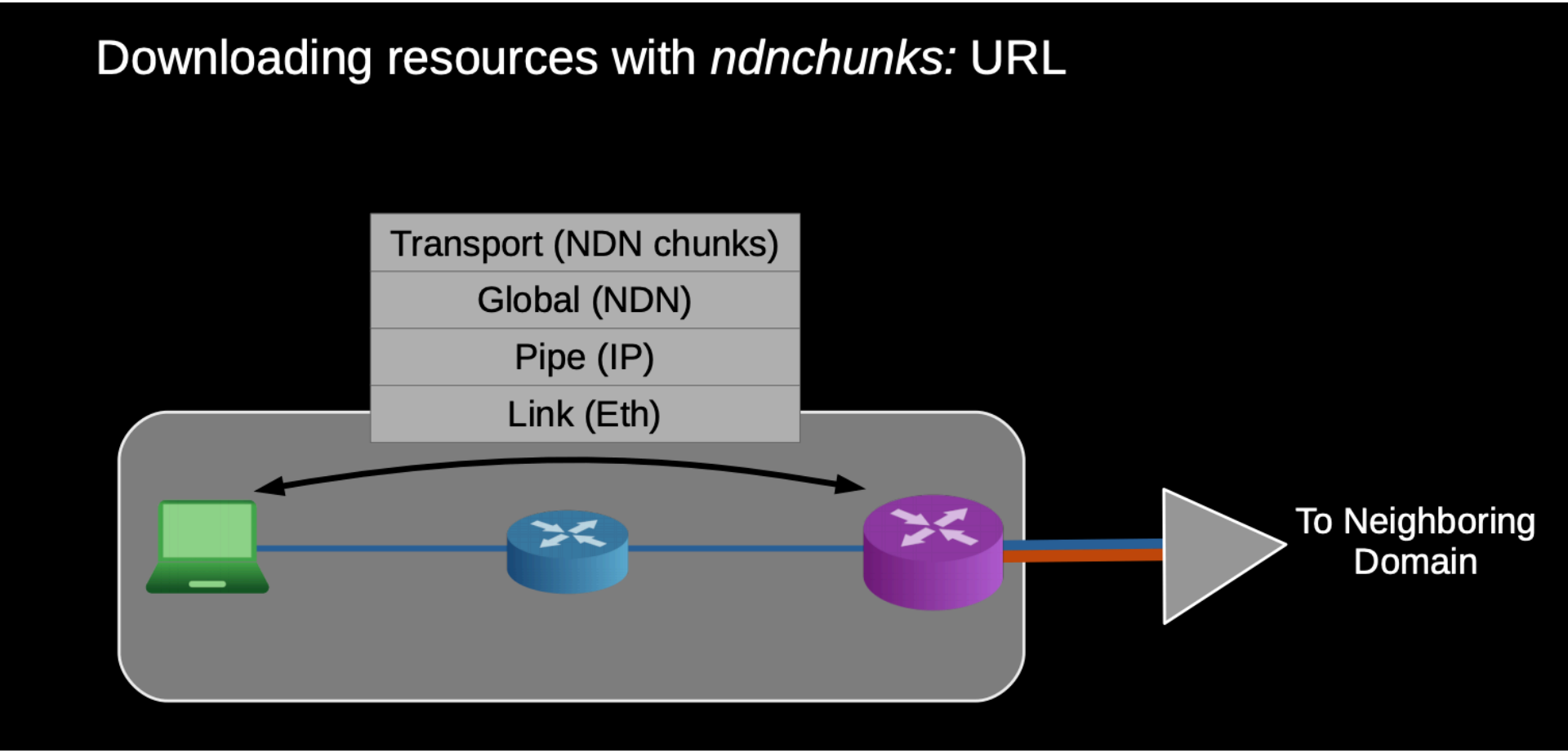
1. Host arrives
2. Bootstrap to learn which pipe and global protocols supported
3. Configure pipe layer (e.g. DHCP for pipe layer IP address)
4. Configure global layer (e.g., DHCP for global layer IP address)
5. Initialization complete



Web Download



Web Download



Partial Deployment of L3.5 Designs

Key Contribution of Trotsky

- A framework that allows incremental side-by-side deployment of new architectures.
- And is itself incrementally deployable.

Summary

- Goal: enable extensibility in Internet architecture.
- Problem: the universal narrow waist.
- Solution: remove it!
  - Decouple intra-domain and inter-domain data planes.
  - Embrace co-existence of multiple inter-domain protocols.
- Result:
  - An incrementally deployable design.
  - ..which can incrementally deploy new architectures.

Discussion

- Is the universal narrow waist truly removed?

- What are the limitations of Trotsky design?

Your opinions

- Pros
  - Backwards-compatible and incrementally deployable.
  - Framework providing only a minimal set of functionality.
  - No need to change all routers.
  - Opens up avenue for future research.

Your opinions

- Cons
  - Overhead of mapping L3.5 to/from underlying layers.
  - Overhead of implementing an L3.5 protocol (in software).
  - Pairwise translators needed at domain edge.
  - Requires some form of cooperation between ASes.
  - Can a network middlebox provide the same functionality at Trotsky?
  - How crucial is the decoupling between L3 and L3.5?
  - To what extent can it provide security?
  - Initial deployment is challenging.
  - "Simplicity is a feature not a bug" – do we really need more complex Internet architectures?

Your opinions

- Ideas
  - Is Trotsky against end-to-end argument?
  - Design DDoS resilient network architecture.
  - Implementation and evaluation of L3.5 protocols.
  - Why not implement Trotsky Processors in programmable switches?
  - What are the limitations of proposed L3 protocols?
  - Explore what incentivizes domains to support an L3.5 protocol.
  - Experiment testbed that allows multiple architecture to co-exist.