The eXpressive Internet Architecture: Architecture and Research Overview

Funded by the NSF under awards
CNS-1040801, CNS-1040757, CNS-1040800,
CNS-1345305, CNS-1345307, CNS-1345284, and CNS-1345249

1 XIA Architecture Overview

The goal of the XIA project is to radically improve both the evolvability and trustworthiness of the Internet. To meet this goal, the eXpressive Internet Architecture defines a single network that offers inherent support for communication between multiple communicating principals–including hosts, content, and services–while accommodating unknown future entities. Our design of XIA is based on a narrow waist, like today's Internet, but this narrow waist can evolve to accommodate new application usage models and to incorporate improved link, storage, and computing technologies as they emerge. XIA is further designed around the core guiding principle of intrinsic security, in which the integrity and authenticity of communication is guaranteed. We have also defined an initial inter-domain control plane architecture that support routing protocols for several principal types. Finally, the XIA core architecture includes the design of Scion, a path selection protocol that supports significant security benefits over traditional destination forwarding approaches.

1.1 Vision

The vision we outlined for the future Internet in the proposal is that of a single internetwork that, in contrast to today's Internet, will:

Be trustworthy: Security, broadly defined, is the most significant challenge facing the Internet today.

Support long-term evolution of usage models: The primary use of the original Internet was host-based communication. With the introduction of the Web, over the last nearly two decades, the communication has shifted to be dominated today by content retrieval. Future shifts in use may cause an entirely new dominant mode to emerge. The future Internet should not only support communication between today's popular entities (hosts and content), but it must be flexible, so it can be extended to support new entities as use of the Internet evolves.

Support long-term technology evolution: Advances in link technologies as well as storage and compute capabilities at both end-points and network devices have been dramatic. The network architecture must continue to allow easy and efficient integration of new link technologies and evolution in functionality on all end-point and network devices in response to technology improvements and economic realities.

Support explicit interfaces between network actors: The Internet encompasses a diverse set of actors playing different roles and also serving as stakeholders with different goals and incentives. The architecture must support well-defined interfaces that allow these actors to function effectively. This is true both for the interface between users (applications) and the network, and between the providers that will offer services via the future Internet.

This vision was the driver for both the original and the current XIA project.

1.2 Key Principles driving XIA

In order to realize the above vision, the XIA architecture follows three driving principles:

1. The architecture must support an evolvable set of first-order principals for communication, exposing the respective network elements that are intended to be bridged by the communication, be they hosts, services, content, users, or something as-yet unexpected.

Performing operations between the appropriate principal types creates opportunities to use communication techniques that are most appropriate, given the available technology, network scale, destination popularity, etc. Such "late binding" exposes intent and can dramatically reduce overhead and complexity compared with requiring all communication to operate at a lower level (e.g., projecting all desired communications onto host-to-host communications, as in today's Internet), or trying to "shoe-horn" all communication into a higher level (e.g., using HTTP as the new narrow waist of the Internet [96]).

A set of evolvable principal types is however not sufficient to support evolution of the network architecture. Experience so far shows that the concept of *fallbacks* is equally important. Allowing addresses to contain multiple identifiers supports only incremental deployment of principal types, but selective deployment by providers and network customization.

2. The security of "the network", broadly defined, should be as *intrinsic* as possible—that is, not dependent upon the correctness of external configurations, actions, or databases.

To realize this principle, we propose to extend the system of self-certification proposed in the Accountable Internet Protocol (AIP) [11]. In AIP, hosts are "named" by their public key. As a result, once a correspondent knows the name of the host they wish to contact, all further cryptographic security can be handled automatically, without external support.

We call the generalization of this concept "intrinsic security". Intuitively, this refers to a key integrity property that is built in to protocol interfaces: any malicious perturbation of an intrinsically secure protocol must yield a result that is clearly identifiable as ill-formed or incorrect, under standard cryptographic assumptions. We extend the application and management of self-certifying identifiers into a global framework for integrity such that both control plane messages and content on the data plane of the network can be efficiently and certifiably bound to a well-defined originating principal. Intrinsic security properties can also be used to bootstrap systematic mechanisms for trust management, leading to a more secure Internet in which trust relationships are exposed to users.

3. A pervasive narrow waist for all key functions, including access to principals (e.g., service, hosts, content), interaction among stakeholders (e.g., users, ISPs, content providers), and trust management.

The current Internet has benefited significantly from having a "narrow waist" that identifies the minimal functionality a device needs to be Internet-capable. It is limited to host-based communication, and we propose to widen its scope while retaining the elegance of "narrowness". The narrow waist effectively is an interface that governs the interactions between the stakeholders, so it defines what operations are supported, what controls are available, and what information can be accessed. It plays in key role in the tussle [27] of control between actors by creating explicit control points for negotiation. The architecture must incorporate a narrow waist for each principal type it supports. The narrow waist identifies the API for communication between the principal types, and for the control protocols that support the communication. The architecture must also define a narrow waist that enables a flexible set of mechanisms for trust management, allowing applications and protocols to bridge from a human-readable description to a machine-readable, intrinsically secure, identifier.

Our experience so far shows that precise, minimal interfaces are not only important as control points

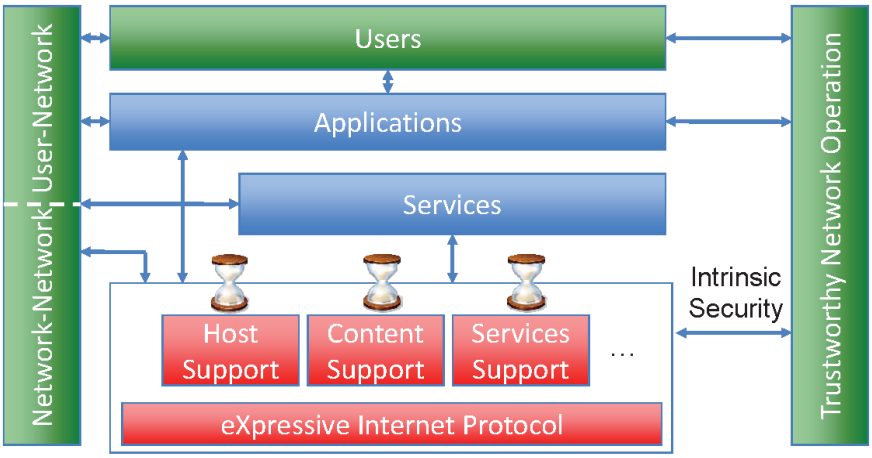


Figure 1: The eXpressive Internet Architecture

between actors, but also because they also play a key role in the evolvability of the network at large. The same was that a hard to evolve data plane (format and semantics of network header) can stiffle innovation, so can interfaces in the control plane (e.g., BGP for routing) or for resource management (e.g., AIMD based congestion control). Well-defined interfaces are critical to evolvability at all levels in the network.

1.3 XIA Data Plane Architecture

Building on the above principles, we introduce the "eXpressive Internet Architecture" (or XIA). The core of XIA is the eXpressive Internet Protocol (XIP), which supports communication between various types of principals. It is shown in red in the bottom center of Figure 1. XIP supports supports various communication and network services (including transport protocols, mobility services, etc.) which in in turn support applications on end-hosts. These components are shown in blue in the center of the figure. This protocol stack forms a "bare-bones" network Additional elements are needed to make the network secure, usable, and economically viable.

First, we need trustworthy network operations (shown on the right in green), i.e., a set of control and management protocols that deliver trustworhthy network service to users. We will describe SCION, a path selection protocol developed as part of XIA, as an example later in this section. Other examples include trust management, protocols for billing, monitoring, and diagnostics. Second, we need interfaces so various actors can productively participate in the Internet. For example, users need visibility into (and some level of control over) the operation of the network. We also need interfaces between network providers. The design of these network-network interfaces must consider both technical requirements, economic incentives, and the definition of policies that reward efficient and effective operation.

XIP was defined in the FIA-funded XIA project. We include an overview of its design in this section because it is the core of the architecture and forms the basis for the the research activities reported on this report. We also give an overview of SCION, a control protocol for path selection, in the next section for the same reason.

1.3.1 Principals and Intrinsic Security

The idea is that XIA supports communication between entities of different types, allowing communicating parties to express the intent of their communication operation. In order to support evolution in the network, e.g. to adapt to changing usage models, the set of principal types can evolve over time. At any given

point in time, support for certain principal types, e.g. autonomous domains, will be mandatary to ensure interoperability, while support for other principal types will be optional. Support for an optional principal type in a specific network will depend on business and other considerations.

IP is notoriously hard to secure, as network security was not a first-order consideration in its design. XIA aims to build security into the core architecture as much as possible, without impairing expressiveness. In particular, principals used in XIA source and destination addresses must be *intrinsically secure*. We define intrinsic security as *the capability to verify type-specific security properties without relying on external information*. XIDs are therefore typically cryptographically derived from the associated communicating entities in a principal type-specific fashion, allowing communicating entities to ascertain certain security and integrity properties of their communication operations [9].

The specification of a principal *type* must define:

1. The semantics of communicating with a principal of that type.

2. A unique XID type, a method for allocating XIDs and a definition of the intrinsic security properties of any communication involving the type. These intrinsically secure XIDs should be globally unique, even if, for scalability, they are reached using hierarchical means, and they should be generated in a distributed and collision-resistant way.

3. Any principal-specific per-hop processing and routing of packets that must either be coordinated or kept consistent in a distributed fashion.

These three features together define the *principal-specific support* for a new principal type. The following paragraphs describe the administrative domain, host, service, and content principals in terms of these features.

Network and *host* principals (NIDs and HIDs) represent autonomous routing domains and hosts that attach to the network. NIDs provide hierarchy or scoping for other principals, that is, they primarily provide control over routing. Hosts have a single identifier that is constant regardless of the interface used or network that a host is attached to. NIDs and HIDs are self-certifying: they are generated by hashing the public key of an autonomous domain or a host, unforgeably binding the key to the address. The format of NIDs and HIDs and their intrinsic security properties are similar to those of the network and host identifiers used in AIP [11].

Services represent an application service running on one or more hosts within the network. Examples range from an SSH daemon running on a host, to a Web server, to Akamai's global content distribution service, to Google's search service. Each service will use its own application protocol, such as HTTP, for its interactions. An SID is the hash of the public key of a service. To interact with a service, an application transmits packets with the SID of the service as the destination address. Any entity communicating with an SID can verify that the service has the private key associated with the SID. This allows the communicating entity to verify the destination and bootstrap further encryption or authentication.

In today's Internet, the true endpoints of communication are typically application processes—other than, e.g., ICMP messages, very few packets are sent to an IP destination without specifying application port numbers at a higher layer. In XIA, this notion of processes as the true destination can be made explicit by specifying an SID associated with the application process (e.g., a socket) as the intent. An NID-HID pair can be used as the "legacy path" to ensure global reachability, in which case the NID forwards the packet to the host, and the host "forwards" it to the appropriate process (SID). In [51], we show and example of how making the true process-level destination explicit facilitates transparent process migration, which is difficult in today's IP networks because the true destination is hidden as state in the receiving end-host.

Lastly, *the content principal* allows applications to express their intent to retrieve content without regard to its location. Sending a request packet to a CID initiates retrieval of the content from either a host, an

in-network content cache, or other future source. CIDs are the cryptographic hash (e.g., SHA-1, RIPEMD-160) of the associated content. The self-certifying nature of this identifier allows any network element to verify that the content retrieved matches its content identifier.

1.3.2 Addressing Requirements

Three key innovative features of the XIA architecture are support for multiple principal types that allow users to express their intent, support for evolution, and the use of intrinsically secure identifiers. These features all depend heavily on the format of addresses that are present in XIA packets, making the XIA addressing format one of the key architectural decisions we have to make. While this may seem like a narrow decision, the choice of packet addressing and the associated packet forwarding mechanisms have significant implications on how the network can evolve, the flexibility given to end hosts, the control mechanisms provided to infrastructure providers, and the scalability of routing/forwarding. We elaborate on the implications of each of these requirements before presenting our design.

The *scalability* requirements for XIA are daunting. The number of possible end-points in a network may be enormous - the network contains millions of end-hosts and services, and billions of content chunks. Since XIA end-points are specified using flat identifiers, this raises the issue of forwarding table size that routers must maintain. To make forwarding table size more scalable, we want to provide support for scoping by allowing packets to contain a destination network identifier, in addition to the actual destination identifier.

Evolution of the network requires support for incremental deployment, i.e. a particular principal type and its associated destination identifiers may only be recognized by some parts of the network. Packets that use such a partially supported principal type may encounter a router that lacks support for such identifiers. Simply dropping the packet will hurt application performance (e.g. timeouts) or break connectivity, which means that applications will not use principal types that are not widely supported. This, in turn, makes it hard to justify adding support in routers for unpopular types. To break out of this cycle, the architecture must provide some way to handle these packets correctly and reasonably efficiently, even when a router does not support the destination type. XIA allows packets to carry a *fallback* destination, i.e. an alternative address, a set of addresses or a path, that routers can use when the intended destination type is not supported.

In the Internet architecture, packets are normally processed independently by each router based solely on the destination address. "Upstream" routers and the source have little control over the path or processing that is done as the packet traverses the network. However, various past architectures have given additional control to these nodes, changing the balance between the *control by the end-point and control by the infrastructure*. For example, IP includes support for source routing and other systems have proposed techniques such as delegation. The flexibility provided by these mechanisms can be beneficial in some context, but they raise the challenge of how to ensure that upstream nodes do not abuse these controls to force downstream nodes to consume resources or violate their normal policies. In XIA, we allow packets to carry more explicit path information than just a destination; thus, giving the source and upstream routers the control needed to achieve the above tasks.

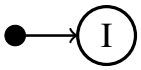
The above lays out the requirements for addressing in packets, and indirectly specifies requirements for the forwarding semantics. We now present our design of an effective and compact addressing format that addresses the evolution and control requirements. Next, we discuss the forwarding semantics and router processing steps for XIA packets.

1.3.3 XIA Addressing

XIA's addressing scheme is a direct realization of the above requirements. To implement fallback and scoping, XIA uses a restricted directed acyclic graph (DAG) representation of XIDs to specify XIP addresses [51]. A packet contains both the destination DAG and the source DAG to which a reply can be sent.

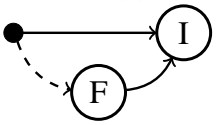
Because of symmetry, we describe only the destination address.

Three basic building blocks are: intent, fallback, and scoping. XIP addresses must have a *single* intent, which can be of any XID type. The simplest XIP address has only a "dummy" source and the intent (I) as a sink:



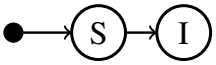
The dummy source (•) appears in all visualizations of XIP addresses to represent the conceptual source of the packet.

A fallback is represented using an additional XID (F) and a "fallback" edge (dotted line):



The fallback edge can be taken if a direct route to the intent is unavailable. While each node can have multiple outgoing fallback edges, we allow up to four fallbacks to balance between flexibility and efficiency.

Scoping of intent is represented as:

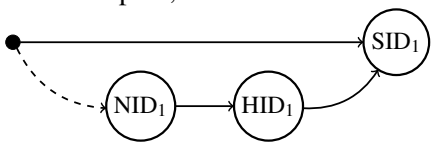


This structure means that the packet must be first routed to a scoping XID S, even if the intent is directly routable.

These building blocks are combined to form more generic DAG addresses that deliver rich semantics, implementing the high-level requirements in Section 1.3.2. To forward a packet, routers traverse edges in the address in order and forward using the next routable XID. Detailed behavior of packet processing is specified in Section 1.3.4.

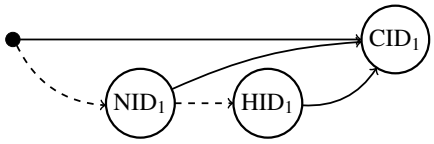
To illustrate how DAGs provide flexibility, we present three (non-exhaustive) "styles" of how it might be used to achieve important architectural goals.

Supporting evolution: The destination address encodes a service XID as the intent, and an autonomous domain and a host are provided as a fallback path, in case routers do not understand the new principal type.



This scheme provides both fallback and scalable routing. A router outside of NID_1 that does not know how to route based on intent SID_1 directly will instead route to NID_1.

Iterative refinement: In this example, every node includes a direct edge to the intent, with fallback to domain and host-based routing. This allows iterative incremental refinement of the intent. If the CID_1 is unknown, the packet is then forwarded to NID_1. If NID_1 cannot route to the CID, it forwards the packet to HID_1.



An example of the flexibility afforded by this addressing is that an on-path content-caching router could directly reply to a CID query without forwarding the query to the content source. We term this *on-path interception*. Moreover, if technology advances to the point that content IDs became globally routable, the network and applications could benefit directly, without changes to applications.

Service binding and more: DAGs also enable application control in various contexts. In the case of legacy HTTP, while the initial packet may go to any host handling the web service, subsequent packets of the same "session" (e.g., HTTP keep-alive) *must* go to the same host. In XIA, we do so by having the initial

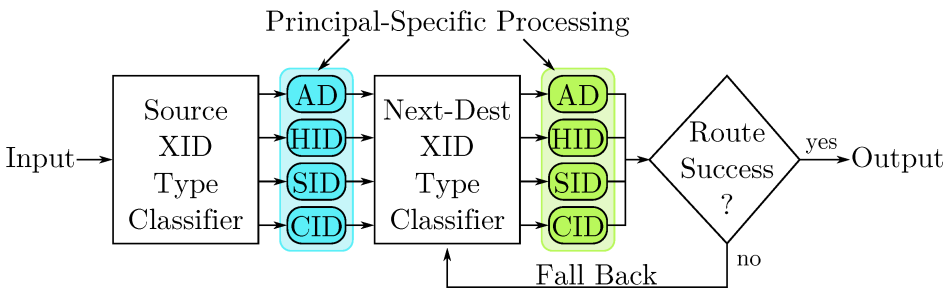


Figure 2: Packet Forwarding in an XIP router

packet destined for: $\bullet \rightarrow NID_1 \rightarrow SID_1$. A router inside NID_1 routes the request to a host that provides SID_1. The service replies with a source address bound to the host, $\bullet \rightarrow NID_1 \rightarrow HID_1 \rightarrow SID_1$, to which subsequent packets can be sent.

More usage models for DAGs are presented in [10]. It is important to view DAGs, and not individual XIDs, as addresses that are used to reach a destination. As such, DAGs are logically equivalent to IP addresses in today's Internet. This means that the responsibility for creating and advertising the address DAG for a particular destination, e.g., a service or content provider, is the responsibility of that destination. This is important because the desitnation has a strong incentive to create a DAG that will make it accessible in a robust and efficient way. Address look up can use DNS or other means.

The implications of using a flexible address format based on DAGs are far-reaching and evaluating both the benefits and challenges of this approach is an ongoing effort. Clearly, it has the potential of giving endpoints more control over how packets travel through the network, as we discuss in more detail in Section 1.8.

1.3.4 XIA Forwarding and Router Processing

DAG-based addressing allows XIA to meet its goals of flexibility and evolvability, but this flexibility must not come at excess cost to efficiency and simplicity of the network core, which impacts the design of our XIA router. In what follows, we describe the processing behavior of an XIA router and how to make it efficient by leveraging parallelism appropriately.

Figure 2 shows a simplified diagram of how packets are processed in an XIA router. The edges represent the flow of packets among processing components. Shaded elements are principal-type specific, whereas other elements are common to all principals. Our design isolates principal-type specific logic to make it easier to add support for new principals.

Before we go through the different steps, let us briefly look at three key features that differ from traditional IP packet processing:

- Processing tasks are separated in XID-independent tasks (white boxes) and XID-specific tasks (colored boxes). The XID-independent functions form the heart of the XIA architecture and mostly focus on how to interpret the DAG.

- There is a fallback edge that is used when the router cannot handle the XID pointed at by the last-visited XID in the DAG, as explained below.

- In addition to having per-principal forwarding functions based on the destination address, packet forwarding allows for processing functions specific to the principal type of the source address.

We now describe the steps involved in processing a packet in detail. When a packet arrives, a router first performs source XID-specific processing based upon the XID type of the sink node of the source DAG. For example, a source DAG $\bullet \rightarrow NID_1 \rightarrow HID_1 \rightarrow CID_1$ would be passed to the CID processing module.

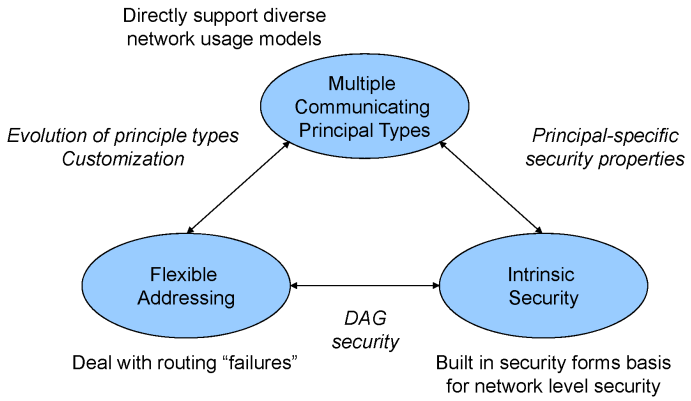


Figure 3: XIA as a combination of three ideas

By default, source-specific processing modules are defined as a no-op since source-specific processing is often unnecessary. In our prototype, we override this default only to define a processing module for the content principal type. A CID sink node in the source DAG represents content that is being forwarded to some destination. The prototype CID processing element opportunistically caches content to service future requests for the same CID.

The following stages of processing iteratively examine the outbound edges of the last-visited node (field LN in the header) of the DAG in priority order. We refer the node pointed by the edge in consideration as the next destination. To attempt to forward along an adjacency, the router examines the XID type of the next destination. If the router supports that principal type, it invokes a principal-specific component based on the type, and if it can forward the packet using the adjacency, it does so. If the router does not support the principal type or does not have an appropriate forwarding rule, it moves on to the next edge. This enables principal-specific customized forwarding ranging from simple route lookups to packet replication or diversion. If no outgoing edge of the last-visited node can be used for forwarding, the destination is considered unreachable and an error is generated.

We implement a prototype of the XIP protocol and forwarding engine. It also includes a full protocol stack. The status of the prototype is discussed in more detail in Section 12.1.

1.4 Deconstructing the XIA Data Plane

While we defined XIA as a comprehensive Future Internet Architecture, we also found it useful from a research perspective to view XIA as three ideas that, in combination, form the XIA architecture. This approach helps in comparing different future Internet architecture, in evaluating the architecture, e.g. by being able to evaluate the benefits of individual ideas in different deployment environments, and in establishing collaborations, e.g. by focusing on ideas rather than less critical details such as header formats. Figure 3 shows how XIA can be viewed as a combination of three ideas: support for multiple principal types, intrinsic security. XIA also proposes a specific realization for each idea: we have specific proposals for an initial set of principal types and their intrinsic security properties, and for flexible addressing. Other future Internet architecture proposals can differ in the ideas they incorporate and/or how they implement those ideas.

The ideas in Figure 3 are largely orthogonal. For example, it is possible to have a network that supports multiple principal types, without having intrinsic security or flexible addressing. Similarly, network can provide intrinsic security for a single principle type (e.g. as shown in AIP [11], which in part motivated XIA), while it is possible to support flexible addressing, e.g. in the form of alternate paths or destinations, without supporting intrinsic security. While it is possible to build an architecture around just one or two of the ideas, there is significant benefit to combine all three, as we did in XIA, since the ideas leverage each

other. This is illustrated by the edges in Figure 3:

- Combining multiple principal types and flexible addressing makes it possible to evolve the set of principal types since flexible addressing can be used to realize fallbacks, one of the key mechanisms needed to incrementally deploy new principal types (e.g., 4IDs [89]). As another example, this combination of ideas makes it possible to customize network deployments, without affecting interoperability, e.g. by only implementing a subset of the principal types in specific networks. Looking at possible deployments of XIA showed that is likely to be common, e.g., core networks will support only a limited number of XIDs (e.g., NIDs and SIDs), while access networks may also support CIDs, HIDs and possibly others. This customization can be done without affecting interoperability.

- Combining multiple principal types with intrinsic security makes it possible to define intrinsic security properties that are specific to the principal type. The benefit is that users get stronger security properties when using the right principal types. For example, when retrieving a document (static content), using a content principal guarantees that the content matches the identifier, while using a host-based principal to contact the server provides a weaker guarantee (the content comes from the right server).

- The benefits of combining flexible addressing and intrinsic security make it possible to modify DAG-based addresses at runtime securely without needing external key management. For example, when mobile devices move between networks, they can sign use the public key associated with the SID of each connection end-point to sign a change-of-address notice for each communicating peer. Similarly, service instances of a replicated service can switch to a new service identifier by using a certificate signed by the service identifier of the service.

Finally, the ideas in Figure 3 directly address the four components of our vision outlined in Section 1.1. Intrinsic security is a key building block of a trustworthy networks, in the sense that it guarantees certain security properties without relying on external configuration or databases; these can be used as a starting point for more comprehensive solutions. Multiple principal types, combined with flexible addressing, support evolution of the network, for example, to support new usage models or leverage emerging technologies.

1.5 Using Multiple Principals

We use a banking service to illustrate how the XIA features can be used in an application.

1.5.1 Background and Goals

A key feature of XIA is that it represents a single internetwork supporting communication between a flexible, evolvable set of entities. This approach is in sharp contrast with alternative approaches, such as those based on a a coexisting set of network architectures, e.g., based on virtualization, or a network based on an alternative single principal type (e.g., content instead of hosts). The premise is that XIA approach offers maximum flexibility both for applications at a particular point in time, and for evolvability over time. We now use an example of how services interact with applications using service and content principals in the context of an online banking service to illustrate the flexibility of using multiple principals. More examples can be found elsewhere [10, 51, 9].

1.5.2 Approach and Findings

In Figure 4, Bank of the Future (BoF) is providing a secure on-line banking service hosted at a large data center using the XIA Internet. The first step is that BoF will register its name, bof.com, in some out of band fashion with a trusted Name Resolution Service (Step 1), binding it to $AD_{BoF} : SID_{BoF}$ (Step 2). The service

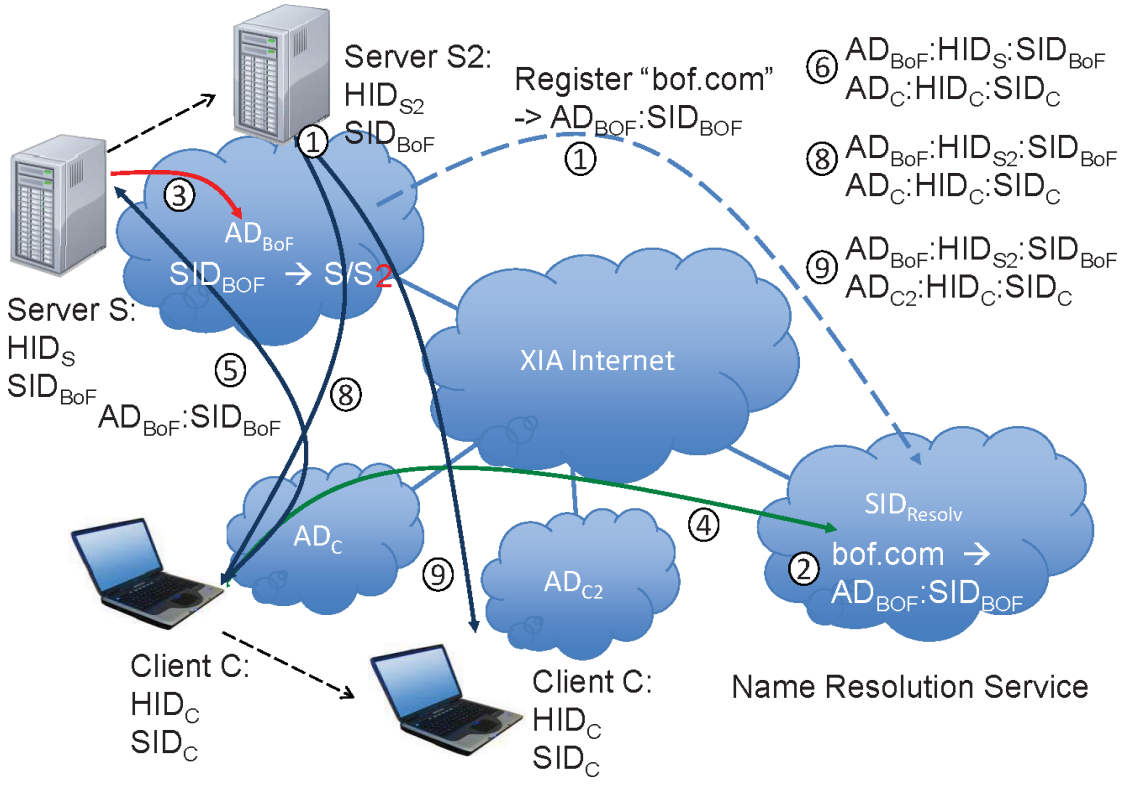


Figure 4: Bank of the Future example scenario.

ID SID_{BoF} to a public key for the service. One or more servers in the BoF data center will also advertise SID_{BoF} within the BoF's data center network, i.e. administrative domain AD_{BoF} (Step 3).

The focus of the example is on a banking interaction between a BoF client C (HID_C). The first step is to resolve the name bof.com by contacting the Name Resolution Service using SIF_{Resolv} (Step 4), which was obtained from a trusted source, e.g. a service within AD_C. The client now connects to the service by sending a packet destined to $AD_{BoF} : SID_{BoF}$ using the socket API. The source address for the packet is AD_C:HID_C:SID_C, where AD_C is the AD of the client, HID_C is the HID of the host that that is running the client process, and SID_C is the ephemeral SID automatically generated by `connect()`. The source address will be used by the service to construct a return packet to the client.

The packet is routed to AD_{BoF}, and then to S, one of the servers that serves SID_{BoF} (Step 5). After the initial exchange, both parties agree on a symmetric key. This means that state specific to the communication between two processes is created. Then the client has to send data specifically to process P running at S, not any other server that provides the banking service. This is achieved by having the server S bind the service ID to the location of the service, $AD_{BoF} : HID_S$, then communication may continue between $AD_{BoF} : HID_S : SID_{BoF}$ and AD_C:HID_C:SID_C (Step 6). Content can be retrieved directly from the server, or using content IDs, allowing it to be obtained from anywhere in the network.

The initial binding of the banking service running on process P to HID_S can be changed when the server process migrates to another machine, for example, as part of load balancing. Suppose this process migrates to a server with host ID = HID_{S2} (Step 7). With appropriate OS support for process migration, a route to SID_{BoF} is added on the new host's routing table and a redirection entry replaces the routing table entry on HID_S. After migration, the source address in subsequent packets from the service is changed to $AD_{BoF} : HID_{S2} : SID_{BoF}$. Notification of the binding change propagates to the client via a packet with an SID extension header containing a message authentication code (MAC) signed by SID_{BoF} that certifies the

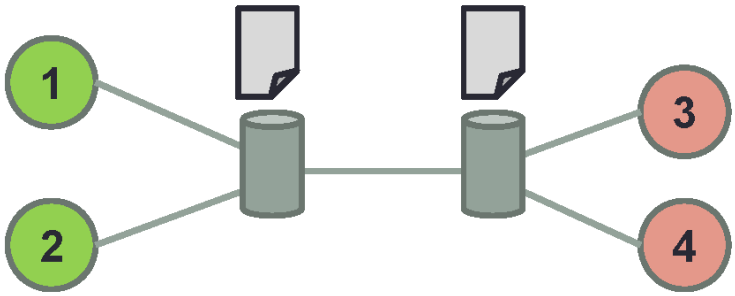


Figure 5: Multi-party conferencing example

binding change (Step 8). A client move to a new AD, AD_{C2}, can be handled the same way (Step 9). The new source address of the client will then be AD_{C2}:HID_C:SID_C (Step 10). When a new packet arrives at the server, the server will update the client's address. Even though locations of both the server and the client have changed, the two SID end-points –and therefore their cryptographic verifiability– remain the same.

1.6 Interactive conferencing systems

We use an interactive conferencing system as another use scenario for XIA. This research was performed by Wei Shi as part of his undergraduate honors thesis, advised by PI Peter Steenkiste.

1.6.1 Background

With the availability of the XIA prototype (and its Xsocket API) as a platform for application development, we started to develop various user applications over XIA. Among them, we started to explore how to best support multi-party interactive conferencing, called Voice-over XIA (VoXIA). This serves a good example to show how the flexibility of using multiple principal types in XIA can deliver benefits for conversational communication applications that are important in the Internet today.

1.6.2 Approach and Findings

We compared different design options for the VoXIA appliation. We implemented a basic application with the following features: (1) each node uses the same (predefined) SID_{VoXIA} for the VoXIA application; (2) each node registers its own unique VoXIA identifier (VoXIA ID); (3) and binds this VoXIA ID to its DAG, e.g., $AD : HID : SID_{VoXIA}$. For the voice software stack, we used the Speex protocol for compressing audio frames and PortAudio for recording and playing back audio.

In order to examine whether the use of multiple principal types in XIA can offer advantages over traditional host-based communication in this context, we developed two different VoXIA designs. The first method is to send and receive all frames via XDP sockets (UDP style) and thus, no in-network caching is supported. This serves as a measurement baseline, which represents the traditional host-based communication. The second one is to use XDP sockets only for exchanging a list of frame pointers (CIDs) and use XChunkP sockets (chunk transmission) for retrieving the actual voice frames. This method is supported by in-network caching, as XChunkP communication uses the content principal type. VoXIA experiments over GENI (five end-hosts and two routers) show that the chunk-style VoXIA is advantageous when multiple nodes request the same set of frames since they can effectively utilize the in-network caching. Figure 5 shows an example with four nodes that are are geographically distributed. After Node 3 retrieves a frame from Node 1, it is likely to be available in the content caches of the routers, where it can be efficiently retrieved by Nodes 2 and 4.

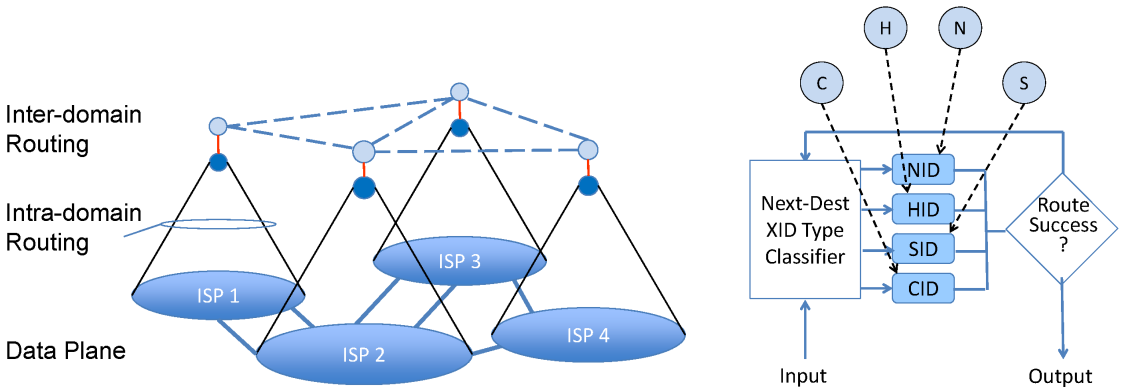


Figure 6: Control Plane Architecture (left) and Routing Protocol Design (right).

The VoXIA experiments showed that the XIA flexibility of using multiple principals can deliver benefits for conversational communication applications. VoXIA can use principal types that match the nature of the communication operation: SIDs for signaling path and CIDs for data path. The experiments also showed that it would be useful to allow nodes to push content chunks in a way that they can be caches along the way (the the current XchunkP transport protocol only allows fetching chunks). Finally, a robust conferencing application will have to deal with issues such as pipelining of chunk requests for future data and synchronization among nodes. These issues are similar to those faced by content-centric network architectures such as NDN [60].

1.7 XIA Control Plane Architecture

As a first step towards a control plane architecture, we designed and implemented a flexible routing architecture for XIA. The designed is based on the 4D architecture [49, 121] which argues that manageability and evolvability require a clean separation separation between four dimensions: data, discovery, dissemination and decision. Software-defined networks have become a popular control plane architecture for individual networks (i.e., intra-domain) and they are based on these 4D principles. XIA is exploring how the 4D architecture can be used as the basis for an inter-domain control plane.

Our initial design is shown in Figure 6(left). We assume that each domain is a software-defined network that uses a logically centralized controller. This controller runs separate applications for intra-domain and inter-domain routing. The inter-domain routing application communications with the routing application in neighboring domains, creating the control plane topology for inter-domain routing that can be used to implement the discovery and dissemination dimensions. Note that this topology is significantly simpler than the topology used by, for example, BGP, which uses a router-based topology.

For routing in XIA, we have logically separate routing protocols for each XID type supported at the intra or inter-domain level, each responsible for populating forwarding tables for intra or inter-domain communication respectively. These are implemented as applications on the SDN controller (Figure 6(left)). The implementation of the routing applications can potential share functionality to reduce the implemention effort. Our implementation supports stand-alone inter-domain routing for NIDs while the protocols supporting Scion (Scion IDs) and anycast for replicated services (SIDs) leverage the universal connectivity supported by NIDs routing and forwarding.

Ongoing research will refine this initial control plane design by providing richer discovery and dissemination dimensions that simplify both the implementation of new control protocols and the extension of existing protocols (evolvability).

As part of the original XIA project, the Scion contributed two addtional pieces of technology that will

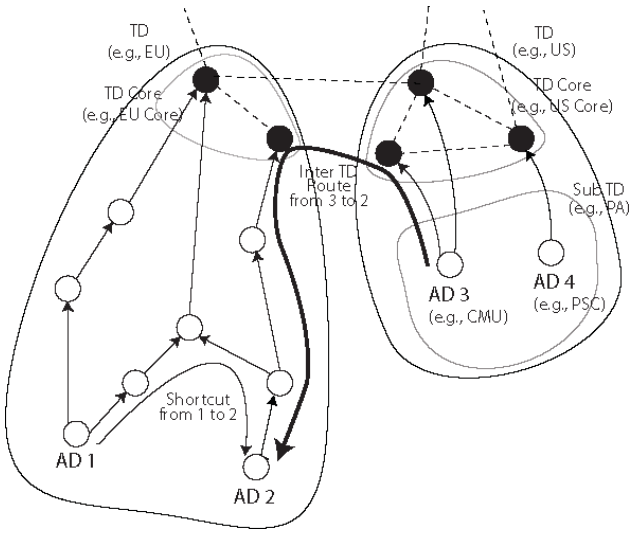


Figure 7: Scion trust domains, beaconing, and path selection.

be incorporated into the XIA control plane. First, Scion [124, 57] supports path-based communication in which packets are forwarded based on a previously allocated inter-domain path (Figure 7). For each edge domain, the ISPs establish a number of paths connecting that domain to the core of the Internet (sequence of straight arrows in the figure). These paths adhere to the business policies of the ISPs and may have different properties. Two domains can then communicate by choosing one of their paths and merging them to form an end to end path (e.g., AD1-AD2 and AD3-AD2 in the figure). Scion path based communication uses a new XID type, Scion IDs, and and edge networks can choose between forwarding based on destination NID or Scion IDs. The latter are more secure and offer more path control than NIDs.

Second, Scion introduced several new concepts and mechanisms to securely establish and maintain paths. First, it introduced Trust Domains (TD) as a way of reducing the Trusted Computing Base for paths. This is achieved by grouping in domains in independent routing sub-planes (two in the figure). Second, it uses a beaconing protocol to identify possible paths; beacons are created by the domain in the TD root and forwarded towards the edge network. Finally, Scion uses a trust domain infrastructure in which domain share certificates to authenticate themselves. While these techniques were designed within the context of Scion, they are independent of the Scion path selection protocol and we plan to use them to enhance the security of XIAs inter-domain control plane.

1.8 Exposing and Using Control Points

The XIP protocol effectively represents the dataplane interface between network users and domains. It represents a control point that allows actors to engage in a tussle "at runtime" to define a communication agreement that meets their policy, economic, legal and other requirements [27].

Compared with IP, XIP offers significant flebility in how communication operations are performed (fourth point of the vision in Section 1.1). First, multiple principal types give users (applications or human users) a choice of how to use the network. This choice affects not only performance but also privacy (CIDs potentially expose more information) and control. Similar, service providers have a choice in what principal types to support, e.g., based on economic considerations. Second, DAGs as flexible addresses that are much more expressive than today's Internet addresses, given users a lot more control over how a packet is handled, through the use of fallbacks, weak source routing and possibly the specification of multiple paths or multiple sinks [10]. Finally, SCION offers the user a choice over the path that its packets will take through

the network, as we explain next.

Experience with XIA so far shows however that flexible data plane interfaces are not sufficient to build a network that is evolvable and flexible. We also need to rethink the control plane interfaces for routing for the various principal types and resource management on a variety of time scales (congestion control, traffic engineering). Appropriate interfaces are needed not only between the ISPs but also between ISPs and their customers (residential and corporate networks, service providers, CDNs, etc.). While we have done some initial research in this area, this a central theme in our current research.

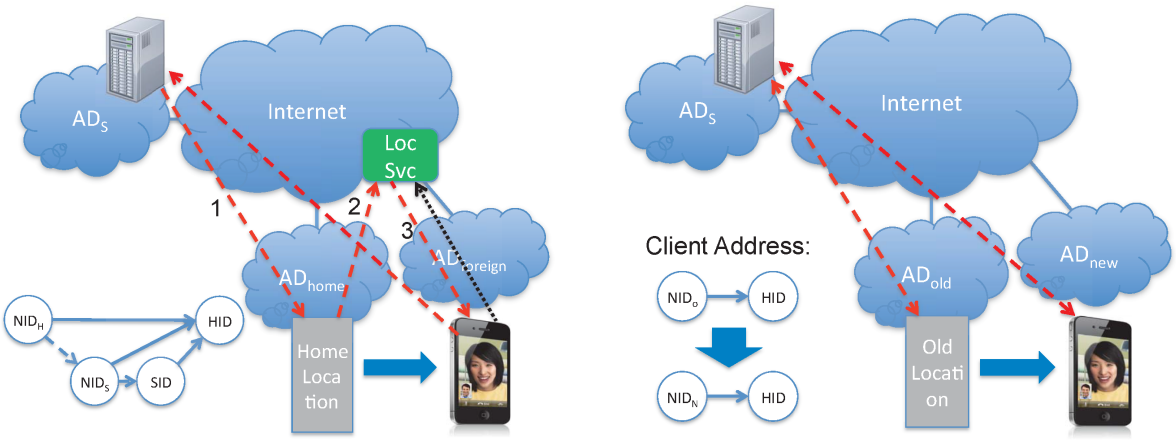


Figure 8: Supporting mobility: first contact (left) and during a session (right)

2 Network Challenges

The XIA architecture has been used to address a number of traditional network challenges, including mobility, incremental deployment of architectures, anycast, support for new architecture concepts, in-path services, and in-network caching.

2.1 Supporting Mobility

We designed and implemented protocols to both establish and maintain a communication session with a mobile end-point, leveraging XIA's DAG-based addresses, service identifiers, and intrinsic security. This research was done by Nitin Gupta and Peter Steenkiste from CMU.

Background and Goals - Supporting cross-network mobility in the current Internet is challenging for a simple but fundamental reason: the IP protocol uses IP addresses both to identify an endpoint ("identifier") and as a topological address used to reach the endpoint ("locator"). When a mobile devices switches networks it must change the IP address that is used to reach it, but in IP this means that its identifier also changes so any peers it is communicating with will no longer recognize it as the same device.

There has been a great deal of work on supporting mobility in the Internet, e.g., [122, 48], to overcome this fundamental problem, but two concepts have emerged as the basis for most solutions. The first concept is the use of a rendezvous point that tracks the mobile device's current address that can be used to reach it (its locator). An example of a rendezvous service is the home agent in the mobile IP protocol. The second is the observation that, once a connection is established, the mobile host can inform its stationary peer (or its point of contact) of any change in location. This is for example done by the foreign agent in mobile IP. Both of these concepts are fairly easy to implement in XIA, in part because XIA distinguishes between the identifier for an endpoint (HID SID, or any XID type) and its locator (an address DAG). Note that MobilityFirst uses a similar split with globally unique host identifiers (GUIDs) that are resolved to locators [105]. We now describe our implementation of each concept for host mobility.

Approach and Findings - In our design, when a mobile device leaves its home network, it needs to do two things. First, it needs to make sure that it registers a DAG address with the name service that includes a rendezvous service; we discuss possible DAG formats below. Second, as it connects to different foreign networks, it needs to keep its rendezvous service (marked as Loc Svc in igure 8(left)) informed of its current location by registering its DAG address (dotted arrow in the Figure). This will allow the rendezvous service to keep an up-to-date forwarding table with entries for all the mobile devices it is responsible for.

To establish a communication session with a mobile device, a corresponding client will first do a name lookup. As shown in Figure 8(left), the address DAG that is returned could include the device's "home" locator ($NID_H : HID$) as the intent, and the location of a rendezvous service ($NID_S : SID$)) as the fallback. If the device is attached to its home network, it is delivered by following dashed Arrow 1 in the DAG. If not, it is automatically delivered to the rendezvous service using the fallback (Arrow 2). The rendezvous service could be located anywhere. For example, it could be hosted by the home network as in mobile IP, or could be a distributed commercial service. When it receives the packet, the rendezvous service looks up the current DAG address for the mobile device and forwards the packet (Arrow 3). The mobile device can then establish the connection with the communicating client. It also provides the corresponding host with its new address as described below, so future packets can be delivered directly to the new address. Clearly, other DAG formats are possible. For example, the mobile device could register the DAG of its rendezvous service with its name service when it leaves its home network, so connection requests are directly forwarded to the rendezvous service.

The above solution is simple but it is not secure. A malicious party can easily intercept a session by registering its own DAG with the rendezvous service, impersonating the mobile device. The solution is to require that address registrations with the rendezvous are signed by the mobile device. This can be done in a variety of ways. For example, the mobile device can receive a shared key or other credentials when it signs up with the mobile device. In XIA, we can leverage intrinsic security. The mobile device uses the public key associated with SID in its "home" DAG address so the rendezvous service can verify the authenticity of registration without needing a prior agreement with the mobile device. We use the SID to sign the registration since it is typically the "intent" identifier, i.e., it is the true end-point associated with the DAG. In contrast, HIDs and NIDs in the DAG can change over time.

The implementation of the rendezvous service raises two design questions. First, where should it be implemented in the protocol stack. Logically, it acts as a layer 3 forwarding element. It uses a forwarding table to forward packets at layer 3, but that table is directly filled in by endpoints, rather than by a traditional routing protocol. However, for simplicity of code development, we decided not to implement it at layer 3 inside click. Instead, it is is an application level process that uses the raw Xsocket interface to send and receive layer 3 packets (that include the XIA header). A second design decision is how the rendezvous service forwards packets to the client. Options include dynamic packet encapsulation or address rewriting. We decided to use address rewriting, i.e. replace the client's old address with its new address, because it does not affect packet length, does avoiding packet fragmentation issues.

To maintain an active session during mobility, the mobile host keeps the stationary host informed of its new address [51], as shown in Figure 8(right). Again, we must decide where to implement this functionality. For network-level mobility, layer 3 would be the natural place in the protocol stack. Unfortunately, since XIP is a connection-les protocol, it does not know what active (layer 4 and higher) sessions exist. In our solution, the XIP protocol prepares a change-of-address record, which is passes on to any software module (transport protocol, applications, etc.) that registered an interest in learning about address changes. These modules are then responsible for forwarding the address change to any relevant parties. In our implementation, the reliable transport protocol inserts the change of address record in the data stream, marked as control information. The change of address record is signed using the private key of the SID associated with The endpoint of the connection, as explained earlier.

Status and future work - We have implemented both mechanisms and have shown that they work. Both mechanisms are very responsive, and the current bottleneck in network-level handoff is the old and slow XHCP protocol that is currently used to join a new network. Future work will focus on optimizing the protocol used to join a network and transport and session level support to optimize network performance during network mobility.

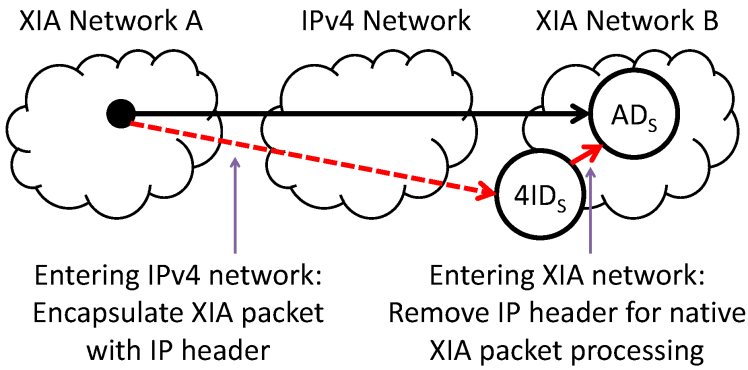


Figure 9: Using the 4ID principal type to incrementally deploy XIA

2.2 Incremental deployment

We explored whether XIA's flexible addressing could be used to help with incremental deployment of XIA nodes and networks in IPv4 networks. This research was performed by CMU Ph.D. student Matt Mukerjee, advised by PIs Seshan and Steenkiste.

Background - One of the bigest hurdles facing a new architecture is deployment in the real world over an existing infrastructure. For example, several past designs including IP Multicast, different QoS designs, and IPv6, have faced significant deployment challenges over the current IPv4 infrastructure. The most common approach that past systems have taken is the use of an overlay network. Both IPv6 and IP Multicast operate a virtual backbone - the 6bone and Mbone, respectively. Users who wish to make use of IPv6 or IP Multicast must create an overlay network link or tunnel that connects them to the backbone. Unfortunately, creating such links is a complicated process and this has limited the wide adoption of these protocols. Recent efforts [86] have tried to automate this process; however, other weaknesses associated with a tunnel-based scheme remain.

By analyzing the range of IPv6 deployment methods, it quickly becomes clear that any proper deployment method must have certain qualities. A first key requirements is that there should be **no explicit tunneling** between disjoint clouds since it is a fragile. Also, there should be **no address mapping** that takes old addresses and puts them into the new address space. Allowing this severely limits the usefulness of the new network to the constrains of the old network. Desirable properties include minimal configuration, automatic adaptation to unforeseen changes in the underlying network topology or failure of nodes/links, and graceful degradation.

Our initial design to address these requirements in XIA introduces a new XID type that we call a 4ID (named for IPv4 ID). Consider a scenario with two nodes, A and B, that are located in different XIA-based networks attached to the IPv4 Internet at different locations. Each of the two networks has a least one node that operates as a dual-stack router (i.e., a router that connects to both the XIA local network and the IPv4 network). In order for A to transmit a packet to B, the destination DAG address will contain the IPv4 address of B's network's dual-stack router as a fallback address. This entry will be encoded inside and XIA address with the type labeled as a 4ID. This design takes advantage of the fact that XIA packets can carry multiple addresses and encode relative priorities across them. In addition, unlike past designs, there is no use of a virtual backbone and no need to setup tunnels.

Figure 9 illustrates the example. The dual stack router in network A will first try to forward the packet based on the intent identifier, AD_S, but it does not not have a forwarding entry AD_S, so it needs to use the fallback. After encapsulating the packet into an IPv4 packet using the IPv4 address enclosed in $4ID_S$, the packet can be delivered over the IPv4 network to AD_S. The dual stack router in network B will remove the IPv4 header and forward the packet to the destination using XIP. Once the IPv4 network is upgraded to XIA,

the same DAG can still be used to reach the destination.

Approach and Findings - We generalized our understanding of incremental deployment techniques by creating a design space based on how different approaches addressed four core questions:

- How and when to select an egress gateway from the new network architecture (NNA) source network
- How and when to select an ingress gateway into the destination NNA network
- How to reach the egress gateway of the source NNA network from the source host
- How to reach the ingress gateway of the NNA destination network from the source NNA network

Based on the above design space, we were able to map all existing designs into two broad classes of solutions. In addition, we were able to identify two new classes of designs. The 4ID-based design represents one of these classes. We created a new design that we called "Smart 4ID" as an example of the second new class. The 4ID mechanism utilizes new data plane technology to flexibly decide when to encapsulate packets at forwarding time. In contrast, the Smart 4ID mechanism additionally adopts an SDN-style control plane to intelligently pick ingress/egress pairs based on a wider view of the local network.

To characterize the real-world performance tradeoffs of the four classes (both new and old), we implemented a representative mechanism from each class and performed wide-area experimentation over PlanetLab [23]. We explored how the choices made in each class directly impact host performance (path latency, path stretch, and latency seen by applications) as well as failure semantics (failure detection and recovery time) through a quantitative analysis. We additionally provide a qualitative analysis of management and complexity overhead of each mechanism. Path latency and stretch provide insight into the quality of the path chosen by each mechanism, whereas application latency shows path impact on hosts. Failure semantics and management/complexity overhead present a fuller picture of the effort needed to use these mechanisms, which is often left out in analysis.

Our results shows that the new Smart 4ID-based approach outperforms previous approaches in performance while simultaneously providing better failure semantics. We contend that the our mechanism performs better because it leverages innovations in the data plane (flexible addressing) and the control plane (centralized local con- troller) rather than relying solely on traditional ideas (routing, naming, etc).

2.3 Service anycast routing

We designed and implemented an inter-domain anycast routing protocol for XIA service identifiers (SIDs).

Background - SIDs represent the end-points of communication sessions (i.e., sockets). Similar to the current Internet, SIDs can be "well-known", i.e., advertised by service providers, or ephemeral, e.g., for clients. Since services can be replicated for performance and availability, an initial service request based on an SID has anycast semantics: the network must deliver the request to exactly one service instance. The choice of service instance is highly service-specific and will depend on both the state of the infrastructure (bandwidth, latency, server load, ..) and provider policies. Once the service request is received by a service instance, the endpoint of the connection is "bound" to that instance by inserting the NID of the service instance in the address DAG. Intrinsic security is used to make this step secure.

The requirements for SID anycast can be summarized as follows:

- **Flexibility** Decision making should be flexible enough to meet the requirements of wide variety of Internet services.
- **Accuracy** The choice of the service instance should accurately reflect that conditions of the server and the network.

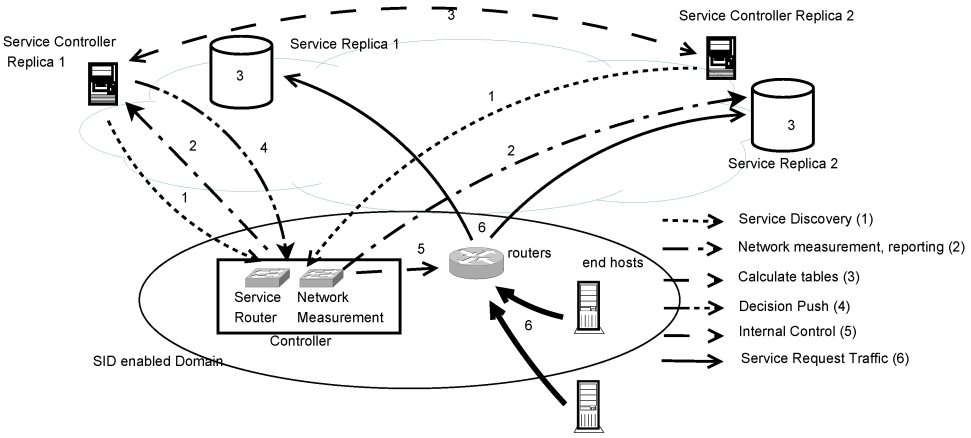


Figure 10: Architecture for SID anycast.

- **Responsiveness** Decision making should be responsive to changes in the system conditions.
- **Efficiency** Initial request packets should be handled efficiently since many service sessions are short-lived.
- **Scalability** The system must have low overhead so it can scale to a potentially large number of services and client domains.
- **Availability** The system must have high availability, even if parts of the anycast architecture fail.

Related work in this area falls broadly in two categories. The first one is exemplified by IP anycast [93], which supports anycast over the current Internet. Its current implementation [2, 66] leverages BGP [101]. This means that instances are selected using BGP-type criteria (AD path length, etc.). While this may be sufficient for DNS [55], it does not meet the requirements of more demanding services.

The most commonly used anycast service today is DNS-based. Specifically, the DNS server of the service provider will selected a service instance and return it in the form of an IP address to the client. This approach has the advantage that the service provider is fully in control of the selection process and can apply any policy they want. There are however a number of limitations and drawbacks. First, the DNS server does not have access to any network information; all it knows is an estimated location based on the client's request. Second, it adds a roundtrip worth of latency to service access. Finally, this approach introduces a fair bit of overhead since caching of DNS responses has to be limited to make the system responsive to failures and changes in load.

Approach and Findings - The key idea underlying our design that we want to place instance selection in the hands of service providers so rich policies can be implemented, but instead of relying on DNS, we want to use routing so we can incorporate more accurate information about network conditions and avoid the roundtrip latency associated with a DNS lookup.

The high level design is shown in Figure 10. The figure shows an SID-enables domain with an SDN style control plane, consistent with the XIA control plane architecture (Section 1.7). The SDN controller runs an SID routing application, *service router*, that, together with a set of *service controllers* that are colocated with each service instance for a set of SIDs, implement anycast. Consistent with the 4D architecture [49], the service routers and controllers together implement discovery, dissemination and decision functions.

As a first step, service controllers will use beacons to advertise their presence, allowing service routers to identify available service instances and establish communication sessions with their service controllers (step

1). Next, a network measurement application in each SID-enabled domain will collect network performance data (e.g., latency and bandwidth to service instances) and make this information available to the service controllers; we discuss the placement of SID-enabled domains later. The services controllers for each SID will then jointly decide on how service routers should forward SID requests to service instances using both the information provided by the service controllers and internal information such as service load and policies (step 3). They then forward these decisions to the service routers (step 4), which will distribute it to routers in their domain as needed (step 5). SID requests from clients will now be forwarded to the appropriate service instances without incurring additional latency (step 6).

Let us elaborate on some of the key steps. First, the simplest outcome of the decision step (step 3) would be to create a forwarding entry for an SID that points at a single service instance. Unfortunately, unless the service provider has a presence in a large number of SID-enabled domains, which may not be practical for smaller service providers, this simple approach would limit load balancing options. For this reason, we are exploring an alternative design where a forwarding entry for an SID can include multiple service instances with a target load distribution. Second, our design assumes that the information collected in step 2 is representative for clients. This requires SID-enabled domains to be at the edge of the Internet, near or even inside client networks. Clearly, increasing the number of domains will improve performance. Finally, in order to keep track of changing network and server conditions, routing updates will happen periodically. Service controllers can also push updates at any time, for example to respond to failures of sudden changes in the network or load. This should improve both responsiveness and availability.

This design meets the requirements we identified above by combining the benefits of the DNS-style approach with those of a routing solution.

2.4 Pushing the evolvability envelop

The XIA group at Boston University has realized a Linux implementation of the core XIA network architecture and has been exploring how other FIA features can be integrated into Linux XIA at the network layer. The primary goal of this effort is fourfold: to streamline upcoming network deployments of XIA, to provide a concrete demonstration of XIA's capability to evolve and import novel architectural functionality created by others, to facilitate other network researchers to conduct architectural evaluations enabled by the existence of Linux XIA, and to reach beyond network researchers to other potential early adopters of XIA. This effort has been spearheaded by post-doctoral researcher Michel Machado and PI John Byers. Other team members contributing in this reporting period are PhD students Cody Doucette and Qiaobin Fu, undergraduate Alexander Breen, and Boston Academy High School student Aaron Handelman.

Background - In the FIA-funded FIA XIA project, Boston University implemented an XIA protocol stack natively in Linux and started to use this implementation as a basis for realizing other FIA architectures as principals within the XIA framework. As described in Michel Machado's Ph.D. thesis [82], the team ported the Linux implementation of the Serval service-centric architecture to XIA, and produced white paper ports of the NDN architecture, as well as the MobilityFirst architecture, and the ANTS active network protocol. XIA was able to accomodate these "foreign" architectures by introducing new principal types and using the flexible addresses, as described in previous annual reports. Another interesting take-away finding was that realizing Serval in XIA directly activates intrinsic security, cleanly remediating a key weakness in the original "Serval over IP" design.

Findings - In this reporting period, the Boston team continued to build out functionality on Linux XIA, providing support for multicast using the zFilter framework from the PURSUIT project (a European future architecture), and focusing on interoperability. The goal was to evaluate how effectively and efficiently XIA's architectural principles supports networking features, in this case supporting multicast over a legacy network. This research will be presented at ANCS'15 [83]

We built a reliable multicast application that combines three principals to deliver content across a hetero-

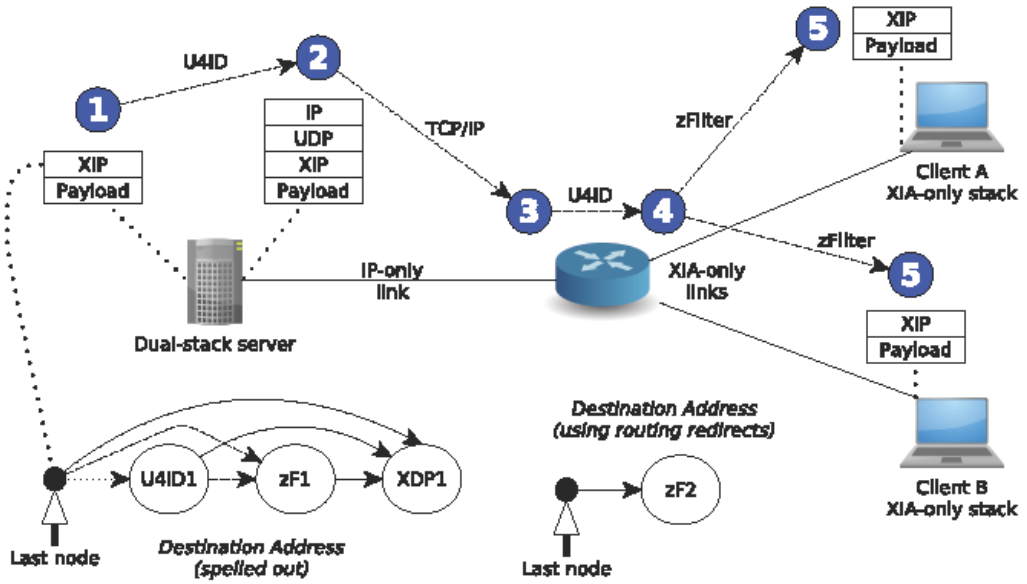


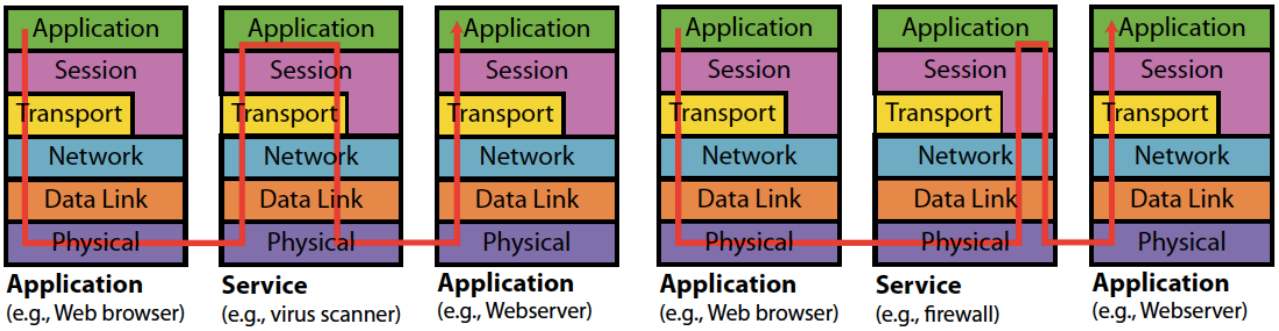
Figure 11: Example of network evolvability and principal interoperability in Linux XIA.

geneous network, to demonstrate the value of principal interoperability. This application employs the U4ID principal to cross an IP-only link, the zFilter principal to duplicate packets in the network, the XDP principal to deliver packets to sockets, and erasure codes to make the transmission reliable. Figure 11 illustrates this application in action, and shows how these principals can compose an XIP address.

The three-node destination address depicted at bottom left in 11 can be understood as expressing the following intent: (1) traverse an IP-only part of the network by encapsulating XIA packets in UDP/IP payloads; (2) multicast the content to multiple hosts; and (3) deliver the content to listening datagram sockets. Alternatively, the depicted single-node destination address can be used in tandem with routing redirects in the network to supply the same functionality. In both cases, this allows the TCP/IP, zFilter, and XIA architectures to interoperate by composing their individual strengths, despite the fact that these architectures were never intended to work together.

Each step in Figure 11 captures a transition in the life of an XIP packet being sent from the server to the clients. The XIP packet is created (Step 1) but while routing the packet, XIP discovers that it cannot forward the packet using the XDP_1 identifier since the link between the dual stack server and the router is IP-only, so XIP transfers control to the U4ID principal, which encapsulates the XIP packet into the payload of a UDP/IP packet (Step 2). The packet is forwarded using IP and arrives on the router, where it is decapsulated and control is handed back to the *U4ID* principal (Step 3). IP decides on following the edge zF_1, which leads to duplicating the packet (Step 4), and sending the copied packets toward the two clients. Once the packets arrive at the clients (Step 5), the XDP principal identifies the listening datagram sockets to which the data must be delivered. This application serves as a proof of concept that XIA has a strong notion of evolvability and that Linux XIA offers a practical realization of allowing interoperation and collaboration between multiple new principal types.

The lessons learned from porting various FIA features to XIA show that XIA can be viewed as an interoperable meta network architecture [83]: a framework that nurtures coexistence of clean-slate designs, letting stakeholders experiment with, improve, and choose the designs that best suit their needs. In this architectural framework, future innovation can occur on the level playing field provided by the XIA framework, lowering barriers to entry, and thus enabling crowdsourced innovation. Our vision is that Linux XIA



(a) Application-Level Service (b) Network-Level Service

Figure 12: XIA network stack with in-path services and a session layer.

makes (formerly intractable) network layer innovation broadly accessible to networking researchers.

2.5 In-Path Network Services

In this section, we explore how to effectively support in-path network services. This project is motivated by, and extends, earlier work on Tapa, transport support for communincation over heterogenous edge networks, which is summarized in Section 3.1. This research was performed by David Naylor, in collaboration with Suk-Bok Lee (postdoc) and PI Peter Steenkiste.

2.5.1 Background

Today's Internet does a poor job handling in-path services, i.e. services data should pass through on its way to a destination. There is currently no mechanism for including in-path services in a communication session without building an overlay on TCP/IP or physically (and transparently) inserting the service into the data path. These transparent middleboxes often introduce new problems, such as creating new failure modes that violate fate sharing.

The goal of this work is to design and implement support for in-path services in XIA such that: (1) all involved parties (applications, hosts, and ADs) can add services to a communication session; (2) all in-path services are visible to all communicating parties (eliminating hidden failures and allowing policy verication); (3) paths do not have to be symmetric; (4) applications are required to implement as little of the service management/error recovery functionality as possible (i.e., the network handles as much as possible).

2.5.2 Approach and Findings

In order to realize the above design goal for in-path services, we proposed adding a *session layer* to the XIA stack. The session layer manages all in-path services in a session. It solicits input from all involved parties as to which services should be included and sets up a session by initiating transport connections between application-level services. Figure 12 shows the XIA stack with a session layer that supports in-path services, either at the application level (Figure 12(a)) or at the network level (Figure 12(b)). Application level services sit above the application layer, while the session layer provides what are essentially raw sockets to network-level services, bypassing the transport layer.

The initiator of a session specifies its *final intent*, or the other endpoint, and optionally a set of *in-path services*. We use the term *path* to describe the ordered set of services through which data should pass between the initiator and the final intent in one direction, including either the initiator in the forward path or the final intent in the return path. A *session*, then, is simply two paths: one from the initiator to the final

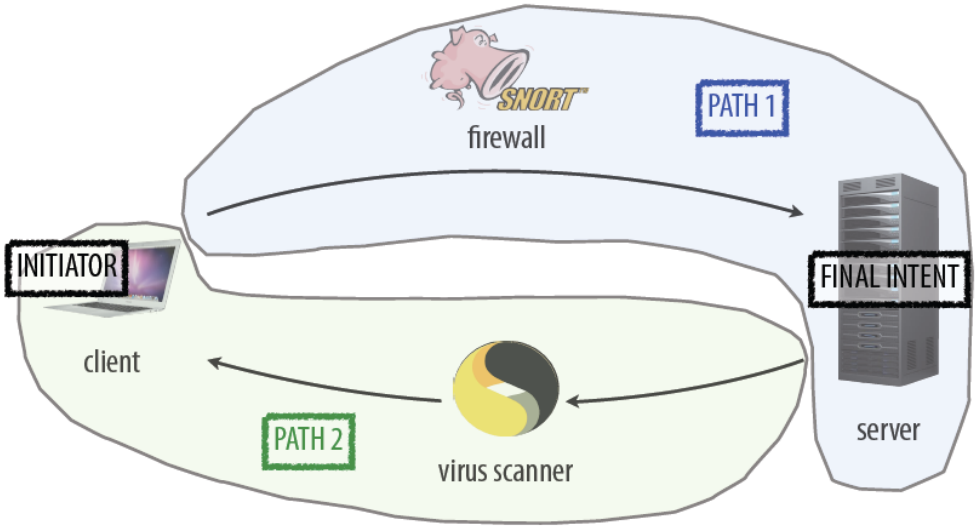


Figure 13: Example of in-path services in XIA.

intent and one from the final intent back to the initiator. The two paths may or may not be identical. We call the DAGs describing these paths the *session DAGs*. Figure 13 shows an example session where the virus scanner is a application-level service (it is a transport endpoint) while the firewall is a network-level service (it is not a transport endpoint). In this case, the session consists of two asymmtric paths that include different in-path services: one from the initiator to the final intent and the other from the final intent back to the initiator. Black arrows indicate transport connections.

We have designed and implemented the session management procedure with three distinct phases: DAG creation, service discovery and DAG negotiation, and data transfer. For DAG creation phase, the initiator must construct two DAGs: one encoding path 1 (from initiator to final intent), the other encoding path 2 (final intent back to initiator). The session layer on a given host uses these session DAGs to make a network-level DAG for the next hop in the path. During the service discovery and DAG negotiation phase, the session layer sets up individual transport connections between consecutive application-level services; which transport protocol it uses is determined by flags passed by the application when the session socket is created. The session *binds* to the "discovered" services. During this phase, each party also has the opportunity to make changes in the session DAGs, e.g., adding some in-path services of its own. It does so by modifying the session DAGs before forwarding the session information packet to the next hop. When the session information packet returns to the initiator, the session layer at the initiator checks for changes in the session DAGs and decides whether the session can be established, or whether further negotiation is needed. Once a session is set up, we propose transmit data in the session as ADUs, similar to Tapa. The reason is that is useful for some services to know ADU boundaries, and it also siplifies end-to-end reliable delivery, since we can no longer use byte or packet sequence numbers when in-path services might be adding or removing bytes/packets.

We have also explored content retrieval via in-path services: path 1 is the sequence of services your content request packets should visit (e.g., a packet scheduler) and path 2 describes the services through which the content itself should pass when returning to the requester. The idea is to set up a normal XSP (stream) session over these paths; a content request traverses path 1 while the content itself returns to the initiator via path 2. We note that the session layer at the first service in path 2 will request the content (to the network), and upon receiving, publish a modified chunk if the content has changed. A service may optionally store CIDs it has processed in a database which maps them to the CID of the modified content

it produced in the past. If it does, then if it receives a request to process a piece of content it has seen in the past, it can immediately send the CID of the modified content to the next hop rather than requesting and processing the chunk again.

2.6 Trace-driven analysis of ICN deployments on video workloads

The XIA data plane can support a richer set of communication abstractions including traditional host-to-host and emerging content-centric routing paradigms using CIDs. As part of the XIA video use case, we would like to understand how to optimally utilize these capabilities and how to deploy and configure the XIA data plane. This work was led by Prof. Sun Yi, a visiting XIA collaborator from the Chinese Academy of Sciences in collaboration with Seyed Fayaz, a student advised by PI Vyas Sekar [109].

Background and Goals - The motivation for CID in XIA and other future Internet architectures is driven in large part by the dramatic rise in Internet video traffic volumes over the last several years [24]. In particular, the promise of ubiquitous caching and simplified network traffic engineering that CID-based ICN architectures offer have a natural synergy with the quality expectations and bandwidth demands of Internet video workloads. In this context, there are two natural questions that arise with respect to the interaction between CID routing and Internet video:

- *How do video workloads impact network-level improvements offered by different caching strategies?*

  The canonical improvement metrics studied in the ICN literature include cache hit rate, origin server load reduction, and the reduction in the overall network footprint [45]. There is a very rich literature on how caching can be using to optimize these metrics including work on *content placement* to decide which routers on a request path should cache the content (e.g., [76, 75, 41, 98, 22]) and *content replacement* to decide how to manage the cache storage when full (e.g., [87, 3]). We would like to understand what specific placement and replacement algorithms work well for video workloads and what magnitude of improvement they can offer.

- *How do caching mechanisms impact key video-specific QoE metrics?*

  Unlike traditional Internet web workloads, Internet video introduces new QoE metrics that impact user engagement such as buffering ratio, startup latency, and average bitrate [33, 74]. Going beyond the aforementioned network-centric metrics, we would also like to understand how different ICN caching strategies impact these key video-specific QoE metrics.

Despite the tremendous interest in both ICN and Internet video, there has been little work on systematically answering these questions with actual video workloads and using real network topologies. Much of the ICN literature today has focused on generic Zipf-like workloads and on small-scale topologies [99, 102, 117]. Similarly, most of the Internet video literature focuses on traditional CDN-based delivery architectures [33, 81, 14, 74, 58]. Thus, there is a gap in our understanding of the interplay between the design space of ICN content placement and replacement strategies and QoE for video workloads.

Approach and Findings - Our work bridges this gap using large-scale video on demand (VOD) request traces consisting of over 196 million video requests from over 16 million users from the PPTV deployment in China [97]. We evaluate a combination of seven CID-based content placement [76, 75, 41, 98, 22], five content replacement schemes [87, 3], and four different cache sizes (1GB, 10GB, 100GB, and 1TB). For each of these scenarios, we run trace-driven simulations on a country-wide router-level topology with around 80K routers. To the best of our knowledge, this is the largest (in terms of the topology and number of requests) and most comprehensive (in terms of the design space) trace-driven analysis of CID/ICN caching strategies on video workloads.

Running such a large-scale workload on a large topology and a broad range of scenarios raises significant scalability challenges for existing ICN simulation frameworks [5]. To enable this analysis, we develop a

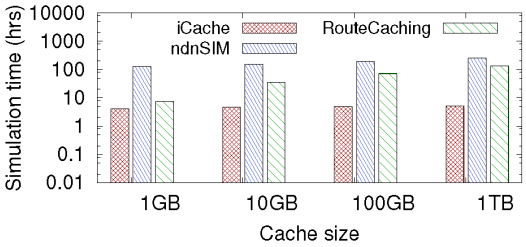


Figure 14: Simulation time of iCache and ndnSIM showing that iCache can provide 20–50× speedup. Note that the y-axis is on log scale.

custom request-level simulator called iCache that provides almost 50× improvement over state-of-the-art ICN simulation platforms. In designing iCache, we make three key decisions:

- ndnSIM [5] does a detailed packet-level simulation carefully capturing every content packet and content response. This leads to a significant number of events in the system and very high memory consumption at high request rates. As a first step, we simplify the simulation by moving to a request-level rather than a packet-level simulation. One concern is that we may not be able to replicate fine-grained QoE metrics with a flow-level simulation. Fortunately, we use a data-driven extrapolation methodology for computing QoE metrics that is amenable to a flow-level analysis. Second, ndnSIM provides a complete implementation of all ICN functionalities; e.g., checks for content-level security. These features significantly add to the per-request processing but are not relevant to our study, . We simplify the per-node operations and do not implement the content-integrity steps.

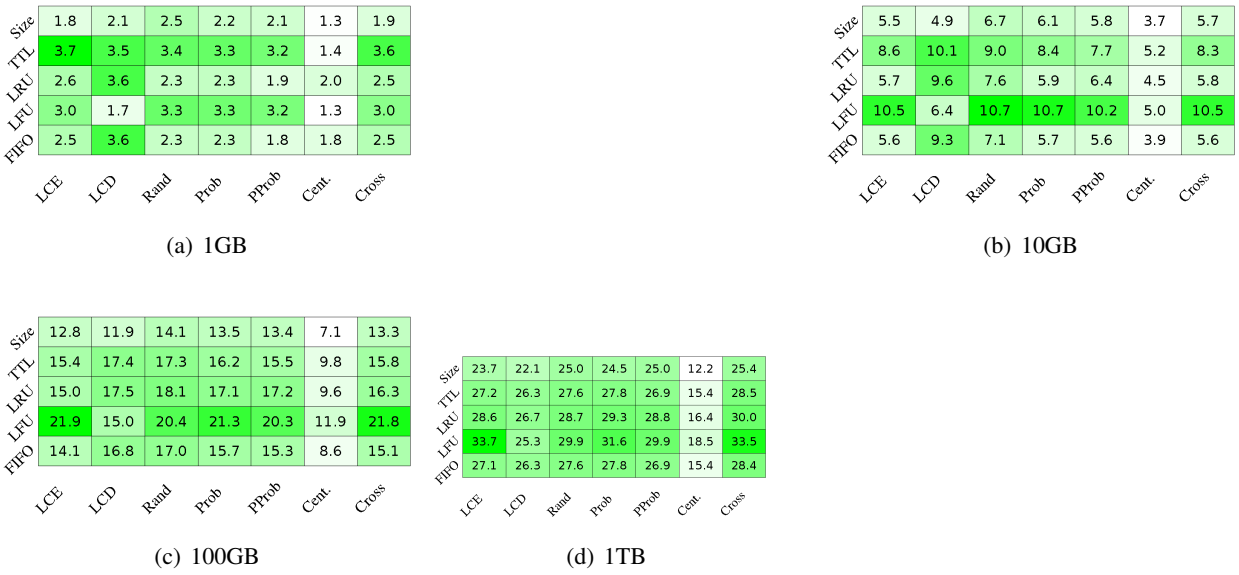
- To emulate complex content routing protocols such as OSPF extensions, ndnSIM effectively recomputes each routing table on each cache eviction event. This creates a large number of routing protocol messages and increases the processing overhead. However, for many common ICN routing strategies (e.g., shortest path to servers [112, 45]) the routing table does not need to be updated. We, therefore, precompute the routing tables and use them throughout the simulation.

- Finally, rather than a node-centric software architecture, we move to a "tier"-based architecture where we group modules with the same function on different nodes into a single layer, such as a caching layer, forwarding layer, and routing layer. Unlike other simulators, where each simulated ICN node maintains its own routing, caching, and forwarding tables, this grouping enables the use of global tables that are shared by all routers avoiding redundant entries. This not only reduces memory consumption, but it also improves the locality of memory accesses, thus minimizing I/O and improving the processing time.

Figure 14 shows the simulation time of a large-scale simulation on a machine with a Xeon 2.13GHz core and 1TB of memory running Linux. The simulation involved about 80K routers, 16M clients, 500K videos, and 196M content requests, and took $\leq$5 hours for different cache sizes. For comparison, the native ndnSIM needs 127 to 252 hours to finish the same simulations. As a point of comparison, we also include a version of ndnSIM with the routing pre-computation optimization described above. This precomputation does help significantly. It cuts the simulation time to 7.5-132.9 hours, but this is still almost 2× slower than iCache.

Using iCache, we answer the two high-level questions raised above in the context of a popular video content provider (PPTV). Our key observations are as follows:

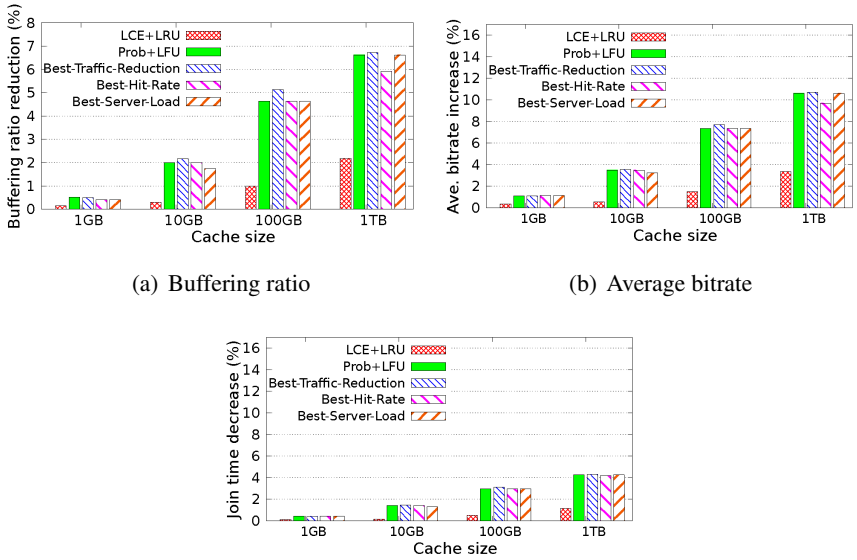
- For a provider like PPTV which already has a substantial server footprint, even with the largest cache size in our evaluations (1TB), the overall video QoE improvement is around 12% (Figure 16).



(a) 1GB (b) 10GB

(c) 100GB (d) 1TB

Figure 15: Total traffic reduction (%).



(a) Buffering ratio (b) Average bitrate

(c) Join time

Figure 16: Improvement in different QoE metrics for different ICN caching strategies and cache sizes.

- Across all strategies, we observe that only very large caches (i.e., 100GB or 1TB) provide considerable ($\geq$ 10%) reduction in the network traffic (Figure 15). The reduction is more than 20% only with cache $>$ 100GB. The best combination of placement and replacement for 1GB, 10GB, 100GB, and 1TB are *LCE +TTL*, *Prob +LFU*, *LCE +LFU*, and *LCE +LFU*, respectively. [1]
- While the best combination of content placement and replacement strategies does depend on the cache

[1] Placement strategies: *LCE* leaves a copy everywhere, and *Prob* is a probabilistic stategy. Replacement strategies: *TTL* uses a popularity based TTL and *LFU* is least frequently used.

Figure 17: Role of XIA concepts in addressing network challenges

size and the metric of interest, the combination of probabilistic content placement (*Prob*) and LFU content replacement (*LFU*) emerges as a near-optimal strategy across all scenarios.

- We also analyze *where*, *when*, and *what* contributes to the improvement and find that: (a) caches in the middle and access portions of the network contribute most to traffic reduction and QoE improvement; (b) requests from highly populated regions without content servers observe the highest QoE improvement; and (c) requests for popular content are more likely to contribute to overall ICN-induced QoE improvements.

2.7 Initial evaluation of the XIA data plane

We looked at how effective the concepts forming the basis of the XIA data plane have been in addressing various network challenges, based on our experience so far.

Background - One way of evaluating network architectures is to evaluate how much it contributes to solutions to deal with specific network challenges. Network architectures can be specified at multiple levels: principles and invariants (e.g., the concepts in Figure 3), concrete specification (e.g., an IETF draft), and implementation (e.g., the XIA prototype). While a proposed solution and an evaluation of its effectiveness will naturally use a specific specification and implementation of the architecture, it is possible to use this approach to evaluate an architecture at the principle level by arguing that the benefits of solution deive from principles, and not a from a particular implementation.

Approach and Findings - The first column in Figure 17 shows a number of challenges that we have addressed using XIA. Details on the solutions can be found in the annual reports for the XIA project funded by the FIA program. The three last columns use a check mark to show whether the availability of multiple principal types, flexible addressing and intrinsic security played a key role in the proposed solution.

All solutions relied on being able to use multiple principal types. They are obviously required for evolvability and network diversity. Scoping and mobility require network identifiers, and incremental deployment of XIA relies on a new "4ID" to represent legacy network addresses, e.g., IPv4. The evolvability study that ported "foreign" FIA concepts to XIA introduced a variety of new XID types and the introduction Scion path-based forwarding requires Scion IDs. A recently started effort on publish-subscribe systems has introduced "RIDs" representing subscribe requests.

All solutions similarly require flexible addressing. Most solutions require fallbacks: evolvability, net-

work diversity, incremental deployment of XIA, pub-sub, and mobility. Scoping and Scion require "chaining" of XIDs, i.e., scoping. Finally, the extreme evolvability study uses a variety of DAG formats.

Finally, mobility and Scion require intrinsic security to guarantee integrity of the end-end communication session (large check mark). Evolvability and extreme evolvability gain benefit from security (small check mark): evolvability benefits from having principal specific intrinsic security properties, while the XIA version of "foreign" FIA concepts was sometimes able to enhance them by adding intrinsic security properties. Finally, scoping and network diversity are agnostic to the presence intrinsic security, i.e., they maintain the properties of individual XIDs, but do not extend them. "4IDs" help with the transfer of packets over legacy networks, which lack intrinsic security. Finally, the pub-sub research is still ongoing but is likely to use intrinsic security to link "RIDs" with responses.

3 Building on the XIA Architecture

The functionality and performance of a network is not solely determined by its architecture but critically depends on other network protocols and services. A number of XIA research activities have looked at how other network functions can contribute to the goals of XIA, e.g., security, evolvability, etc.

3.1 Transport over Heterogeneous Networks

Tapa is part of Fahad Dogar's PhD thesis [34], advised by PI Steenkiste.

3.1.1 Background and Goals

The original Internet was based on the "intelligent endpoints, dumb network" principle, so most of the data transfer functionality was implemented at end-points, as part of the transport protocol (i.e., TCP). However, the demands of today's Internet require revisiting this design approach. Specifically, two trends are increasingly challenging this end-point-based approach to transport protocol design.

Heterogeneous Networks and Devices: Unlike the original Internet, the networks that make up the Internet now are very diverse, ranging from very high speed optical backbones, to low speed, unreliable wireless networks, causing problems for end-to-end protocols such as TCP, e.g. [13, 39]. Similarly, devices, such as sensors and mobile nodes, are highly resource-constrained, compared with traditional endpoints. This heterogeneity affects how we should distribute functions across devices. For example, when applying the end-to-end argument [103], we can no longer view the communication subsystem as a single homogeneous module.

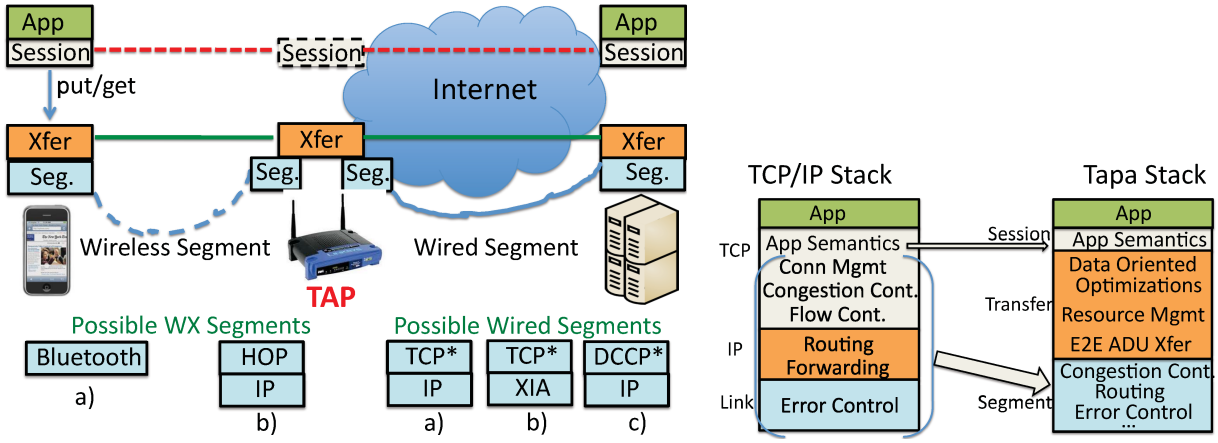
Rich Network Services: Today's network needs to provide a wide variety of services, including data oriented services, such as application independent caching and retrieving content from multiple sources, and higher level services, such as transcoding proxies and virus scanners. Unfortunately, inserting services in an end-to- end path is hard, as the data units used by applications and their naming are not exposed to the transport as well as element within the network. Moreover, TCP's end-to-end semantics are very rigid: they do not allow role of an intermediary or accommodate different delivery semantics.

So far, our response to the above trends has been to implement ad-hoc solutions, such as splitting transport connections at APs, transparent proxies, CDNs, and various other forms of application middleboxes. Unfortunately, these solutions are fragile and complex to manage, and they often interact poorly with other protocols [13, 18, 47, 26].

3.1.2 Approach and Findings

The goal of Tapa is to provide an architecture that can accommodate heterogeneous networks and rich in-network services in a systematic manner. The initial concepts underlying Tapa are described in [35]. The basic idea (Figure 18(a)) is to partition end-to-end paths into *segments*, homogeneous sections of the path that are connected using Transfer Access Points (TAPs). Segments might for example cross a wireless access network, or part of the wired Internet backbone. End-to-end connectivity is provided by the transfer layer, which forwards data across segments in a best-effort fashion, similar to how IP forwards packets across routers. Finally, there is a session layer that implements specific end-to-end semantics over the path. Another way of looking at Tapa is that it unbundles the thick IP transport layer into a set of thinner protocols that focus on addressing separate concerns (Figure 18(b)).

This architecture has several benefits. A first benefit of using segments is that they can use segment-specific mechanism to deal transport functions such as error, congestion, and flow control. Wired segments can for example use end-to-end solutions, while wireless segments can use native solutions. Segments also offer a nice abstraction for managing topology, e.g. handoff for mobile users. Second, TAPs provide



(a) Tapa overview (b) Tapa protocol stack

Figure 18: Tapa Overview and Protocol Stack

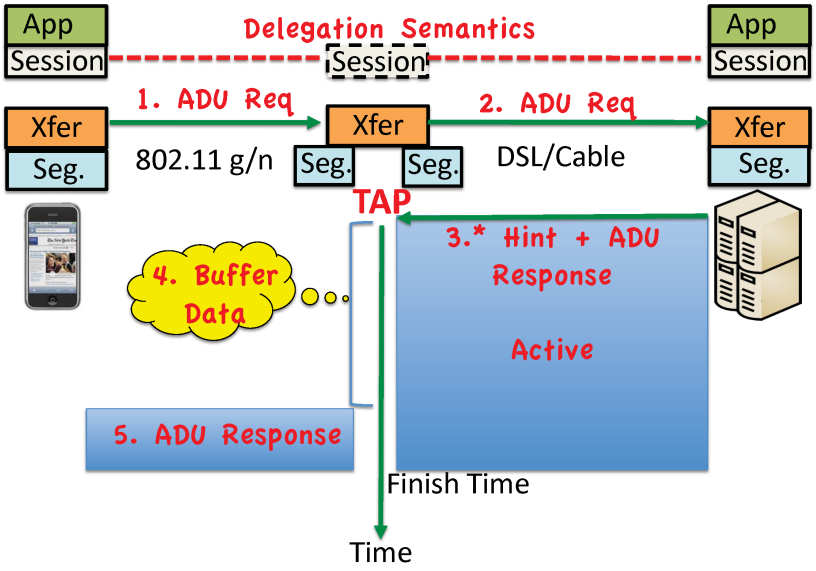


Figure 19: Catnap Overview

a natural insertion point for services. The original Tapa effort focused on content centric service, so the transfer layer operates on ADUs (or chunks using XIA terminology) and the service offered by segments is best-effort ADU delivery. Finally, since the session layer is relatively light-weight since it only operates over a short multi-segment path, implementing diverse semantics should be easier than in today.s Internet.

We applied the Tapa concepts to design a transport architecture for heterogeneous networks that can leverage the novel XIA features. This led to a number of changes to Tapa that improve its flexibility and allow it to be used over diverse network architectures, such as IP and XIP. The starting point for this effort was the question of how Tapa can leverage the content and service principals supported by XIA. This turns out to be somewhat tricky since the original Tapa provided content and service support in an overlay over IP, while XIA supports for these functions sits much lower in the network - below the narrow waist rather than above it. The solution was to view the transfer function as a service that is simply instantiated differently, depending on the network architecture. The transfer service can semantically sit at multiple levels, ranging from simply forwarding ADUs, to content optimization (e.g. caching, which can supported natively in XIA,

but uses a DOT-style overlay on IP), and higher-level services such as firewalls or transcoding.

A second effort was to explore a broader set of services, which the goal to evaluate the breadth of the Tapa design. While the initial focus was on fairly simple transfer level services, e.g. support for mobility [35], we now also considered higher services, for example, distributed caching in the context of a social networking service, data processing, and security related services. The Tapa architecture turned out to be effective for all these examples. We also found that the concept of segments of different types is more general than we originally envisioned. In Catnap [36], we used the concept of a "bursting" segment to improve the battery life of mobile devices. Existing transport protocols such as TCP mostly deliver "ASAP" service. This is a problem for mobile devices that using technologies such as 802.11 connected to slower wired access networks since the slower wired connnection effectively paces packets over the wireless link, preventing it from going into power save mode. Catnap is a simple service that batches data on the access point so it can be sent efficiently as a single burst over the wireless link, as is illustrated in Figure 19. In the context of social networking applications, we generalized this idea by introducing "slow" segments that forward data to a central server as a background task. Clearly, the notion of a segment that can change the timing of data forwarding is a generally applicable concept.

3.2 Evolvable transport protocols

We explored how we can make transport protocols evolvable. This research was done by Robert Grandl and PI Aditya Akella from the University of Wisconsin, in collaboration with Dongsu Han and PI Srini Seshan from Carnegie Mellon University.

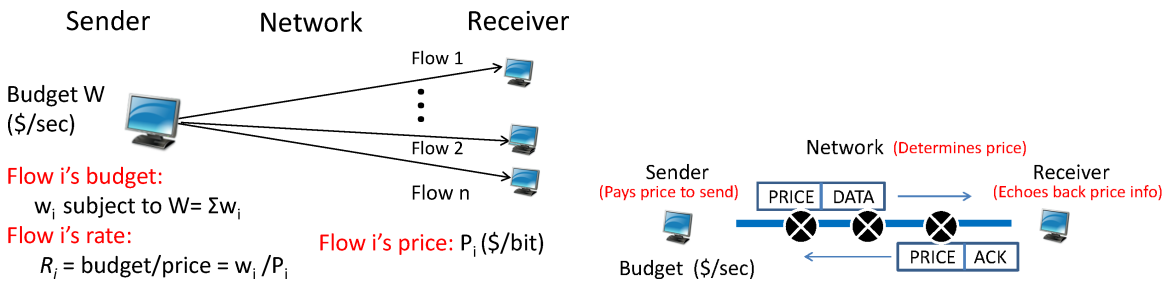
3.2.1 Background

Networked applications often require a wide range of communication features, such as reliability, flow control, and in-order delivery, in order to operate effectively. Since the Internet provides only a very simple, best-effort, datagram-based communication interface, we have relied on transport protocols to play the key role to implementing the application's desired functionality on top of the simple Internet packet service. As application needs and workloads have changed over time, transport protocols have evolved to meet their needs. In general, changes to transport protocols, such as adding better loss recovery mechanisms, have been relatively simple since they require only changes to the endpoints. However, among the functions that transport protocols implement, congestion control is unique since it concerns resource allocation, which requires coordination among all participants using the resource. As a result, two very different styles of congestion control cannot coexist, making evolution of congestion control difficult.

The need for evolution in congestion control becomes apparent when we look at the history of TCP. While TCP's congestion control was not designed with evolution in mind, the development of TCP-Friendliness principles [46] enabled the development of a wide range of congestion control techniques to meet different application requirements; these include support for: streaming applications that require bandwidth guarantees [46, 25, 17, 88], or low-latency recovery [77, 50], non-interactive applications that can leverage low-priority, background transfer [114], applications that require multi-path communication for robustness [118], and Bittorrent-like content transfers that involve in transferring chunks from multiple sources.

The ability for TCP's congestion control to evolve has been enabled by two key aspects of its design: 1) Purely end-point based nature allowed new algorithms to be easily deployed and 2) its AIMD-based congestion avoidance led to the notion of TCP-friendliness.

Unfortunately, recent efforts, such as RCP [38] and XCP [65], rely on explicit congestion feedback from the network. While these designs are far more efficient than TCP, they limit the range of different end-point application behaviors that the network can support. It is not understood whether such explicit



(a) Design overview (b) Price feedback generated by the network

Figure 20: Overview of FCP, XIA's evolvable transport protocol

congestion control algorithms can be made flexible to allow evolution. The goal of our work is to design a novel congestion control framework that is as efficient as explicit congestion control algorithms (e.g., RCP and XCP), but retains (or even expands) the flexibility of TCP-friendliness based solutions.

3.2.2 Approach and Findings

We have developed a new congestion control architecture called FCP (Flexible Control Protocol) that leverages explicit network feedback but still supports diverse endpoint and network behaviors. We present a brief overview of FCP in Figure 20. FCP leverages ideas from economics-based congestion control [67, 68] and explicit congestion control. In particular, to enable flexible resource allocation with in a host, we allow each domain to allocate resources (budget) to a host (Figure 20(a)), and make networks *explicitly* signal the congestion price (Figure 20(b)). The flexibility comes from the end-points being able to assign their own resources to their flows (an example is shown in (Figure 20(a))) and the networks being able to aggregate flows and assign differential price to different classes of flows to provide extra functionality. The system maintains a key invariant that the amount of traffic a sender can generate per unit time is limited by its budget and the congestion price (Figure 20(a)). Co-existence of different styles and strategies of rate control is ensured simply by maintaining this key invariant, allowing evolution.

To make economics-based congestion control [67, 68, 69] practical, our work extends past theoretical work in three critical ways:

- FCP uses "pricing" as a *pure abstraction* and the key form of *explicit feedback*. In contrast, others [67, 20, 69, 29] directly associate congestion control to the real-world pricing and modify existing congestion control algorithms to support weighted proportional fairness.
- We address practical system design and resource management issues, such as dealing with relative pricing, and show that FCP is efficient.
- Finally, unlike previous explicit rate feedback designs [38, 68], FCP accommodates high variability and rapid shifts in workload, which is critical for flexibility and performance. This property of FCP comes from a novel *preloading* feature that allows senders to commit the amount of resource it wants to spend ahead of time.

A related research effort on transport protocols within XIA developed the RPT protocol for loss-protection in content-aware networks [50]. The key contribution is the concept of Redundant Transmission where data is sent is multiple times. This is done in such a way the reduncy is exposed to the content-aware network, so it can optimize the transmission. RPT was developed within the context of the XIA project but since it was funded through a separate NSF award, we refer to the paper for details.

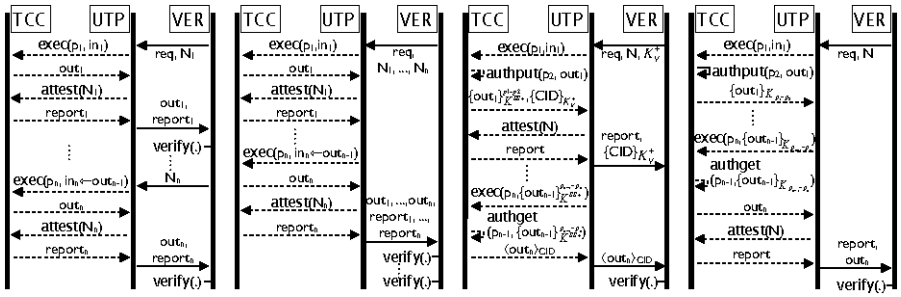


Figure 21: Protocol executions: a) naive; b) message efficient; c) verification efficient; d) skSS. Dashed arrows corresponds to (usually local) interactions between the third-party provider and the TCC, while full arrows correspond to messages exchanged between the verifier and the third party over the network. In (b), each of intermediate results out1, . . . , outn1, sent by the UTP to the verifier, is a measurement of the corresponding PALs output.

3.3 Secure Third-Party Services

The XIA team explored protocols and mechanisms for enabling trusted third party computations. This research was conducted by Bruno Vavala, Nuno Neves, and Peter Steenkiste [113].

Background - XIA's intrinsic security, in which a hash of data serves as its identifier, enables trusted retrieval of static content, even from untrusted sources. However, it is unclear how to enable trust for the results of outsourced computations. This is especially an issue for computations involving a third party, in which some trusted service provider re-directs the client to an external entity, who then performs the computation. While clients can use the intrinsic security associated with SIDs to make sure they talk to the right service, this does not guarantee that the right computation is performed since the service may be compromised or may not be trusted by the client.

Previous work on verifying outsourced computations does not scale, as most treat the entire computation as one monolithic unit. It is sometimes also not applicable to general purpose computational tasks. In this work, we aim to identify a protocol for verifying outsourced general computations with runtime independent of the computation size.

Approach & Findings - Our approach leverages existing work on trusted platform modules (TPMs). Namely, we assume the third-party provider exposes a trusted computing component (TCC), consisting of the CPU, the TPM, the LPC bus, the memory available to the program, the communication bus with the memory. We assume adversaries can take control of the third party and can use the TCC, but cannot compromise it. To enable efficient verification of computation results, we split computations into multiple individual pieces, called PALs, via existing program partitioning techniques. We then describe a tamper-evident and efficient chain of trust among the PALs that makes per-PAL verification overhead on the client negligible.

Efficiency is achieved by carefully considering the different contributors to overall verification cost and devising optimizations for them. For example, compared to a naive scheme that verifies each PAL independently, interactions and communication load can be avoided by combining data hashing and message batching. The first reduces the amount of information delivered to the verifier for a single PAL execution. The output of an intermediate PAL can be replaced with its (typically) smaller measurement (i.e., hash). The second decreases the interactions between the verifier and the UTP, by piling the verification information into the final message. Figure 21 shows the naive approach, successive modifications made to improve its efficiency, and the end result (denoted skSS).

Our evaluation based on a prototype implementation uses a service we built that applies filters to transform input images. Our results shows that our final protocol is efficient. Specifically, we find that the skSS

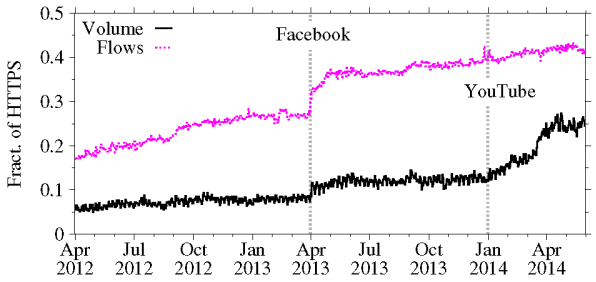


Figure 22: Evolution of HTTPS volume and flows share over 3 years in a residential ISP. Vertical lines pinpoint the transition towards HTTPS for Facebook and YouTube.

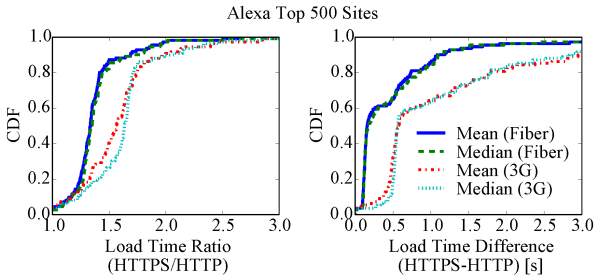


Figure 23: Webpage load time inflation for the top 500 Alexa sites.

protocol adds only 12% overhead compared to normal runtime.

3.4 Exploring the Hidden Costs of HTTPS

One of XIA's fundamental goals is a more secure Internet. HHTPS is a common security technique used in today's Internet so we study it to better understand tradeoffs in securing data transfers. This work is being done by David Naylor and Peter Steenkiste at CMU in collaboration with Telefonica Research.

Background - HTTP is on the verge of a new milestone: the standardization of HTTP 2.0 [59] is slated for the end of 2014. In some discussions, TLS is considered to be the default transport layer, mirroring a fundamental design decision of SPDY [111], which was used as a starting point for HTTP 2.0. Given users' growing concerns about security and privacy on the Internet, adopting encryption by default in all HTTP communication sounds like a good idea. However, security always comes with costs; in this paper, we aim to categorize and quantify the cost of "S" in HTTPS.

Using three different datasets, captured in residential and cellular networks, as well as controlled experiments, we evaluate the cost of HTTPS in terms of (i) infrastructure/management cost, (ii) load time, (iii) data usage, (iv) battery life, and (v) loss of value-added in-network services.

Initial Results - The analysis of the dataset led to the following observations:

1. HTTPS accounts for 50% of all HTTP connections and is no longer used solely for small objects (Figure 22. The stunning increase in adoption suggests the cost of deployment for services is manageable.

2. Most users are unlikely to notice significant jumps in data usage due to loss of compression (about 2MB/day), but ISPs stand to see a large increase in upstream traffic due to loss of caching (about 10TB/day for the cellular provider we studied).

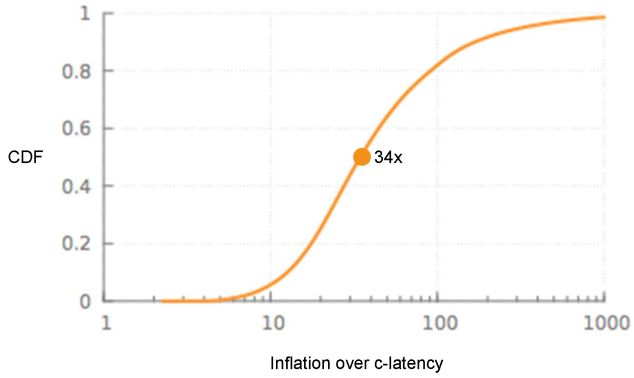


Figure 24: CDF for inflation over speed-of-light.

3. The extra latency introduced by HTTPS is not negligible, especially on 3G, where the extra latency is larger than 500ms for about 90% of websites and 25% of pages suffer an increase of at least 1.2 s (i.e., a inflation factor larger than 1.5x). On fiber, the extra latency is smaller; still, for 40%, HTTPS adds more than 500ms (1.3x). (Figure 23.)

4. HTTP proxies can both help and harm users' battery life. In one experiment we saw a proxy increase energy consumption by nearly 50% in some cases and decrease energy consumption by roughly 30% in others.

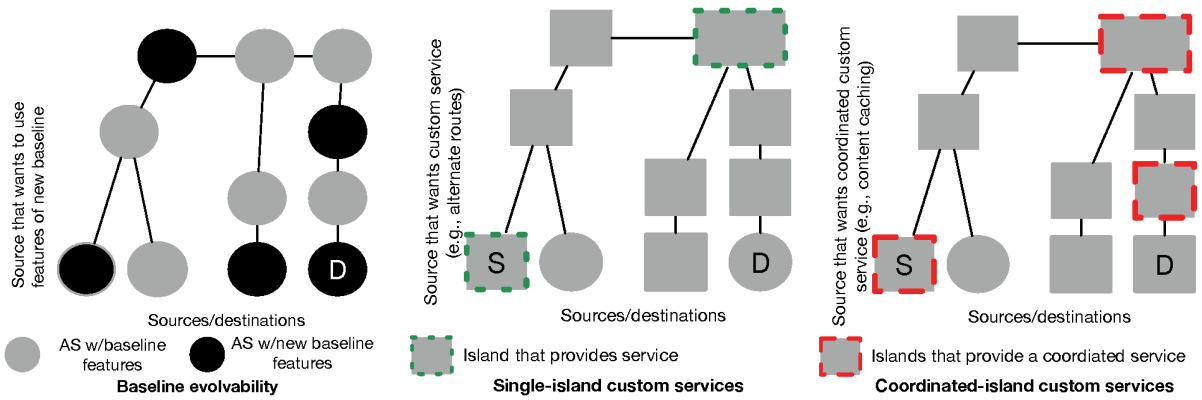
5. Though difficult to quantify, the loss of in-network services is potentially substantial; some of that functionality could be equally well performed on the client, while others may require a total rethink, like non-stealth DDoS-detection.

3.5 The Internet at the Speed of Light

PI Maggs has been investigating how a future Internet Architecture could provide intrinsic support for low-latency networking. In particular, Maggs has argued that while improvements in network performance have traditionally focused on increasing bandwidth while latency has been neglected, many interactive applications would be enabled if the Internet could transport data from point to point at close to the speed of light.

Approach - An initial study conducted by Maggs and his collaborators [106] determined that today's Internet operates much more slowly than the speed of light. For example, Figure 24 shows the inflation over speed-of-light latency when downloading a random sample of the Alexa 500 most popular Web sites from a variety of PlanetLab nodes. The median inflation factor is 34. The study found that a factor of about 4 in the slowdown can be attributed to the physical infrastructure (e.g., long fiber paths), while a factor of about 9 can be attributed to inefficiencies in network protocols (e.g., gratuitous round trips).

Findings - This work suggests several requirements for future Internet architectures. First, eliminating round-trips from existing network protocols such as TCP and TLS is essential to reducing latency. Because "handshakes" serve in part to deter address spoofing, e.g., in protocols like TCP, alternate mechanisms will have to be put in place to authenticate hosts. The latency incurred by DNS may be reduced by deploying DNS resolvers much more widely and perhaps by introducing new protocols for sharing cached results, by combining DNS with other protocols such as TCP, or perhaps by introducing Information-Centric Networking-like primitives, such as CIDs, can provide direct support for routing requests to content or services. Second, protocols may have to be developed that ensure that packets follow the shortest-latency



(a) Adding functionality to baseline (b) Single-island custom services (c) Coordinated-island custom services

Figure 25: Evolvability scenarios for routing protocols.

paths. To carry the small, but valuable, fraction of network traffic that requires minimum latency, it may be necessary to deploy a parallel low-latency-but-low-bandwidth physical infrastructure.

3.6 Supporting evolvable routing protocols

We explored mechanisms for enabling evolvability in the Internet's control plane. This work is being performed by David Tran-Lam and Aditya Akella (Wisconsin), and Raja Sambasivan and Peter Steenkiste (CMU).

Background - The ability to route traffic across Network domains is one of the fundamental building blocks of the Internet. As such, a significant impediment to the Internet's evolvability is the extremely slow rate at which newly introduced routing features or capabilities can be utilized. This slow pace has for example depressed significant opportunities [120, 95] for custom, value-added routing services. Difficulty of utilizing new features has also depressed attempts to change the Internet's baseline protocol for connectivity, BGP, despite known significant flaws (i.e., lack of security or conflating connectivity with traffic engineering).

The difficulty of using new features results from the fact that, typically, all domains involved in routing packets along a path must support or understand a new feature before it can be used. For example, a domain might introduce a new feature called source-driven path selection that exposes all of its path options to a destination to sources. However, this feature cannot be used by sources unless intermediary domains also expose these alternate paths or an explicit application-level protocol is used to communicate this information. One of the goals of this work is to create architectural support for exposing new features to sources, even when intermediary domains do not understand or have support for them.

Evolvability scenarios - To help focus this work, we have identified a set of *evolvability scenarios*, corresponding to the key reasons why ASes may wish to add new features. Each scenario entails different challenges and lends itself to different approaches. Figure 25 illustrates the different scenarios.

Scenario 1: Adding new functionality to a baseline protocol: Entities involved in routing have a vested interest in periodically updating BGP with new features. The expectation is that all domains will eventually use these features. Past examples include migrating from BGPv3 to BGPv4, which involved adding classless advertisements, and current efforts to migrate to S-BGP [70], which involves adding authenticated routing advertisements. Though we cannot predict the future, additional updates to BGP could include QoS metrics [84], inter-domain extensions to MPLS [16], and richer interfaces to support content.

Scenario 2: Single domains that wish to expose custom services: Transit providers or other domains

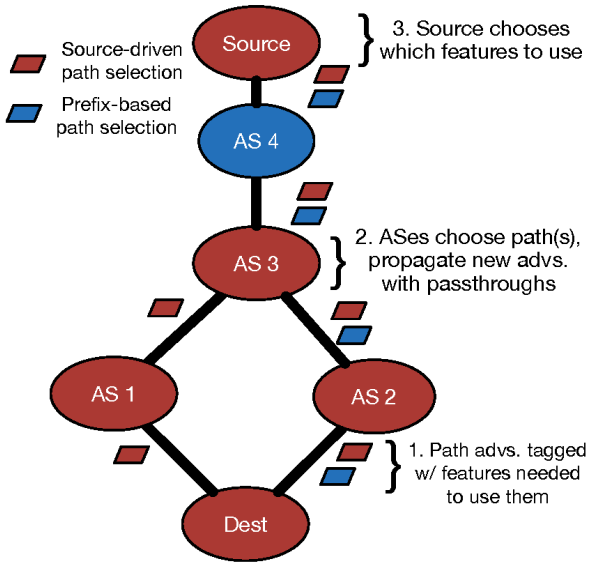


Figure 26: How the common-language approach enables use of new functionality.

may wish to provide custom value-added services to sources, for which they can charge extra. Examples may include "alternate paths as a service" [120, 95] (i.e., allowing sources to choose alternate paths to a destination) or "denial of service resiliency as a service" (i.e., allowing destinations to turn off or limit traffic destined to them).

Scenario 3: Coordinated domains that wish to expose custom Services: This scenario is similar to the single AS case described above, except that a set of domains may wish to coordinate to expose a new service, requiring that traffic be routed among them. One example may be a content-caching service that relies on content requests to be routed among paths that include many content-caching domains.

We decided to initially focus on the first scenario.

Results for Scenario 1 - We are initially focusing on approaches for evolving the Internet's baseline protocol (Scenario 1). Since the goal is to add new features that all domains will eventually use, this scenario is best served by an *in-band* approach, in which the mechanism used to transmit the baseline protocol itself provides support for evolvability. In contrast, only a fraction of domains may be involved in a given custom service, so Scenarios 2 and 3 may be best served by *out-of-band* approaches that communicate new features via an independent protocol.

Our in-band approach uses three key elements that combined allow domains to "pass-through" functionality they do not support. The first element is a *common language* that allows domains to describe what capabilities or features they expect other upstream domains should support when using a given route. The same route can be tagged with multiple capabilities, corresponding to acceptable alternates if the original desired capability is not supported. The second element is a *common language import/export* module implemented within an AS's centralized controller, which decides which routes the AS can use. A domain must choose routes tagged with a supported capability, but can pass-through any additional, non-supported capability associated with that route. The third element is *common language data-plane support* which is used to specify how to associate packets to be forwarded with a specific route (e.g., forwarding entry).

Figure 26 provides an example of how these elements combine to enable source-driven path selection. The figure shows a sample topology in which red (dark grey) domains have a new feature that allows them to expose all path options to neighbors from their border routers, allowing source-driven path selection. Blue (light grey) nodes do not support this feature and can only expose one path per border router. With

our approach, domains use a common language to advertise paths (Step 1). This language allows them to express which features must be used with which paths. It also allows them to express alternate features that can be used. The set of features that can be used may depend on what support the router advertising that path has. In this case, the destination domain has two border routers and the rightmost one supports both source-driven path selection and prefix-based-path selection. The leftmost one only supports source-driven-path selection.

Common language advertisements are interpreted by each domain's SDN-based routing application (step 2). Controllers can only advertise path(s) that are tagged with the functionality they support and want to use, but can "pass-through" features associated with this path even if they do not support them. This pass-through functionality allows domain 4, which does not support source-driven path selection, to propagate that feature, enabling the source to use it. Finally, the source selects which features it wants to use (step 3). This selection must be enforced by the data-plane during packet forwarding. Note that the feature is enabled despite the existence of an intermediate AS that does not support this feature.

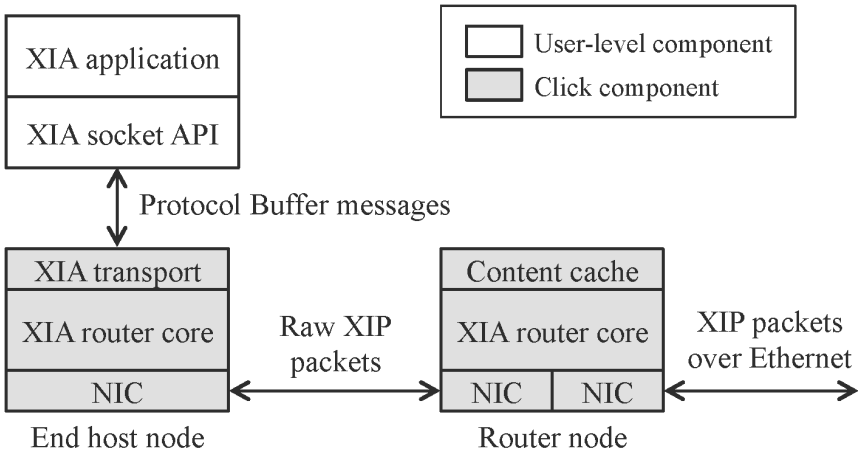


Figure 27: XIA forwarding engine prototype

4 Building an XIA Network

Implementing networks based on the XIA architecture involves a number of challenges. These include high speed packet forwarding based on flat identifiers and very large forwarding tables, implementing XIA in a production OS, and making XIA's functionality available to applications and users.

5 XIA Prototype: Design and Evaluation

The XIA prototype was developed with significant contributions from David Naylor, Matt Mukerjee, Suk-Bok Lee, and Dan Barrett.

5.1 Background and Goals

The development of a system as complex as a network must consider many factors, including implementation complexity and cost. For this reason, building a prototype XIA network is an important part of the project. The development of the XIA prototype network has the following goals:

- Evaluate the performance of key components of the XIA architecture. Some initial results are presented below.
- Provide a foundation for prototyping and evaluating networking research that builds on XIA, e.g. XIA-based transport protocols, mobility services, content optimizations, etc.
- Provide a platform for application development that will help define appropriate interfaces for using XIA based networks and for evaluating the value of XIA to end-users.
- Provide a platform that can be used for course projects at the graduate and undergraduate level.
- Form of the basis for collaboration with other research groups.

We developed a click-based prototype of the XIA forwarding engine and used it in evaluating the performance of packet forwarding functions using the packetshader platform.

5.2 XIA fowarding engine

We first prototyped the XIA forwarding engine using the Click modular router. Our prototype of the XIA forwarding engine consists of 4400 SLOC of C/C++ excluding the original Click code. A high-level design of the prototype is shown in Figure 27.

The prototyped XIA forwarding engine performs all the functions identified in Figure 2; it performs forwarding and per-hop processing for ADs, HIDs, CIDs, and SIDs. DAG processing is the core part of the forwarding engine. We implemented the DAG processing logic in the Click configuration file, and each element performs simple task such as looking up an XID in the table, classifying the principal type, and choosing fallback XIDs. It also includes support for content caching, i.e. can store chunks and serve chunks in response to XIA packets that use a CID as their intent destination (final node in the DAG).

The prototype uses the Click network interface, so it can run both as an overlay over IP, or it can run natively over Ethernet. XIA-enabled end hosts can invoke Click-based XIA network service by making an RPC, which uses TCP to connect to the XIA router, as is shown in the figure in the top left corner. We have also implemented a simple end-to-end web browsing scenario on top of XIA routers.

5.3 Performance of XIP Packet Forwarding

We believe that XIA forwarding can benefit considerably from recent trends in emerging programmable software routers. The programmability would help to quickly enhance routers with support for new principal types, or replace principal-specific forwarding algorithms with better algorithms. The recently developed Routebricks platform is a software router architecture using multicore systems that can forward IPv4 packets at the rate of 35 Gbps and is fully programmable. PacketShader uses GPUs for forwarding 64B IPv4 packets at 39 Gbps on a single commodity PC. However, there are fundamental differences between XIA packet forwarding and simple IPv4 forwarding. The per-hop processing of XIA is more complex than that of IP, which raises obvious concerns about the performance of routers, especially in scenarios using more complex DAGs for addressing. Despite those concerns, our evaluation of XIP packet forwarding, summarized below, shows that an XIP router can achieve comparable performance to IP routers and that well-known optimizations, such as parallel processing and fast-path evaluation, also apply to XIP.

To measure the packet processing speed, we set up a software router and a packet generator to exploit packet-level parallelism by leveraging their NIC's receiver-side-scaling (RSS) function to distribute IP and XIP packets to multiple CPU cores. More details on the set up can be found in [51]. We use the Click-based forwarding engine implementation described above, and PacketShader's I/O Engine (NIC driver and library) [54] for sending and receiving packets to and from the NICs. We used a forwarding table of 351K AD entries based on a Route Views RIB snapshot of Jan 1, 2011.

Figure 28 shows the throughput in Gbps as a function of packet size for multiple forwarding options. The FBx curves correspond to the case where x fallbacks had to be considered to forward the packet, e.g. FB0 corresponds to the basecase that only the intent identifier had to be considered. The VIA curve represents the case where an intermediate node in the DAG has been reached, so the router must also update the last-visited node field in the packet header.

For large packets, forwarding is limited by I/O bandwidth, and therefore XIA and IP show little performance difference. For small packets of 192 bytes, XIA.s FB0 performance (in pps) is only 9% lower than IP. As more fallbacks are evaluated, performance degrades further but is still comparable to that of IPv4. For example, FB3 is 26% slower than IP). XIP's goodput is lower than IPv4's [51] due to XIA's large header size. In cases where the goodput is important, header compression techniques may be an effective solution. Also, by processing multiple packets concurrently, *parallel packet processing* can speed up XIP forwarding. In XIA, AD and HID packet processing resembles IP processing in terms of data dependencies; the forwarding path contains no per-packet state changes at all. In addition, the AD and HID lookup tables are the only

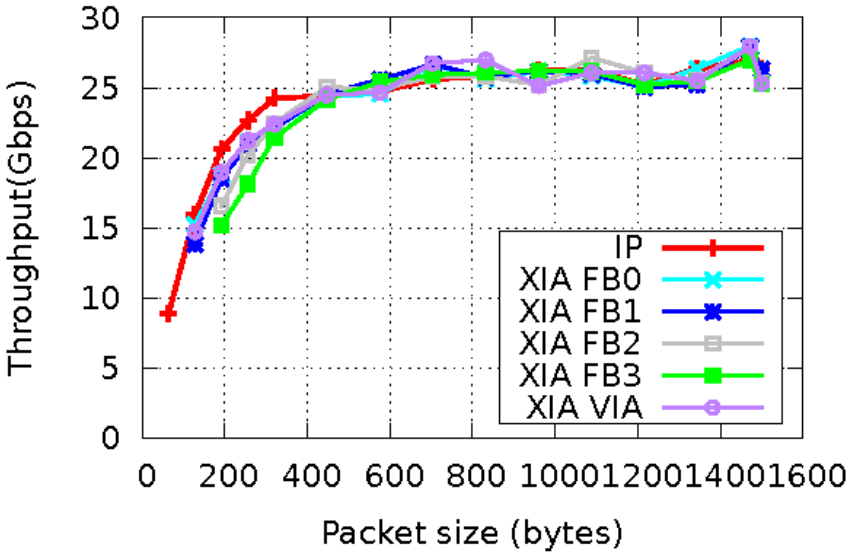
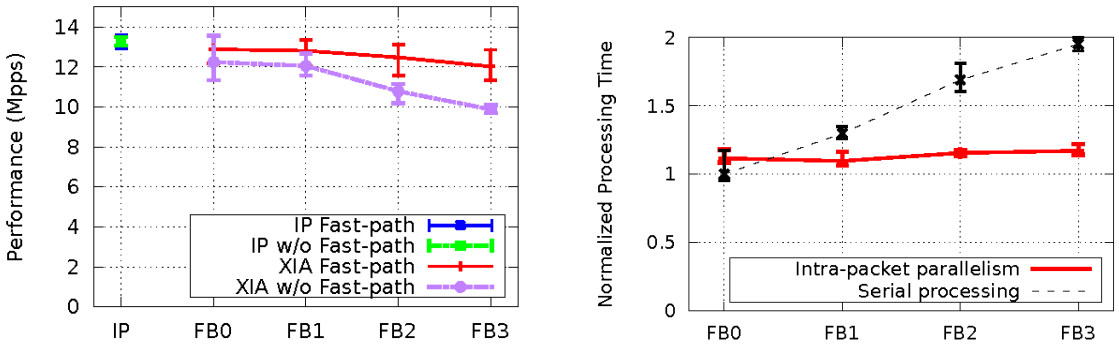


Figure 28: XIA forwarding throughput in Mbps



(a) Fast path processing

(b) Intra-packet parallelism

Figure 29: Performanc with fast path processing and intra-packet parallelism

shared, global data structure, and like IP forwarding tables, their update rate is relatively infrequent (once a second or so). This makes it relatively straightforward to process packets destined for ADs and HIDs in parallel. CID and SID packet processing can similarly be parallelized.

XIP design can also use *fast-path* processing to speed commonly observed addresses—either as an optimization to reduce average power consumption, or to construct low cost routers that do not require the robust, worst-case performance of backbone routers. In our prototype we added a per-thread look-aside cache that keeps a collision-resistant fingerprint of the destination DAG address and the forwarding result (the output port and the new last-node value). When a packet's destination address matches an entry in the cache, the router simply takes the cached result and skips all other destination lookup processing. Otherwise, the router pushes the packet into the original slow-path processing path. Since the processing cost of this fast path does not depend on the address used, this performance optimization may also help conceal the impact of packet processing time variability caused by differences in DAG complexity.

Figure 29(a) shows the result with and without this fast-path processing for packet size of 192 bytes and a per-thread cache with 1024 entries. Without the fast-path, the performance degrades by 19% from FB0 to FB3. However, with fast-path this difference is only 7%. With a marginal performance gain of IP fast-path,

the gap between FB3 and IP fast-path is reduced to 10%.

Since the fast-path optimizations do not improve worst-case performance (which is often critical for high-speed routers that must forward at line speed), we used micro-benchmarks to estimate the performance that might be possible if using specialized hardware for XIP. Since route lookup time, which dominates [51], increases as more fallbacks are needed, we evaluate the benefit of prececcing the fallbacks with the DAG in parallel so that the performance gap between no fallback and 3 fallbacks can be eliminated.

Figure 29(b) shows a comparison between serial and four-way parallel lookup costs without the fast-path processing in our software router prototype. We observe that with intra-packet parallism, the lookup cost for all four FBx cases. We deliberately exclude the I/O and synchronization cost in the parallel lookup since the overhead of such intra-packet parallelism is high in our current software implementation. However, we believe that such parallel processing overhead will be minimal in hardware router implementations or highly-parallel SIMD systems such as GPU-based software routers.

5.4 Fast Flat Table Lookup

This research was performed by Hyeontaek Lim, Bin Fan, Dong Zhou, and Anuj Kalia, CMU Ph.D. students supervised by David Andersen. Some of the research was done in collaboration with Dr. Michael Kaminsky (Intel Labs).

Background and Goals - One of the challenges of the XIA project is to be able to forward packets at high rates based on large flat addresses. Forwarding tables can be very large, especially in some of the more radical designs for handling, e.g., CIDs. While we reiterate that the goal of XIA is not to solve these problems perfectly—but instead to provide a framework in which the best solutions can take over in an evolutionary manner—progress and reasonable prototypes are critical for evaluating whether XIA is likely to be able to support such designs in the future. Therefore, one of our research efforts has been directed at exploring questions surrounding the "what-if" scenarios of large flat addresses: Is it worth exploring designs that require certain large sizes of forwarding tables? Can we reasonably expect future switches and routers to be able to support those designs, were they to become important and popular? Ultimately, solutions will combine a number of techniques, include improved storage and processing technology, lookup algorithms, and cache management strategies.

Within exact lookup, we have developed highly memory-efficient hash table implementations based upon novel data structures of our own design, and using our extensions to Cuckoo hashing. Next, we applied these algorithms to create a highly memory-efficient, Flash-based datastore for small key-value pairs in which the key is a flat, cryptographic identifier such as the SHA hashes used in XIA to identify hosts and contents. The "storage" aspects of this work were presented at SOSP 2011 by Ph.D. student Hyeontaek Lim [80]. Our work in this area also includes two different algorithms that work well together: A compact "trie-based" approach that can locate key-value pairs on external storage (Flash, etc.) using less than 1 byte of RAM, and a memory-efficient, high-performance in-memory map based upon Cuckoo hashing [79].

Ph.D. student Bin Fan has also taken the cuckoo hashing ideas on their own and implemented a faster, more memory-efficient variant of the popular "memcached" in-memory cache; this work was published at NSDI 2013 [42]. This work forms the initial foundation for a high-performance DRAM-based content cache for XIA CID routers. From an algorithmic standpoint, the core contribution of this work was the development of a multiple-reader, single-writer high speed version of cuckoo hashing. In general, this data structure is suitable for extremely high-throughput, parallel accesses to a large, efficiently packed table of items such as cache entries or forwarding table entries. We have been able to scale this data structure to handle up to one million updates per second while simultaneously answering hundreds of millions of read queries per second.

Approach and Findings - We improved the performance of our Cuckoo-hashing data structures beyond the level described in the NSDI paper, and specialized it for ethernet-style forwarding table lookups. We

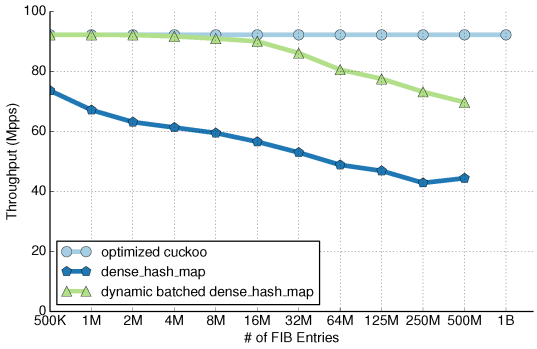


Figure 30: CuckooSwitch 64-byte packet forwarding throughput w/ different lookup tables. The CuckooSwitch design saturates the forwarding hardware regardless of table size, and requires less DRAM to do so. (Throughput is limited in this example by the NIC cards' PCI bandwidth.)

have combined it with an advanced software router infrastructure, Intel's "Data Plane Development Kit" (or DPDK), to create a best-in-breed commodity computer-based Ethernet switch that can saturate 80 gigabits per second from a lookup table of one billion entries using 192 byte packets. This work was published at CoNext in December 2013 [125].

We further generalized these tables for write-heavy workloads, work described at EuroSys 2014 [78]. As a result, we now understand not only how to construct huge forwarding tables for flat-addressed networks, but also that those designs *can* be effectively extended to designs that require extremely fast updates, such as might be required in fine-grained mobility or per-flow state updates. One figure from our result in shown in Figure 30: our CuckooSwitch design is capable of maintaining full forwarding throughput even up to 1 billion forwarding table entries; the best other hash tables are both slower and run out of memory at 512M entries.

We also began preliminary efforts to extend this work to even larger—scalable cluster switches. The question we sought to explore was whether a sufficiently large cluster would be able to efficiently route to trillions of flat addresses, such as might be required for a hypothetical XIA-based design that provided global service routing [44]. (It bears emphasizing that we do not necessarily believe doing so is a good idea – this exercise is designed to understand what is *feasible*). We recently extended this research to a full cluster switch architecture that can effectively scale up to 16-32 switch nodes, supporting over a terabit of forwarding capacity.

5.5 Exploration of Hardware Environments

Along a different dimension, we wanted to understand how concepts such as XIA's addressing and extensibility might apply in certain hardware-dictated environments.

Background and Approach - Our research on implementing XIA has so far focused on traditional software protocol stacks. Some implementations are however pushed towards leveraging significant hardware support for performance reasons. We selected two example environments: the LTE-to-IP gateways used in 4G cellular networks, and the RDMA-capable hardware now becoming increasingly popular in modern datacenters. To a certain degree, this effort is a (carefully selected) "fishing expedition", in which our goal is to begin exploring an unfamiliar corner of the network as a way of discovering and challenging hidden assumptions built into XIA.

Findings - On the RDMA front, we have made good progress in understanding the capabilities and opportunities afforded by RDMA hardware, with a paper published at SIGCOMM'14 [63]. This work

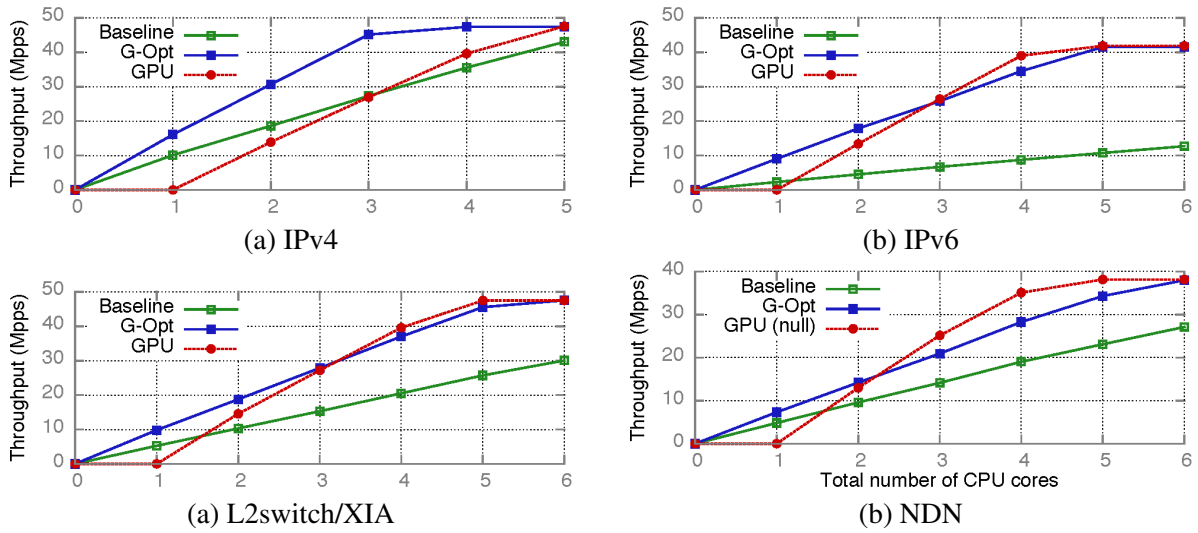


Figure 31: Throughput for different address formats.

builds on a careful measurement study of the throughput and latency that can be achieved by combinations of the communication methods available in RDMA hardware, and demonstrates the effectiveness of the study by constructing a best-in-class RDMA-based key-value store. With this study under our belt, we are just now beginning to explore questions such as the role of RDMA in future network architectures, both as a *target* (i.e., can XIA addresses be extended to make RDMA easier, or to work across the wide-area?), and as an enabler (i.e., can RDMA support be used in conjunction with new network architectures to achieve better throughput or scalability?).

On the LTE cellular front, we are beginning to extend our work on high-performance cluster switches to the IP gateways used in LTE cellular, in conjunction with researchers from Intel. Because these gateways maintain state on a per-connection basis, they require large flat tables, and we believe that gaining expertise with this increasingly important network edge architecture will enhance our ability to evaluate XIA and other network architectures. This work is underway, with one Ph.D. student from CMU (Dong Zhou) spending four months at Intel Research in Oregon to begin this line of work.

Finally, in the process of developing large, flat lookup tables, we incidentally developed an improved "Bloom filter"-like data structure, a function that has been widely used in networking applications. For inexact matching and caching, we have developed a new cuckoo-hashing based algorithm that we call Cuckoo Filters. These filters operate much like traditional Bloom filters providing set membership tests: If an object O is in set S, the test will always return true. However, if the object is not in the set, the filter may sometimes incorrectly return true (a false positive). Our Cuckoo Filters are substantially more memory efficient than Bloom filters are when the desired false positive rate is low (1% or less). We further increase their memory efficiency by ordering the entries in a particular way through the cuckoo filter so that they can be compressed. We have collaborated with domain expert Michael Mitzenmacher to finish the development and analysis of this data structure, and it has been accepted for publication at CoNext in December 2014 [43]. While this result was not one of our goals at the outset, we believe that its improved memory efficiency may also have bearing on many of the "large scale" aspects of FIA projects—for example, Bloom filters have been proposed as mechanisms for supporting name lookup and gossip in Named Data Networking and as one way of supporting mobility in earlier designs from the Mobility First team.

5.6 Fast Forwarding for FIAs

We continued our efforts to answer questions surrounding the performance of practical implementation paths for both XIA and other FIA designs (specifically NDN). Building upon our results using Intel's high-performance Data Plane Development Kit (DPDK) as a foundation for implementing software forwarding for XIA and other next-generation, flat table designs, we have extended our efforts towards other commodity implementation technologies. This research was performed with CMU Ph.D students Hyeontaek Lim, Bin Fan, Dong Zhou, and Anuj Kalia and with collobarations with Dr. Michael Kaminsky (Intel Labs).

Current progress - We have completed our exploration of examining LTE cellular gateways, with Ph.D. student Dong Zhou having returned from Intel and having built a relationship with a software LTE gateway vendor. This work resulted in a submission to SIGCOMM'15 (under submission as of this writing). The results were positive: In this context, the techniques we developed for scaling flat addressed XIA networks proved effective at improving the large, flat NAT tables used in LTE gateways. They reduced memory overhead by a large amount, and improved packet forwarding performance by over 20%. We believe these results will have application both in today's networks, but also in creating scalable and high performance gateways to connect, e.g., FIA network designs back to the core Internet, for which NAT or NAT-like techniques are typically useful. (For example, while XIA's 4ID can be used to tunnel XIA packets between XIA "islands" on the IPv4 Internet, NAT-like techniques are needed to connect XIA-only hosts to IPV4-only hosts. Similar restrictions apply to most other FIA projects.)

Several FIA groups have been exploring the application of GPUs (Graphics Processing Units) to packet forwarding in their architectures. For example, a group from Tsinghua and the University of Science and Technology of China implemented NDN-style name lookup on the GPU in work published in NSDI'13 [116]. Strong claims have been made about the ability of GPUs to speed up forwarding in NDN [107], IPv6 [53], as well as non-architectural uses such as intrusion detection systems[61]. In our work to appear at NSDI'15, Ph.D. student Anuj Kalia demonstrates that most of these gains are not due to the use of the computational hardware in a GPU, but rather, because the GPU provides hardware-assisted thread switching that can mask DRAM lookup latency [64]. By providing a thin wrapper around software-based latency hiding techniques, Kalia's work demonstrates how to achieve similar *and often better* performance using the CPU alone. This work has important implications for x86-based implementation of all FIA designs, and is accompanied by a software release that facilitates creating high-performance artifacts for further experimentation. It also suggests that the GPU is not, at present, an interesting middle ground for many FIA projects, including XIA—designs are likely to benefit from skipping directly to FPGA implementation if the performance of an x86-based router prototype is insufficient.

Figure 31 compares the throughput for different address formats. Full system throughput with increasing total CPU cores, N. For the GPU, N includes the master core (throughput is zero when N = 1 as there are no worker cores). The x-axis label is the same for all graphs. Without our software optimization, GPUs appear beneficial for several forwarding designs for both hierarchical lookups (IPv4, IPv6), flat lookups (l2switch), and hierarchical name-based lookups (NDN). With the g-opt transformation, however, CPU-based forwarding can match the throughput of GPU-based forwarding, without the additional hardware cost and code complexity.

Future work - Based upon our experience, we plan to next switch our XIA prototype router's data plane to use the DPDK and the advanced forwarding designs we have created, as a part of deploying a cost-effective but sufficiently performant testbed.

5.7 A Native XIA Stack

The XIA group at Boston University has realized a Linux implementation of the core XIA network architecture and has now progressed into exploration of other architectures onto Linux XIA. The primary

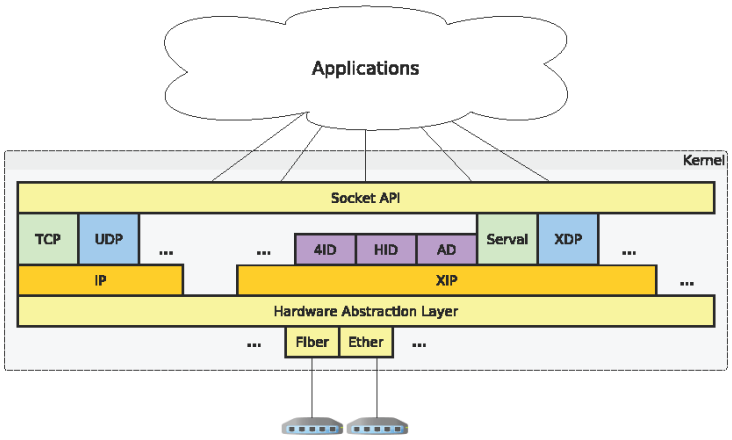


Figure 32: The XIA Linux Network Stack

goal of this effort is fourfold: to streamline upcoming network deployments of XIA, to provide a concrete demonstration of XIA's capability to evolve and import novel architectural functionality created by others, to facilitate other network researchers to conduct architectural evaluations enabled by the existence of Linux XIA, and to reach beyond network researchers to other potential early adopters of XIA. The centerpiece of this activity is the growing open-source implementation, publicly available on Github at `https://github.com/AltraMayor/XIA-for-Linux`. This effort has been spearheaded by graduated Ph.D. student (now XIA post-doctoral researcher) Michel Machado and PI John Byers. Other team members contributing to the effort include Cody Doucette, and B.U. undergraduate Alexander Breen.

Background - While we provide more detail on the status of the Linux XIA implementation and our proposed research next, the brief synopsis is that the current implementation provides a full implementation and integration of the BSD socket interface running with XIP as the internetwork layer protocol, as well as forwarding and routing support for host, content, and autonomous domain principals (HIDs, CIDs, and ADs respectively), and basic UDP-style transport furnished by the XDP transport layer protocol. In addition, implementations of the various X4ID principals to provide backwards compatibility with IPv4 legacy networks were implemented, tested, and evaluated. A depiction of the architecture is provided in Figure 32. On top of this foundation participants at Boston University worked towards realization of other network architectures as principals within the XIA framework. As described in Michel Machado's Ph.D. thesis [82], the team ported the Linux implementation of the Serval service-centric architecture to XIA, and produced white paper ports of the NDN architecture, as well as the MobilityFirst architecture, and the ANTS active network protocol.

Approach and Findings - The native implementation of XIA for Linux's kernel has a realization of core principal types AD, HID, and XDP (a UDP-like protocol), and a simple version of NWP (our replacement for ARP (the Address Resolution Protocol). In addition, a large set of new principals have been added to accommodate the various ports of alien architectures. These include a realization of service IDs following the Serval design, which also necessitated principal types for Serval flow IDs and a new longest prefix match (LPM) principal to realize hierarchical routing. To realize legacy IP compatibility, a set of four X4ID principals were introduced, and reuse of the interoperable LPM principal type was also integrated into the design.

The current codebase sports state-of-the-art technologies such as LXC (a lightweight form of virtualization), and RCU (a lockless synchronization mechanism), as well as innovation with XIA principals

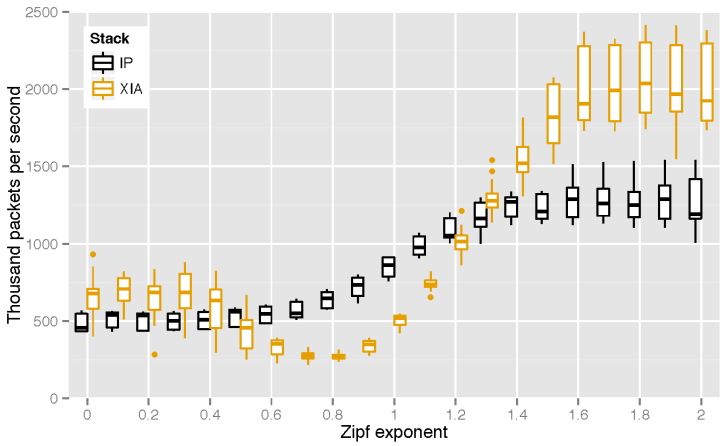


Figure 33: Linux IP vs. Linux XIA forwarding performance

implemented as loadable kernel modules (LKMs). To the best of our knowledge, Linux XIA is the first network stack whose binary image that can be extended or reduced, on-the-fly, according to the needs of network administrators. The development process follows a bottom-up approach that we hope will allow us to make sure that XIA meets or exceeds all hardware and operational demands asked from a production-level network stack.

The knowledge and experience accumulated in Linux's IPv4 and IPv6 stacks are being considered in our design not only at the code level, but also at code structure and design level. For example, we have adopted Linux's DST abstraction (a cache of routing decisions), and the shortcomings of ARP for XIP have led us to the design a new protocol to map network addresses to hardware addresses, namely, Neighborhood Watch Protocol (NWP). Our experiences with re-designing XIP in conjunction with companion protocols IP and ARP have led us to a set of design alternatives, the best of which seem to offer a cleaner, more elegant, and potentially more evolvable functional decomposition that aligns with the XIA notion of extensible principal types. In addition to sound design, extensive performance evaluation of our data plane functionality was conducted and demonstrating that XIP forwarding in the Linux kernel is competitive with IP forwarding in the Linux kernel. A plot of IP vs. XIP forwarding throughput obtained by a 16-core Linux router in forwarding a representative workload to (a heavy-tailed distribution of) 40K destination addresses is depicted in Figure 33 (full details appear in Machado's Ph.D. thesis).

Moving specifically to design and evaluation of new XIA principals spurred by an interest in demonstrating evolvability, the most significant effort was in porting the Serval architecture to XIA. Porting Serval to XIA derived many benefits: a realization of the service ID (SID) functionality in XIA, a demonstration of XIA's capability to evolve, a deepening of our understanding on the role and scope of new principal types in XIA (as Serval most naturally decomposed into three principal types, not a single one). While space limitations preclude a full description here, Figure 34 demonstrates connection set-up in Serval XIA and conveys some of the intricacy of our design. As a final note, a key take-away finding from our port is that realizing Serval in XIA directly activates intrinsic security, cleanly remediating a key weakness in the original Serval/IP design.

5.8 Toward Comprehensive, Smart Session Management

The XIA team started to explore the design and added benefits of a session layer that allows users and applications to control the richer network control offered by future Internet architectures such as XIA. We refined

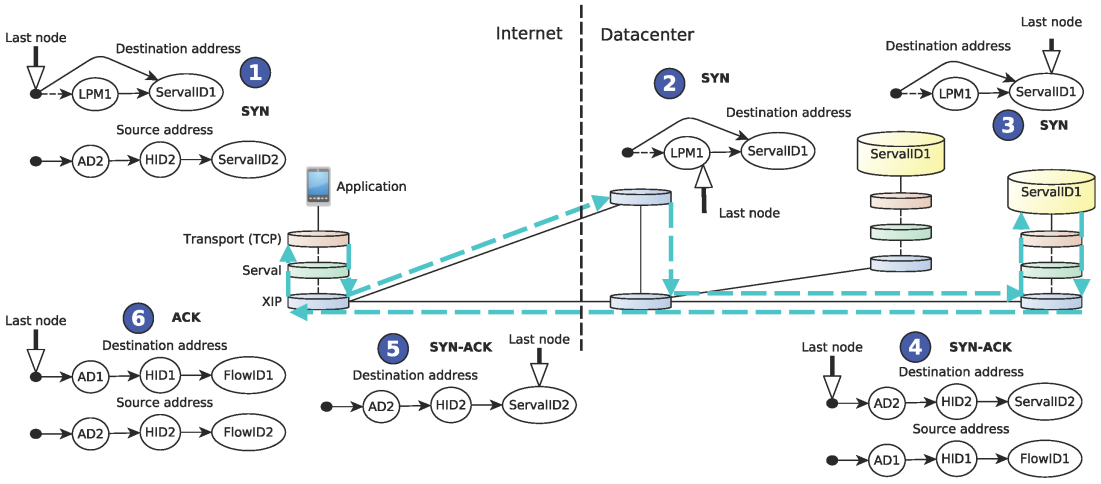


Figure 34: XIA Serval three-way connection handshake

an earlier session-layer design, built a prototype, and devised new experimental methods for evaluating it. This research is being conducted by David Naylor, Matthew Mukerjee, and Peter Steenkiste.

Background and Goals - Originally, the Internet was based on a model in which stationary applications communicated end-to-end over a very simple core [104]. All functionality was implemented at the endpoints and applications were satisfied with only the reliable bit pipe provided by TCP. As a result, most network communication sessions were simple TCP connections. The reality today is entirely different. First, applications' networking needs have become much more complex. For example, many applications desire mobility (e.g., ability to switch networks, transport connections), various in-path services (e.g., to process or translate documents), and authentication/security (e.g., to protect sensitive data). Second, future internet architectures such as XIA offer applications and users much more control over how their communication operations are executed (Section 1.8). Examples for XIA include choice of XIA type and DAG, path selection using Scion, and the network-supported servies and content retrieval.

To accommodate these needs, many of the features described above have been implemented as application-level libraries—for example, TLS libraries offer encryption and authentication. Also, the "knobs" for managing lower-level functionality implemented by the OS, like choosing among multiple NICs, were simply added to the existing socket interface for lack of a better place to put them. This approach leaves applications in nearly complete control over our network communication. This is undesirable for two reasons: First, many aspects of a session directly impact users, yet how much control (if any) the user has is left up to the application. Second, applications either do not have access to contextual information about the network and the device—like battery level or current cellular data usage—that could be used to configure "smarter" sessions, or they do not have an incentive to gather and use it.

The focus of this work is the design of a session manager that can configure rich sessions, with the following goals:

1. All parties (users, administrators, and applications) have control over the session.
2. Sessions should be configured "intelligently" based on relevant contextual information.
3. The session manager itself should be organized so as to be maintainable and extensible.

Approach and Findings - Figure 35 shows our new design for managing rich communication sessions.

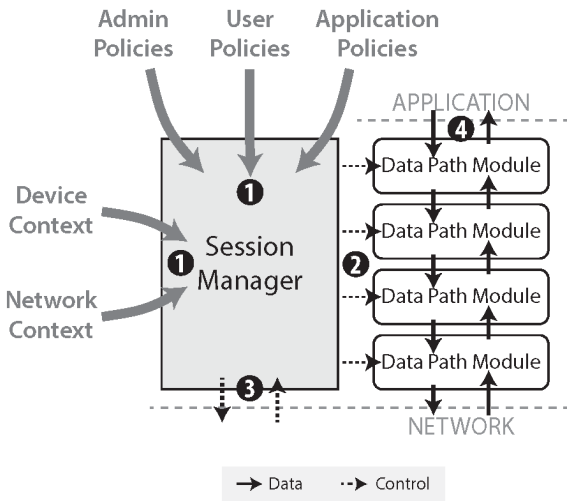


Figure 35: High level design: a session's data path is implemented as a sequence of modules, which are selected, ordered, and configured by the session manager.

The data path consists of a sequence of *data path modules* that implement specific pieces of functionality (e.g., encryption or a transport protocol). Each module encapsulates both data processing code (e.g, encrypt and decrypt) and control code to initialize internal state and potentially signal a remote counterpart module (e.g., TLS handshake). Data flows from one module to the next in an order determined at session setup time by the *session manager* (SM) based on input from all relevant parties combined with information about the status of the network and the device. A key difference with the original design is that the execution flow in the session manager is now necessarily mirror the data path flow, but is instead determined by any dependencies that may existing session configuration decisions.

The life of a session proceeds as follows:

1. The SM collects user, application, and administrator preferences as well as network and device context.

2. Based on the preferences and context, the SM selects a set of data path modules to use for the new session.

3. The SM finalizes the set of data path modules with the remote endpoint and then invokes each module's initialization code.

4. The application endpoints begin exchanging data over the newly configured data path, using control functions as needed.

5. The SM monitors the context for changes, updating the data path appropriately.

Our design differs from today's protocol stack in two important ways. First, generic functionality above the transport layer, which naturally fits in the "missing" session layer, is today implemented as application-level libraries. Instead, we have packaged this generic functionality as data path modules, which all implement uniform data and management interfaces, making them easier to "stack" for combined use in the same session. Second, we have separated management from the data path—each layer is no longer responsible for managing the layer beneath it. Since session management happens outside the application, it is easy to (1) use contextual information that the application does not have (or has no incentive to gather) to help configure the session and (2) take into account the preferences of users and administrators.

We have a prototype implementation of our session manager and of data path modules for compression and encryption; the next step is add features and port an existing application to run on top of our new protocol stack.

6 Secure Network Operations

Security is a major goal of the XIA project. Research activities include defining a security architecture for XIA, the Scion path selection protocols, source authentication and path validation, secure and accountable key infrastructures, and accountability delegation for packets.

6.1 Security Architecture

The definition of the XIA security architecture was led by David Andersen and Adrian Perrig.

6.1.1 Background and Goals

In contrast to today's ad-hoc approach of securing individual components of the network, one of the goals of XIA is to constructively build security into our architecture as a well-defined, foundational property. As such, security (broadly defined) should be considered a key consideration throughout the design, starting on day 1. But this is not sufficient. Internet security involves many challenges, and addressing them one by one is not practical. Instead, we need a systematic approach that applies a coherent set of principles to all aspects of security. The security architecture described below does exactly this.

The development of this security architecture early on in the project has two major goals. First, we want to ensure that the key elements of the XIA architecture put us in a position to address all aspects of security, including not just traditional security properties such as identity authentication, reliability, and integrity, but also more user-oriented considerations such as transparency (as defined by the User group, users can learn what happens to their data as it traverses the network) and concerns about privacy. This is assessment is done at the conceptual level, in the sense that the XIA team is not a position to design, implement and evaluate security mechanisms for all possible security threats. Second, the security architecture will help us identify specific security questions to focus on throughout the XIA project.

6.1.2 Approach and Findings

The major questions tackled by the security group thus far are: (1) What is an appropriate meaning of "self-certifying" for the principals we are defining; (2) How do we ensure that the composition of components in XIA result in desirable high-level security properties; and (3) addressing the specific problem of availability and isolation, described in the following subsection.

One of the emerging security challenges in XIA is that "security", loosely defined, is an emergent property. As a highly flexible, evolvable architecture, the behavior of the system depends not only on the core XIA architecture, but also upon numerous decisions about how principals are defined, what mechanisms operate on them, and other specific extensions that future designers choose. For example, the "core" XIA architecture does not define a specific way for routing packets to other hosts—one could choose a BGP-like mechanism, or use the SCION architecture described in the next section.

XIA has three high-level security goals:

- Support today's Internet-style host-to-host communication with drastically improved security (SCION, next section);
- Provide even better security for the two classes of communication we identify as being important today: content retrieval and accessing services;
- Lay the groundwork for future extensions / principals to make good decisions regarding security and availability.

We consider five major security properties: *Availability, Authenticity, Authentication and Accountability, Secrecy, and Trust management*. Privacy is covered by secrecy of identity. XIA applies four design principles to achieve these properties:

- *Well-foundedness*: Identifiers and associations match the user's intent;
- *Fault isolation*: Designs that reduce dependencies and insulate correctly functioning parts of the network from misbehaving parts;
- *Fine-grained control*: Allow users to clearly specify their intent, so that the network gains maximum flexibility to act to satisfy that intent;
- *Explicit chain of trust*: Allow users to understand the basis for trust and what assumptions underlie that trust.

The majority of our security work to date uses the first three principles in order to ensure correctness and high availability in the face of malicious or misbehaving components. We have worked through numerous cases of how to specify principals such as content or services to increase availability and provide confidence in the correct operation of the network. Consider the case of accessing a service as one representative example. In XIA, services are identified by the hash of a public key, where the service holds a copy of the corresponding private key.

Network-level availability -
XIA enables the use of the SCION architecture to provide robust network-level connectivity to an individual node. SCION further can provide multiple paths from the client to that node, increasing robustness to path failures or DoS.

Intrinsic integrity verification -
Because services are named by a public key, any client accessing the service can intrinsically verify that it is communicating with an appropriate instance of the service.

Trustworthy service replication -
Services are identified in a manner distinct from the node(s) hosting the service. As a result, they can be replicated easily, either within a cluster or across the wide-area. Any authorized instance of the service can authenticate itself to the network (for reachability announcements) and to clients (for integrity).

Easy client or server migration -
For mobile clients or to fail-over rapidly, XIA's self-certifying addressing makes it easy to re-bind an existing connection. When a client moves to a new network, for example, it retains its same host ID (the hash of the client node's public key). It can thus communicate with the service to inform it of the client's new location in the network, re-bind any existing connections, and seamlessly continue to communicate.

In the case of content, XIA can further increase availability by allowing trustworthy caching and retrieval of content by untrusted caches—clients can use these caches without fear, because they can verify that the resulting content hash matches what the client was trying to retrieve.

The XIA security architecture was presented at the NSF FIA PI meeting in Oakland. The slides are available at `http://www.cs.cmu.edu/afs/cs/project/xia/www/Documents/xia-security-arch.pdf`.

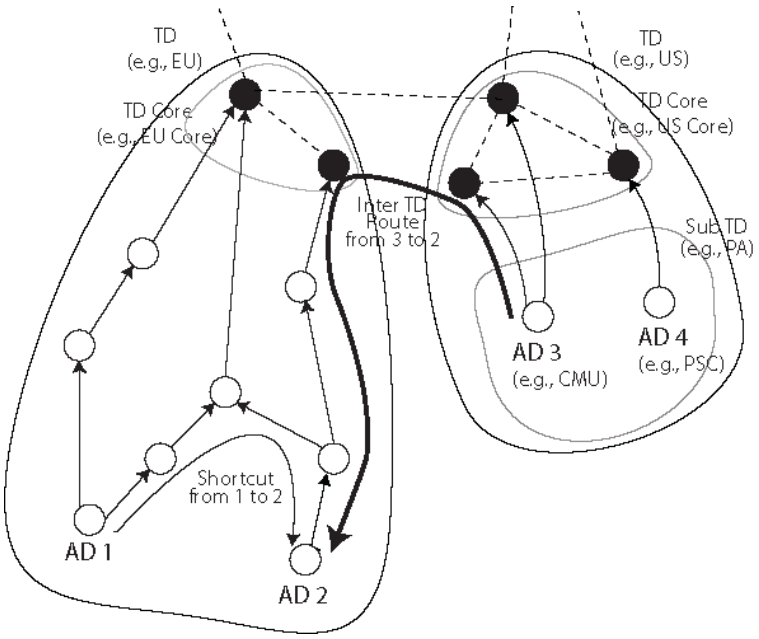


Figure 36: Trust domain architecture.

6.2 Scalable, Secure Path Selection

The SCION research was conducted by the Security group headed by PI Perrig. The Network, and Policy and Economics groups were involved in the discussion on implications for network forwarding and inter-ISP relations.

6.2.1 Background and Goals

Path selection in the current Internet is based on BGP. It has a number of well-known security problems and it gives users very control over the paths their traffic takes through the network. There have been a number of proposals, such as BGPsec, to secure BGP, but these approaches have a very narrow security goals and they do not address many of the shortcomings of BGP. In contrast, SCION takes a clean slate approach to path selection. The goal for SCION is to provide route control, failure isolation, and explicit trust information for end-to-end communications, in other words, the design of SCION is consistent with the security design principles driving the security architecture for XIA, as described in the previous section. SCION provides one mechanism for basic point-to-point communication in XIA. The hope is that SCION offers a very high level of availability that XIA can rely on.

6.2.2 Approach and Findings

SCION separates ASes into groups of independent routing sub-planes, called Trust Domains (TDs), 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.

Figure 36 presents the general trust domain architecture, depicting two trust domains as well as a sub-trust domain. Black nodes are Autonomous Domains in the Trusted Domain Core. Arrows indicate customer-provider relationships. Dashed lines indicate peering relationships. The top-tier ISPs within a trust domain form the TD Core, and they help manage the trust domain. This separation is highly useful for achieving security properties by isolating and excluding untrusted entities from the trust domain, and grouping entities that can agree on a uniform trust environment. We have worked out a complete basic architecture for SCION and have published our paper at IEEE Symposium on Security and Privacy [124].

6.3 Light-weight Anonymity and Privacy

Given the importance of anonymous communication, we have explored architectural extensions that would enable highly efficient topological anonymity. This research was conducted by the PI Perrig and his students, in collaboration with Marco Gruteser from the Mobility First team. We published a joint paper called "LAP: Lightweight Anonymity and Privacy" at the IEEE Symposium on Security and Privacy in 2012 [57].

6.3.1 Background

Popular anonymous communication systems often require sending packets through a sequence of relays on dilated paths for strong anonymity protection. As a result, increased end-to-end latency renders such systems inadequate for the majority of Internet users who seek an intermediate level of anonymity protection while using latency-sensitive applications, such as Web applications. This paper serves to bridge the gap between communication systems that provide strong anonymity protection but with intolerable latency and non-anonymous communication systems by considering a new design space for the setting. More specifically, we explore how to achieve near-optimal latency while achieving an intermediate level of anonymity with a weaker yet practical adversary model (i.e., protecting an end-hosts identity and location from servers) such that users can choose between the level of anonymity and usability. We designed Lightweight Anonymity and Privacy (LAP), an efficient network-based solution featuring lightweight path establishment and state-less communication, by concealing an end-hosts topological location to enhance anonymity against remote tracking. To show practicality, we demonstrated that LAP can work on top of the current Internet and proposed future Internet architectures XIA / Scion and Mobility First.

6.3.2 Approach and Findings

As a result of this research, we were able to reduce the high communication latency with high in-network computation and storage state of current anonymous communication systems. Especially the high latency causes the Internet browsing experience to endure a significant slowdown. Anonymous communication would thus be more usable with reduced overhead. Indeed, we believe that many users can live with a relaxed attacker model, as they can trust their local ISPs but want protection from tracking by ISPs that are further away (potentially in other countries with different privacy laws) and from tracking by websites. Given such a weaker attacker model, we attempt to provide source and destination anonymous communication, session unlinkability, and location privacy at a very low overhead, barely more than non-anonymous communication.

In this framework, our approach is simple yet effective: by leveraging encrypted packet-carried forwarding state, ISPs that support our protocol can efficiently forward packets towards the destination, where each encrypted ISP-hop further camouflages the source or destination address or its location. Although encrypted packet-carried forwarding state is currently not supported in IP, we design simple extensions to IP that could enable this technology. In particular, our approach is especially applicable in future network architectures,

where the design can be readily incorporated. This new point in the design space of anonymity protocols could also be used in concert with other techniques, for example in conjunction with Tor to prevent one Tor node from learning its successor. Despite weaker security properties than Tor, we suspect that LAP contributes a significant benefit towards providing topological anonymity, as LAP is practical to use for all communication.

6.4 Source Authentication and Path Validation

Source authentication and path validation are fundamental network primitives to help mitigate network-based attacks, such as DDoS, address spoofing, and flow redirection attacks. Earlier work developed on a light-weight protocol that defends against such attacks [71]. Here we describe a formal validation of this protocol [123].

Background - Source authentication and path validation are fundamental network primitives to help mitigate network-based attacks, such as DDoS, address spoofing, and flow redirection attacks. Furthermore, path validation provides a way to enforce path compliance according to the policies of ISPs, enterprises, and datacenters. In particular, end-hosts and ISPs desire to validate service level agreement compliance regarding data delivery in the network: Did the packet truly originate from the claimed client? Did the client select a path that complies with the service providers policy? Did the packet indeed travel through the path selected by the client?

We proposed Origin and Path Trace (OPT) – a lightweight, scalable, and secure protocol for source authentication and path validation. OPT enables all the entities on a path to authenticate the origin and the content of the received packet, and to validate the path on which the packet traveled.

Although several protocols have been proposed to implement source authentication and path validation, they have not been formally analyzed for their security guarantees. We thus applied proof techniques for verifying cryptographic protocols (e.g., key exchange protocols) to analyzing network protocols.

Approach and Findings - To formally verify the security of OPT, we collaborated with Dr. Limin Jia, an expert in formal verification and security protocols. As a first step to verify OPT, we encode LS2, a program logic for reasoning about programs that execute in an adversarial environment, in Coq. We also encode protocol-specific data structures, predicates, and axioms. To analyze a source-routing protocol that uses chained MACs to provide origin and path validation, we construct Coq proofs to show that the protocol satisfies its desired properties. To the best of our knowledge, we are the first to formalize origin and path authenticity properties, and mechanize proofs that chained MACs can provide the desired authenticity properties.

Analyzing network protocols is far more complex than analyzing cryptographic protocols, as the analysis needs to consider arbitrary network topologies. Furthermore, the protocol programs as well as the properties are recursive, often bound by the size of the network. If we were to apply model checking tools for analyzing cryptographic protocols naively, we would have to fix the size of the network topology and limit ourselves to verifying properties specific to a set of topologies. It is unclear how to use these model checking tools to prove a general property that holds for all topologies. In some special cases, small-world theorems have been attempted for simpler ad hoc wireless protocols, where only a finite number of topologies need to be checked to conclude that a property holds on all topologies. Unfortunately, to the best of our knowledge, general small-world theorems applicable to verifying path validation protocols do not exist. Therefore, model checking techniques cannot be immediately applied to proving properties of these protocols. As a first step towards understanding the formal aspects of these source authentication and path validation protocols, we manually construct proofs of general security properties that are independent of the network topology.

We analyze the OPT protocols that use chained Message Authentication Codes (MACs) to provide

source authentication and path validation for routers. These protocols have two phases: key setup and packet forwarding. We prove the secrecy and authenticity of keys of the key setup phase, and origin and path authenticity properties of the forwarding phase. To the best of our knowledge, we are the first to formalize origin and path authenticity properties of packet forwarding provided by chained MACs, and construct machine-checkable proofs of these properties.

Our main findings include:

- We encode LS2 and basic constructs for reasoning about cryptographic protocols in Coq.
- We formalize and construct machine-checkable proofs of the secrecy and authenticity properties of the key setup protocol.
- We formalize and construct machine-checkable proofs of the origin and path authenticity properties of the forwarding protocol.

6.5 Secure Public-Key Infrastructures

In this reporting period, we continued our research on creating secure public-key infrastructures. We published two papers [15, 110]

Background - In the current trust model of TLS PKI, a single compromised (or malicious) Certification Authority (CA) can issue a certificate for an arbitrary domain. Moreover, such bogus certificates can go unnoticed over long periods of time. This glaring vulnerability is widely recognized. In response, the research community has proposed different approaches to mitigate this problem. Recent proposals include Certificate Transparency (CT), which adds accountability by using log servers to make compromises visible, and the Accountable Key Infrastructure (AKI) that prevents attacks by using checks-and-balances to prevent a compromised CA from impersonating domains. Although such proposals provide good starting points and building blocks, they require many interacting features to be viable and thus are inherently highly complex. History has shown that humans will miss cases when considering the security of such complex systems. Moreover, they must satisfy effi- ciency requirements and fit with existing business models, as well as offer improved security. Finally, even advanced proposals such as CT and AKI are still incomplete and have been designed in an ad-hoc fashion, without a formal proof of correctness.

We now took the next step and get assurance of both completeness of features as well as correctness of the security claims, which can only be achieved by using a principled approach. We thus create the Attack Resilient Public-Key Infrastructure (ARPKI), the first co-designed model, verification, and implementation that provides accountability and security for public-key infrastructures. In contrast to other PKI proposals, ARPKI offers:

- substantially stronger security guarantees, by providing security against a strong adversary capable of compromising n - 1 entities at any time, where n *geq* 3 is a system parameter that can be set depending on the desired level of security;
- formal machine-checked verification of its core security property using the Tamarin prover; and
- a complete proof-of-concept implementation that provides all the features required for deployment and demonstrates its efficient operation.

Approach and Findings - We propose ARPKI, a public-key infrastructure that ensures that certificate-related operations, such as certificate issuance, update, revocation, and validation, are transparent and accountable. ARPKI is the first such infrastructure that systematically takes into account requirements identified by previous research. Moreover, ARPKI is co-designed with a formal model, and we verify its core

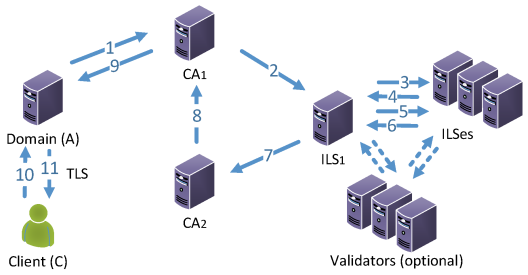


Figure 37: Basic communication flow of ARPKI.

security property using the Tamarin prover. We present a proof-of-concept implementation providing all features required for deployment. ARPKI efficiently handles the certification process with low overhead and without incurring additional latency to TLS. ARPKI offers extremely strong security guarantees, where compromising n - 1 trusted signing and verifying entities is insufficient to launch an impersonation attack. Moreover, it deters misbehavior as all its operations are publicly visible.

ARPKI achieves strong security guarantees using three entities for the certificate operation: two CAs and an Integrity Log Server (ILS). In particular, ARPKI's CAs conduct active on-line confirmations with validator-like capabilities. Consequently, ARPKI prevents compromise attacks such that even when $n-1$ trusted entities are compromised, the security guarantees still hold.

We briefly provide a high-level summary of the actors and their responsibilities in this scheme: a domain registers an ARPKI certificate (ARCert) for itself with the ARPKI infrastructure, and can afterwards use the resulting ARCert to securely serve webpages to clients. The CAs check the identity of the domain owner on registration and then sign and give guarantees for the presented certificate. Throughout the lifetime of the ARCert, the CAs are responsible for checking the logs for this ARCert and assuring the correct operation of other entities involved in creating the ARCert. To check the ILSs behavior, the CAs download all accepted requests from the ILSs and compare them to the published integrity trees. The ILSs keep a log of all ARCerts registered with them, in a publicly verifiable way, and provide proofs of existence for ARCerts that are then used by CAs and domains. The set of ILSs synchronizes with each other in a secure and accountable manner. Optionally there can be additional validators, that execute checks similar to those made by CAs, but without issuing ARCerts themselves.

We illustrate the process in Figure 37. We denote that entity E signed message M by $\{M\}_{K_E^{-1}}$, and $H(\cdot)$ stands for a cryptographic hash function. All signatures include timestamps and unique tags such that they cannot be mistaken for one another.

ARCert generation ARPKI supports trust agility, meaning that the domain owners can select their roots of trust and modify their trust decisions using extension parameters. A domain owner creates an ARPKI certificate (ARCert) by combining multiple standard certificates from trusted CAs. Note that in this step each CA checks the identity of the domain owner to authenticate domains correctly. We now consider the owner of domain A registering her domain.

ARCert registration request (Steps 1–2) In ARPKI, three designated entities are actively involved in monitoring each other's operations. The core idea behind a REGISTRATION REQUEST (REGREQ) message is to let the domain owner explicitly designate the trusted entities, namely two CAs and one ILS (CA_1, CA_2, and ILS_1 in Figure 37). Solid numbered lines represent the message flows to register an ARCert, and dotted arrows represent optional flows. 10 and 11 represent a TLS connection after registration is complete.

ARPKI requires the domain owner to contact just one CA (CA_1). The main responsibilities of CA_1 are to validate the correctness of the other two entities' operations and act as a messenger between the domain

owner and ILS_1 and CA_2.

The domain owner also designates ILS_1 to ensure that ARCert_A is synchronized among all ILSs. CA_2 mainly takes the validator's role and ensures that ILS_1 as well as other ILSs operate accordingly, e.g., add ARCert_A to their Integrity Trees as promised.

ILS synchronization (Steps 3–6) Ideally the same ARCert_A should be publicly visible among all ILSs. However, synchronizing *all* ILSs may be inefficient, incurring significant time delay, and unrealistic. Instead, in ARPKI ILS_1 takes responsibility on behalf of the domain owner to synchronize ARCert_A among at least a quorum of all existing ILSs. This ensures that only one ARCert_A is registered for the domain A, and the majority of the world maintains a consistent certificate entry for the domain A in order to prevent impersonation attacks.

Registration confirmation (Steps 7–9) When the majority of ILSs agree to add ARCert_A to their public Integrity Trees, ILS_1 schedules domain A's ARCert to appear in its Integrity Tree during its next update (i.e., at the end of the current `ILS_UP` time interval), which is stated and signed in an ACCEPTANCE CONFIRMATION (ACCEPT) message. ILS_1 then sends to CA_2 a REGISTRATION RESPONSE (REGRESP) message, which serves as a proof to CA_2 that ILS_1 (and a quorum of ILSs) indeed accepted domain A's REGREQ.

CA_2 now takes the validator's role to monitor and ensure that ILS_1 indeed made the majority of ILSs agree to accept ARCert_A for their next update time. CA_1 also takes the validator's role to monitor that CA_2 correctly monitors ILS_1.

TLS connection (Steps 10–11) The domain A now has a confirmation message (ACCEPT) that is signed by three trusted entities, and upon receiving ACCEPT along with ARCert_A, clients can ensure that they are establishing a TLS connection with domain A.

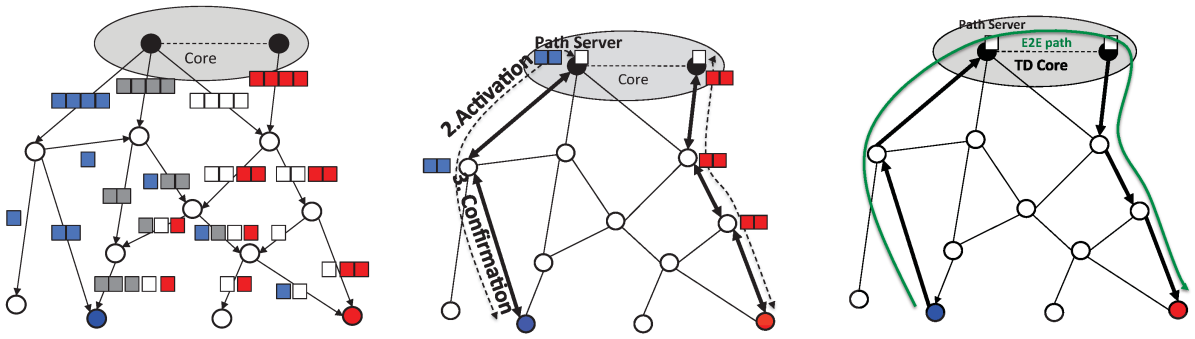
Clients can validate an ARCert against an ACCEPT message by verifying that the confirmation (1) is authentic, meaning the confirmation is signed by trusted entities, (2) has not expired, and (3) is correct.

6.6 DDOS Defense

We have worked on DDoS-resilient Internet architecture by leveraging SCION. One of the key high-level features of XIA is that it offers users more control over how communication operation are performed (Section 1.8). As emphasized in the XIA security architecture (presented at the PI meeting in Oakland [119]), having alternative ways of communicating improves availability. Leveraging Scion path selection, as presented here, is one example of this.

Background - DDoS attacks are still prevalent in the Internet today. In fact, a recent world-wide security survey [91] suggests that Botnet-driven DDoS attacks have become common as a low cost, high-profile form of cyber-protest. Both attack intensity and frequency have drastically accelerated: the largest reported attack size doubled every year, to more than 100 Gbps seen in recent attack against Spamhaus [19, 85]. The majority of network operators in the survey also ranked DDoS attacks as the biggest threat.

To address this problem, we designed a new DDoS-resilient Internet architecture named STRIDE [56] that isolates attack traffic through viable bandwidth allocation, preventing a botnet from crowding out legitimate flows. This new architecture presents several novel concepts including tree-based bandwidth allocation and long-term static paths with guaranteed bandwidth. In concert, these mechanisms provide domain-based bandwidth guarantees within a trust domain; each administrative domain in the trust domain can then internally split such guarantees among its endhosts to provide (1) connection establishment with high probability, and (2) precise bandwidth guarantees for established flows, regardless of the size or distribution of the botnet outside the source and the destination domains. Moreover, STRIDE maintains no per-flow state on backbone routers and requires no key establishment across administrative domains. We also demonstrate that STRIDE mitigates emerging DDoS threats such as Denial-of-Capability (DoC) [12] and Coremelt attacks [108] based on these properties that none of the existing DDoS defense mechanisms can achieve.



(a) Announcement (b) Activation/Confirmation (c) End-to-end Path

Figure 38: End-to-end bandwidth-guaranteed path-establishment in STRIDE. The small rectangles in the figure represent a unit bandwidth.

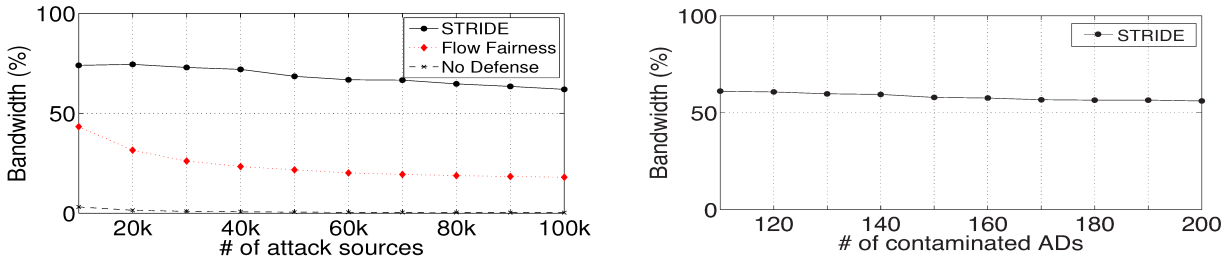
Approach and Findings - Our architecture is based on the following insights: (1) Bandwidth allocation is simple in a tree-based topology, as the available bandwidth can be split from the root down to each leaf; (2) with network capabilities encoded in packet-carried state and fixed bandwidth classes, routers can perform enforcement in a per-flow stateless fashion using probabilistic detection of the largest flows; (3) by combining a static long-term traffic class guaranteeing low bandwidth with a dynamically-allocated short-term traffic class guaranteeing high bandwidth, we can provide a variety of guarantees to both privately communicating endhosts and public servers. Figure 38 illustrates the steps for establishing a end-to-end bandwidth guaranteed channel: (1) bandwidth announcement along Path Construction Beacons (PCBs), (2) activation/allocation of a chosen bandwidth-guaranteed path, (3) confirmation of the activation request, and (4) end-to-end path establishment.

Even with our relatively simple operations, we can achieve protection against DDoS from large botnets. Figure 39 shows some simulation results performed with real datasets.

STRIDE's bandwidth guarantees effectively isolate the bandwidth of attack traffic from that of legitimate traffic. As a consequence, in STRIDE, the effects of attacks are confined within the paths they follow regardless of whether attack sources flood a single path (or a link) or multiple paths simultaneously. We show this bandwidth isolation via large-scale simulations. For realistic simulations, we construct simulation topologies using a CAIDA SkitterMap [21], attach 10,000 legitimate sources to 200 ASes proportional to the AS size, and attach attack sources (hosts) to 100 ASes. Paths are probabilistically sampled from the SkitterMap to satisfy both the number of sources and the number of ASes. We increase the attack size from 10K to 100K to compare STRIDE bandwidth guarantees with those of a per-flow fair-sharing based mechanism. We consider a baseline case, labeled as "No Defense", where packets are randomly dropped during congestion.

Figure 39(a) shows the bandwidth used by the legitimate flows that originate from clean ASes. Under "No Defense", the legitimate flows obtain almost no bandwidth. DDoS attacks. When per-flow fair-sharing bandwidth control is employed, attack flows cannot completely exhaust the target's link bandwidth, yet the attack effects grow linearly with the attack size.

Next, we increase the number of contaminated ASes by 10 up to 200 ASes. As one can imagine, the bandwidth of legitimate flows decreases as Figure 39(b) shows. However, the effects of attack dispersion are marginal (i.e., proportional to the number of attack ASes) because the dynamic channel bandwidth is proportional to the static channel bandwidth and the static channel bandwidth that can be used by attack traffic is limited by the number of attack ASes in STRIDE.



(a) Effects of attack size

(b) Effects of attack dispersion

Figure 39: STRIDE's bandwidth guarantees to legitimate traffic under various DDoS scenarios.

6.7 Accountable Key Infrastructure

XIA relies on an explicit, trusted name to address translation mechanism. This is especially important in XIA since DAG-based address are based on cryptographic identifiers. We developed an accountable key infrastructure which proposes a new public-key validation infrastructure [72]. Public keys form the basis for both XIA HIDs and SIDs.

Background - Recent trends in public-key infrastructure research explore the tradeoff between decreased trust in Certificate Authorities (CAs), resilience against attacks, communication overhead (bandwidth and latency) for setting up an SSL/TLS connection, and availability with respect to verifiability of public key information. To further minimize latency, the web server can acquire validator information right after getting registration confirmation from Integrity Log Servers (ILSes). Then, the web server can staple validator information during the HTTPS connection setup. AKI integrates an architecture for key revocation of all entities (e.g., CAs, domains) with an architecture for accountability of all infrastructure parties through checks-and-balances. The parties involved for checks-and-balances include CAs, public log servers called Integrity Log Servers to make CAs behavior publicly visible, and Validators that monitor log servers. AKI efficiently handles common certification operations, and gracefully handles catastrophic events such as domain key loss or compromise. AKI would make progress towards a public-key validation infrastructure with key revocation that reduces trust in any single entity.

Approach and Findings - We provide a high-level overview of AKI with the following example.

Figure 40 depicts an overview of our AKI architecture. Alice owns domain A.com and wants to obtain an AKI-protected certificate, as she wants to protect herself against compromise of the CAs that signed her certificate and other rogue CAs, and protect her clients against compelled certificates. To define the security properties that she intends to achieve for her domain, Alice defines CAs and Internet Log Server (ILS)[2] operators that she trusts, the minimum number of CA signatures that she recommends her clients for validation, and rules for certificate revocation, replacement, and updates. Alice includes these parameters with her public key and contacts more than the minimum number of trusted CAs (according to her security policy) to sign her certificate. She then registers the certificate with one or multiple ILSs. Each ILS adds A.com to its database, by placing it in the Integrity Tree. The ILS then re-computes hash values over all stored certificates for updated verification information. Alice now supplements her certificate with the verification information that she downloads from every ILS, and sends it to browsers that connect to her web site via HTTPS. For certificate validation, the browser uses the trusted root-CA certificates as in current practice, and uses the pre-installed ILS public key(s) on her browser to validate ILS information. Before trusting the ILS information received from Alice, the client browser occasionally checks with validators to confirm that the ILSs current root hash values are valid. To reduce latency, we also describe later how the

[2]that log domains certificates and make them publicly available

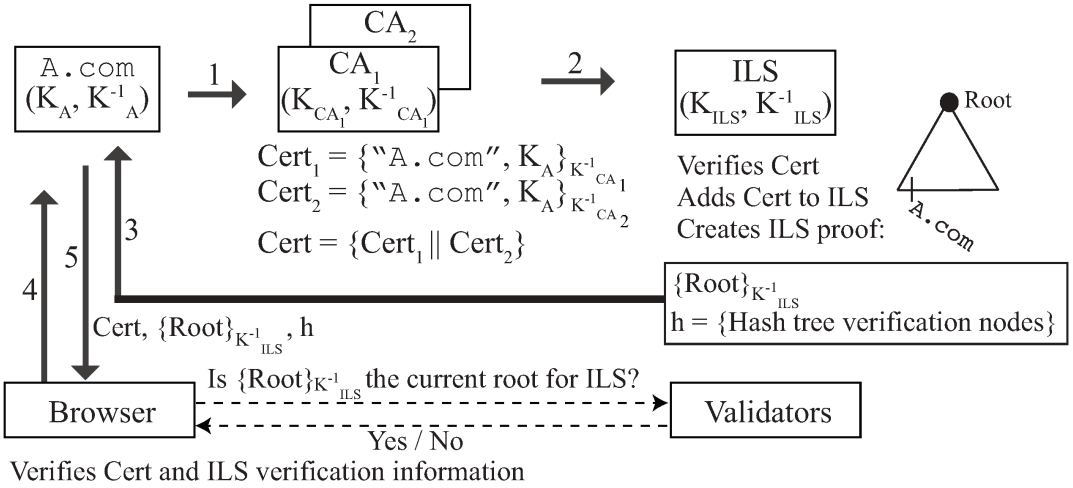


Figure 40: AKI certificate registration process. This figure depicts A.com registering a certificate (signed by two CAs) to a single ILS, but the domain can acquire multiple certificates from multiple CAs and register to multiple ILSs. Dashed arrows represent occasional communications.

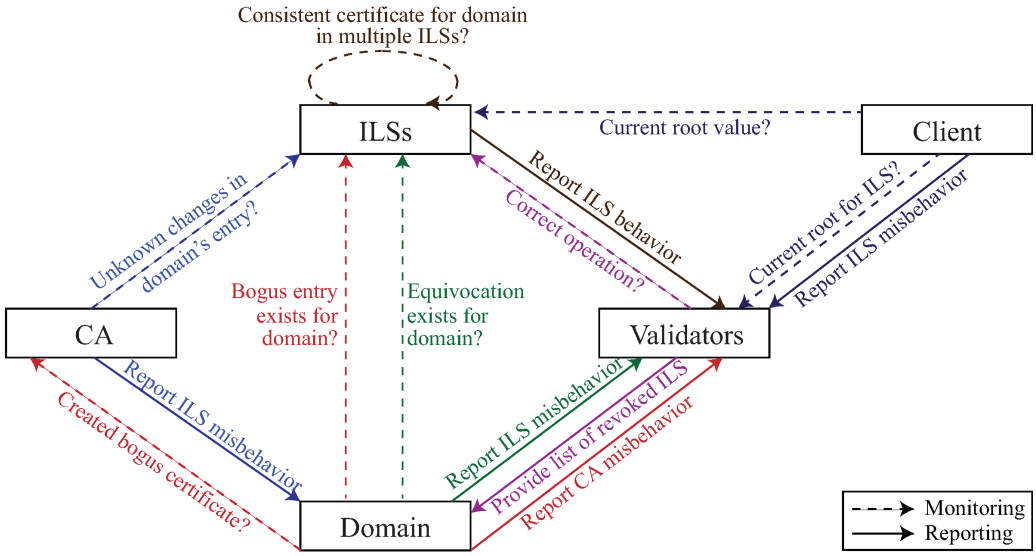


Figure 41: Checks and balances among AKI entities. In AKI, each entity monitors other entity for misbehavior detection and reports to other entities.

the web server can staple validator information during the HTTPS connection setup.

An important aspect of AKI is that all actions are digitally signed, such that any misbehavior can be demonstrated based on the entities signatures. Consequently, an accusation for misbehavior can be checked without trusting the accuser, ruling out slander attacks. Equivocation is an example for a misbehavior, where one party provides different answers to the same query – for example an ILS server who would create two different Integrity Trees in a given time period and uses either tree to respond to different entities. Since only

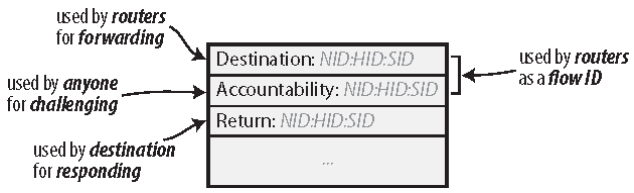


Figure 42: Packet carry a destination address (used by routers for forwarding), an accountability address (used to report malicious packets), and an optional return address (used by the receiver to respond).

a single Integrity Tree can exist per ILS per time period, the demonstration of the two ILS-signed Integrity Trees demonstrates ILS misbehavior. The notion of checks-and-balances are illustrated in Figure 41.

7 Balancing Accountability and Privacy

Throughout the project's lifetime, the role of a packet's source address has been a recurring discussion topic. Here we explore how an architecture's treatment of the source address enables a balance between accountability and privacy. This work was done by David Naylor, Matthew Mukerfee, and Peter Steenkiste and was published in SIGCOMM 2014 [90].

Background - Today's Internet is caught in a tussle [27] between service providers, who want accountability, and users, who want privacy. At the network layer, mechanisms for providing one or the other often boil down to either strengthening or weakening *source addresses*. In an accountable Internet, source addresses undeniably link packets and senders so miscreants can be punished for bad behavior, so techniques like egress filtering and unicast reverse path forwarding (uRPF) checks aim to prevent spoofing. In a private Internet, senders hide source addresses as much as possible, so services like Tor work by masking the sender's true source address.

We argue that striking a balance between accountability and privacy today is fundamentally difficult because the IP source address is used both to to identify the sender (accountability) *and* as a return address (privacy). Given the freedom to redefine how source addresses are interpreted in XIA, we asked the question "What could we do if the accountability and return address roles were separated?" Our answer, the Accountable and Private Internet Protocol (APIP), does just that, creating an opportunity for a more flexible approach to balancing accountability and privacy in the network.

Approach - APIP ***separates accountability and return addresses*** (Figure 42). A dedicated accountability address allows us to address the limitations of an accountability-above-all-else approach like AIP by introducing ***delegated accountability***. Rather than identifying the sender, a packet's accountability address identifies an *accountability delegate*, a party who is willing to vouch for the packet. With accountability handled by delegates, senders are free to ***mask return addresses*** (e.g., by encrypting them end-to-end or using network address translation) without weakening accountability.

Figure 43 traces the life of a packet through APIP.

1. The **sender** sends a packet with an *accountability address* identifying its **accountability delegate**. If a *return address* is needed, it can be encrypted or otherwise masked.

2. The **sender** "briefs" its **accountability delegate** about the packet it just sent. We call this operation **brief()**.

3. A **verifier** (any on-path router or the receiver) can confirm with the **accountability delegate** that the packet is a valid packet from one of the delegate's clients. Packets that are not vouched for are dropped. We call this operation **verify()**.

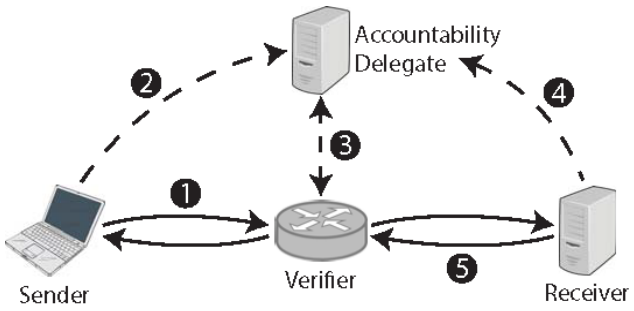


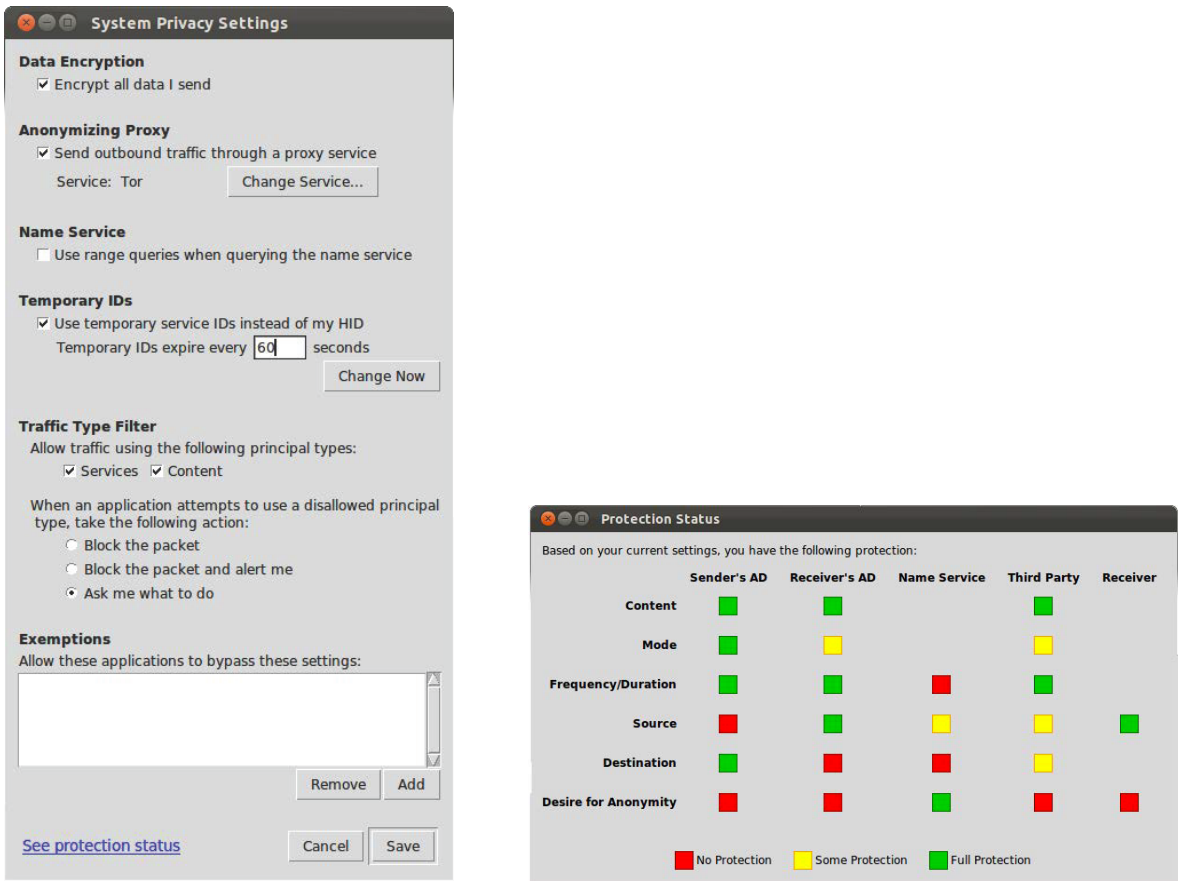
Figure 43: High-level overview of APIP.

4. If the **receiver** determines that packets are part of a malicious flow, it uses the *accountability address* to report the flow to the **accountability delegate**, which stops verifying (effectively blocking) the flow and can pursue a longer term administrative or legal solution. We call this operation **shutoff()**.

5. The **receiver** uses the *return address* in the request as the *destination address* in the response.

Findings - We evaluated the feasibility of delegated accountability using a trace of NetFlow data from CMU's border routers containing ten million flows. Here we highlight two results about the cost of the **brief()** operation, which, since performed for every packet, is potentially high-overhead. First, briefing requires sending the delegate information about each packet a host sends. In our trace, briefs consume only an additional 2.5% of the original traffic volume, which was just under 10 Gbps; batching briefs in data structures like bloom filters could reduce this to 0.25%–0.5%. Second, the delegate must store the briefs it receives in preparation for **verify()**s and **shutoff()**s. Assuming a single delegate serves all hosts in our trace, it would need a maximum of 3GB of storage space to keep each brief for two minutes (likely longer than necessary).

Finally, we evaluated the privacy benefits of APIP. If a sender uses its source domain as a delegate, this depends on the size of that domain. Raghavan et. al. find that, if ISPs were to aggregate prefixes geographically, over half of the prefixes advertised by many popular last-mile ISPs would include more than 10 million IPs [100]. If a sender uses a third party delegate, the anonymity set grows the farther a packet travels from the source domain. To see how much, we use Route Views [1] data from January 1, 2014 to roughly estimate AS "fanout." For each AS, we track how many customer networks sit beneath it in the AS hierarchy. Notably, 50% of ASes originating prefixes have at least 180 first-hop "siblings" and 90% have over 900 second-hop siblings.



(a) Controlling network-level privacy

(b) Privacy status window

Figure 44: Example designs for controlling and monitoring network level privacy

8 User Privacy

The network traffic that users generate exposes possibly sensitive information about them to observers, such as who they communicate with (web sites or individuals) and what information they exchange. While protocols such as TLS make it relatively easy to encrypt content, hiding the actual data being exchanged, the network headers may expose the identity of the communicating parties. What is precisely visible to observers depends on the format of the network-layer header which is part of the network architecture. The goals of this research is to better understand the privacy concerns users have, understand the privacy implications of network architecture features, and to develop techniques to help users manage their privacy for different architectures. This research was led by Sara Kiesler and Laura Dabbish.

8.1 Privacy Button

The user group and core networking group are collaborating on an application that will users manage various privacy aspect of their network communication.

Internet users are faced with a number of privacy concerns. Some of these concern relate to the services that users access over the network, e.g. search engines and social networking sites; these concerns must be addressed using privacy controls that are specific to the service. Other concerns relate to the use of the network, e.g. confidentiality of the data or the identity of the communicating end points. These privacy threats tends to more difficult for users to understand and there is no support for systematically addressing

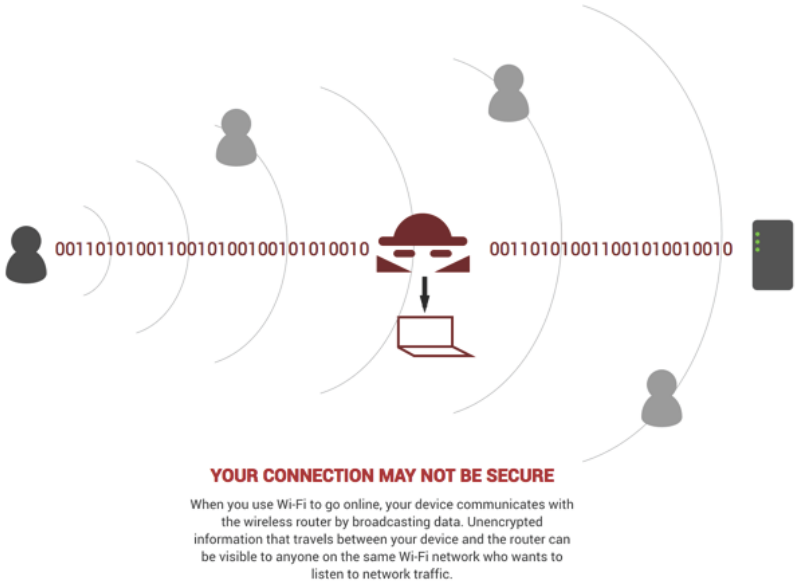


Figure 45: Example output from the leak detector

them. For example, users often rely on applications to maintain their privacy, but application support is uneven. Our aim in this project is to develop a practical means for users to maintain their privacy and anonymity throughout for a session by providing support at the level of the operating system, and therefore across many types of online activity. This project started as a course project in the Fall 2011 CS graduate networking course at CMU and is now led by David Naylor and Peter Kinnard, PhD students in the CS and HCI departments at CMU, advised by PIs Kiesler and Steenkiste.

We have designed a partial solution to maintaining privacy and anonymity in online communications. The idea is develop a user interface that gives users information about their current communication operations. For example, the interface can show whether traffic is encrypted or not and what ISPs are used for forwarding traffic. More importantly, the interface also also gives them controls over changing how communication is performed, allowing users to address specific privacy concerns. Figure 44 shows a possible user interface based on the basic prototype that was developed as part of the course project. For example, users can enable/disable encryption or enable onion routing solutions such at Tor. On architectures such as XIA, users can also avoid using certain principal types (such as CIDs) that reveal too much information. The application takes a new approach to privacy and security by moving control from the application (e.g., a web browser or email client) to the operating system, allowing users to manage their privacy at the session level rather than per-application. Finally, since some of these techniques involve additional extra overhead, we would like to give user an indication of the cost of enabling a specific technique, e.g. in terms of increased latency that might affect delay-sensitive applications.

While this is a simple idea, realizing it raises many challenges. A first question is where to best intercept the traffic, e.g. at the socket level or deeper in the stack, or possibly even as a proxy in the network. Another challenge is what techniques to use, how to incorporate them in the system, and how to deal with new failure modes that my be created. A final challenge is designing a user interface that is easy to use and interpret by users.

8.2 Leak Detector

We explored ways to communicate information about sensitive data leaked by users' machines. This research is performed by PhD student Ruogu Kang and a group of undergraduate students, supervised by Sara Kiesler and Laura Dabbish.

Background - The leak detector, developed by David Naylor of the original XIA project, is a tool that extracts publicly visible and potentially personal information from a user's traffic in real time. There is great potential to use this information not only to help users control what information is revealed over the network, but also to improve the accuracy of users perceptions of anonymity and privacy. Prior work suggests that people are not well aware of threats of unencrypted information leakage over, for example, Wi-Fi networks, and when they learn about threats, they may not care or be motivated to change their privacy and security behaviors, or may show only short-term behavioral changes. Furthermore, social and environmental context plays a key role in people's privacy concerns and disclosures. Therefore, this research will examine what features of privacy/security threat (how the threat is presented and social/environmental context of the threat) influence participants' security sensitivity (e.g., awareness and knowledge of security threats and motivation to take privacy/security enhancing behaviors). The results of this research will help us improve the effectiveness of the leak detector in terms of helping users by changing their behavior.

Approach and Findings - This research consists of a series of user studies using a variety of tools, such as the leak detector, that provide feedback on privacy/security. An initial user study conducted as part of an HCI course and continued last summer with the help from a group of undergraduate students, used a set of different interfaces providing feedback to users in different formats, levels of detail, and frequency. The preliminary results suggested that users did not like detailed and frequent information because it disrupts the activity they are trying to complete. The study also showed that warnings about privacy leaks often result in only short-term changes in behavior. Follow up interviews suggest that making users aware of the impact of privacy leaks, e.g., explaining how leaked information could be used to collect additional information on the user, might be more effective, but more research is needed to support this conclusion.

Next we will build on these initial results by varying not only the nature of the information given to the user, but also the scial/environmental context where study tasks take place: (1) showing Leak Detector output of leaked search history vs. showing no Leak Detector output; (2) showing Leak Detector output vs. showing Leak Detector output plus inferred information about the user; (3) presenting A concrete threat like a hacker vs. a broader threat like Amazon; and (4) varying social context such as a public area with others physically present vs. an isolated setting such as a lab. During our user studies, participants will search for products on shopping web sites they will receive feedback on the privacy/security risks of their searches, e.g., who might see their searches and feedback. Participants will also be provided with information about different privacy and security strategies that they can use to protect their online activity (Figure 45).

8.3 Mental models of the Internet

We used user studies to gain a better of understanding how lay users perceive the Internet. This research was performed by Tatiana Vlahovic, Ruogu Kang, Laura Dabbish and Sara Kiesler.

Background - Results from the first XIA project show that users are often not clear on what privacy threats are associated with specific activities on the Internet. For example, our early work identified some of the reasons why users try to cover their digital traces online or communicate anonymously. The results showed that the threats perceived by users often do not match actual threats for reasons based in inaccurate knowledge about the Internet. More recent studies show that users are often not aware of real threats, leading to risky computer use. Since threats are context dependent (e.g., activity, location), a first step in helping users deals with privacy is to educate them on what (real) threats they should be concerned about in specific

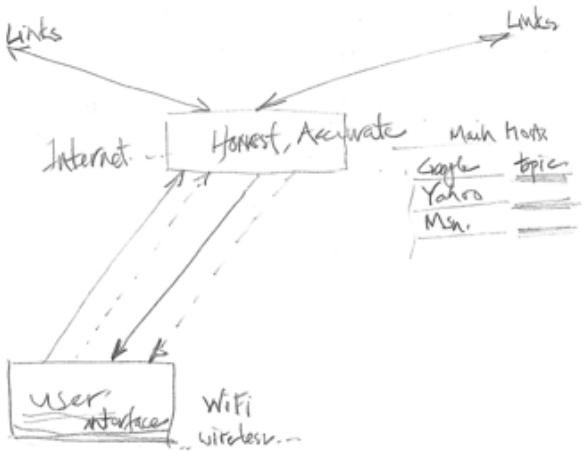


Figure 46: Example of a lay user's perception of "Internet as a service."

contexts.

A commonly used method in psychology to elicit users understanding about a problem is mental models, which are mental representations of things, processes, and situations. Mental models describe how a person thinks about a problem or system; it is the model in the persons mind of how things work. These models are used to make decisions by supporting mental simulations of the likely effect of an action. Mental models of a system can be useful in informing the design of systems because they suggest natural ways to visualize complex system components or user interactions. Our focus this reporting period was on understanding peoples mental models of the Internet so we can understand what information needed for for informed decision making is missing, so we can determine how to best make available through tools or education.

Approach - One of the first steps we took was to develop a short test of Internet knowledge that could be used in privacy and security surveys. The test we developed has three main scales: one scale to test basic knowledge of how the Internet works, another to scale to test more advanced knowledge of the network, and a third test of know-how in using privacy and security tools. We also developed a scale measuring a persons sense of privacy threats, and another to measure preventive actions the person takes. We gave these scales to a large group of MTurk users, smaller samples at CMU, and students in one networking class at CMU. The scales have good reliability, and correlated well with formal computer science background; scoring high on the scales is associated with a more technical education. Scores on the network scale correlated .40 with grades in the small computer science network class.

Although the above results are a step to understanding users and what they know, they do not give us a holistic understanding of what users think is going on when they communicate online. To answer these questions we did three rounds of data collection in which we screened for technical Internet knowledge, privacy and security concerns, and assess users mental models of the Internet by asking them to explain how the Internet works, and to draw a general diagram of it in whatever form they chose on a large sheet of paper. Then we asked them to draw several diagrams about specific tasks they do on online such as watching a YouTube video, sending an email, making a payment online, receiving an online advertisement and browsing a webpage.

In the mental model tests, the users included participants (technical) with computer science or software engineering education (n = 11) and participants (lay) with non-CS college or graduate education such as finance, law, or culinary arts (n = 17). We also gave our survey to students in a computer science network course (n = 19) and to a large sample of mTurk users (n = 400).

Findings - Based on our mental model studies and our surveys on larger groups, lay users, in general, have a mental model of the Internet as a simple system or service (for example, see a users drawing in

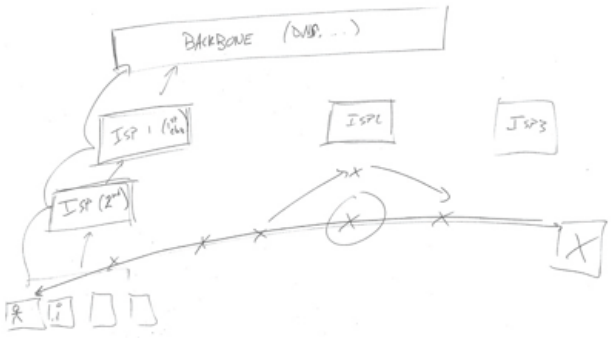


Figure 47: Example of a technical user's mental model of the Internet.

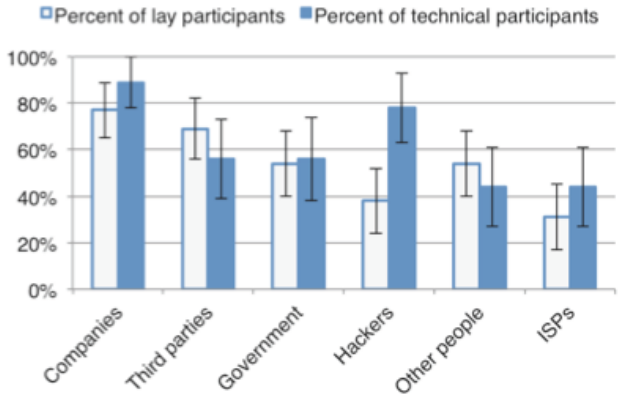


Figure 48: Percent of lay and technical participants who mentioned each group that might have access to their data.

Figure 46). They only have an awareness of entities that they interact with directly, and they lack awareness of underlying layers, structures, and connections. They also use inconsistent or invented terminology. By contrast, most technical users have a more articulated mental model of the Internet (for example, see Figure 47). They represent the Internet as a complex, multi-level system, have more awareness of components and organizations in the network, and tend to use accurate, detailed, and consistent terminology.

These different mental models, in turn, are associated with differing awareness of privacy and security threat. Lay users think of the Internet as a black box and have little idea of the sources of privacy or security threat, although they believe threat exists and are worried about it. By contrast, technical users have a more concrete notion of who can access their data (Figure 48). Lay people allude to unknown third parties and strangers whereas those with a technical background mention specific entities.

When it comes to protective actions, however, we have found fewer differences between technical and lay participants. The two groups are about equally likely to be proactive in protecting their data by, for example, using anti-virus programs and being cautious when using public Wi-Fi. In our mental model tests, only a third of each group said they managed risks by changing their passwords when asked and exiting possibly malicious websites. Technical participants, however, are more likely to know about and actively use tools such as a proxy server, anonymous search engine, or encryption. Most lay users had only vague knowledge of those tools.

We have also found that users in both groups have many reasons not to take protective actions, as has been reported by other researchers [4]: feeling they have nothing to hide, being too busy with their main

task, and being uncertain or doubtful that any action they took would really be effective. There are also too many privacy and security choices to make, and users in both groups feel overwhelmed by them. Further, as expert reviewers who looked at our data noted, technical users might be overconfident about their knowledge as protective in itself.

Our surveys do show that formal education in computer science, and even more, in networking, raises awareness of the sources of privacy risks online and the likelihood of taking protective actions. However, the findings are correlational and do not establish that users need a computer science or networking background to learn to be safe or to manage their expectations of privacy. We are therefore engaged in further research to discover what kinds of education or information needs to be conveyed to users about the Internet to improve their behavior. Ruogu Kangs dissertation is meant to help answer this question by giving her participants visualizations that explain the structure of the Internet and sources of threat. She will be testing if this information changes peoples mental models and gives them more accurate knowledge of the sources of threat and how to mitigate it.

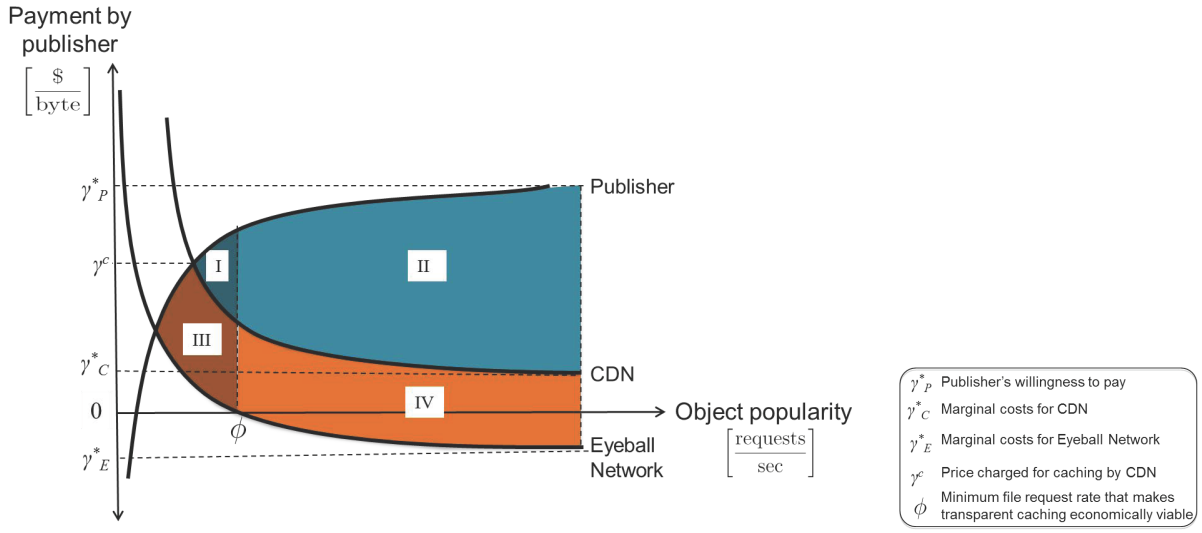


Figure 49: Analysis of viability of caching by CDNs and eyeball networks

9 Public Policy and Economics

The goals of this research is understand how design decisions at the network architecture level impact public policy and economic incentives for deployment new network services.

9.1 Policy Study on Caching

This research was conducted by the Policy and Economics group, led by PIs Jon Peha and Marvin Sirbu at Carnegie Mellon, with participation by PI John Byers at Boston University. Participating PhD students included Patrick Agyapong (CMU) and Chong Wang (BU). Michel Machado (BU) also participated in the regular telephonic group meetings.

9.1.1 Background and Goals

In our proposal we highlighted the importance of understanding the implications of XIA architecture on industry structure and revenue models. The focus of our work to date has been on understanding these implications, particularly with respect to content-centric networking.

9.1.2 Approach and Findings

First, we analyzed the economic incentives for operators to provision router level caches [7]. Our results show that, in the absence of direct payments from publishers, ISPs will only invest in caching to the extent that it saves internal bandwidth. This level is below the global optimum that would be realized if ISPs received direct payments from publishers who reap additional benefits from better performance and reduced server costs of their own.

We also examined the competitive positioning of edge networks, transit networks or third party content delivery networks in pricing caching services. We find that edge networks, because of the terminating access monopoly, are in a position to dominate other providers of caching services in the competition for publisher payments for caching services, either by providing these services themselves, or extracting any rents through inter-carrier payments. Quality of service issues that have been subject to debate under the

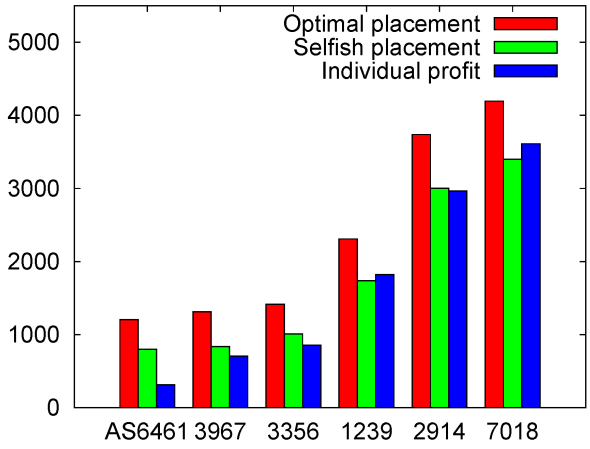


Figure 50: Profit of each ISP under different pricing models

heading of Network Neutrality, emerge in CCNs as debates over allocation of limited caching capacity among publishers. More details can be found in [7].

Next we extended this analysis to examine the implications for the sharing of producer surplus between networks, CDNs and publishers [8], including an identification of the deadweight loss when edge networks charge above optimal prices to transit networks or CDNs for interconnection. For example, Figure 49 shows how publisher payments and the cost of caching for CDNs and eyeball networks impacts the viability of caching. For example, when an eyeball network performs commercial caching (i.e., gets payments from publisher), then the surplus in all four regions is realized among the eyeball network and publishers. However, when a CDN does commercial caching, then only the values in regions I and II are realized. When an eye ball network performs only transparent caching (i.e., opportunistically caches content that it forwards), then only files with popularity greater than phi are cached with the surplus in regions II and IV; in this case the gain to the eyeball network is limited to a reduction in bandwidth to its provider. We also generalized the results for the case of multiple competing edge networks.

Supporting caching as an ISP service has a number of implications for core protocol design in XIA. We developed a requirements document for caching. The document considers issues such as: metadata required in content chunks for billing and cache management purposes including support for third party billing agents; publisher requirements for information on content deliveries from cache; and mechanisms for aggregating across an AD and reporting to publishers the provision of caching services by routers within the AD. Next, we further developed these ideas with a focus on the special requirements of content chunking for video and streaming media.

We have also been interacting with the security group to clarify differences between trust models for routing and for naming, and the economic implications if name resolution does not yield globally unique results.

9.2 Cooperative Concent Caching

We investigated the benefits and viability of cooperative content caching across ISPs in future architectures such as XIA. This research is performed by graduate student Chong Wang and PI Byers, both at Boston University. Chong Wang will graduate at the end of the summer.

9.2.1 Background

Consider the economic incentives for ISPs to place contents inside their networks in a global content-oriented network. In the absence of cooperation, each ISP's caching decisions will be based independently from one another, to optimize what is best for themselves and their customers. This has in part fostered the rise of overlays such as CDNs, which can orchestrate content delivery and optimize content placement *across* ISP boundaries.

9.2.2 Approach and Findings

With the opportunity for greater visibility and control afforded by expressive routing, we show that a set of cooperating providers can decrease costs by judiciously sharing information, coordinating caching actions and making side payments based on the Shapley value solution concept from coalitional game theory. We also considered this question more broadly, and focused on the question of content placement in existing and proposed content-oriented networks.

We argue that lack of appropriate market incentives inhibits ISPs from cooperation, and study design of an appropriate compensation mechanisms to facilitate cooperation. We argue for a model of the content placement decisions made by transit ISPs as a cooperative game, in which the goal for each ISP is to maximize surplus, i.e. minimize transmission costs by cooperative placing contents. A Shapley-value based mechanism is used to distribute the global surplus to all players. Using a general model for transmission cost, we show that with such a mechanism, ISPs have incentives both to cooperate and to employ placement strategies aligned with the global optimal.

The culmination of this work was a comparative modeling and simulation study evaluting design and performance tradeoffs between today's content delivery networks and future content-oriented networks (encompassing two architectures loosely modeled on NDN and XIA, respectively) [115]. One outcome of the work relates to mechanism design and market design, insofar as the work demonstrates the value of architectural features that could facilitate and incent this type of inter-ISP cooperation. Future work could investigate the economic and policy implications of more extensive information sharing between ISPs, as well as the cost-benefit analyses.

Figure 50 shows our simulation results based on real ISP topologies and representative traffic workload. In the plot the profits in the following cases for each ISP: Shapley value in case each ISP's placement strategy is aligned with the global optimal placement; Shapley value in case each ISP only places contents if they are delivered to a client inside the ISP; and individual profit each ISP obtains if ISPs do not cooperate in terms of placing contents. ISPs are sorted by size from the smallest to the largest. We can see that although the largest ISPs can make most of its profit without cooperation, an ISP always makes the most profit by cooperating with other ISPs and employing the optimal placement strategy, which also leads to better welfare for the whole network.

9.3 Edge Directed Routing

This research was conducted by CMU PhD student Patrick Agyapong, advised by PI Sirbu.

9.3.1 Background and Goals

New approaches to trust management and routing, including XIA's SCION, and the ICING proposal being advanced by the Nebula project, shift the economic balance of power between end users and administrative domains (autonomous systems) in the selection of packet routes. Our goal is to understand the economic motivations for such a shift, the change in equilibrium traffic distribution that might result from this shift in economic power, and its implications for pricing network services and for industry structure.

9.3.2 Approach and Findings

Currently network administrative domains (autonomous systems) forward packets over routes which minimize the operators costs (or maximize its revenues). Enabling edge networks to influence route choice may increase these costs. Using a simple economic model, we investigated the conditions under which network operators are better off from deploying edge-directed routing, under various models by which they might price the offering of such a capability to those edge networks which demand it. We looked at four scenarios: 1) no change in prices; 2) an increased in fixed access charges; 3) an increase in volume charges; and 4) per route pricing and identified conditions under which each of those pricing models is sustainable [6].

We also identified five features that must be supported in routing protocols desgined to support edge-directed routing:

- *Knowledge: Edges must have a way of learning of alternative paths;*
- *Choice: Edges must have a way of specifying a policy-compliant path;*
- *Enforcement: Routers must have a mechanism for enforcing path choice;*
- *Metering: Networks need to measure and track the amount of traffic that customers send over edge-directed routes if per-volume pricing is used;*
- *Verification: Verification is the complement of enforcement–it is the ability of the edge to verify that its chosen path was used.*

We noted that verification and enforcement can add significantly to packet overhead, which raises the volume-related costs of supporting edge directed routing, thus making the third pricing option more likely to be necessary.

9.4 Policy-Compliant DAG-based Forwarding

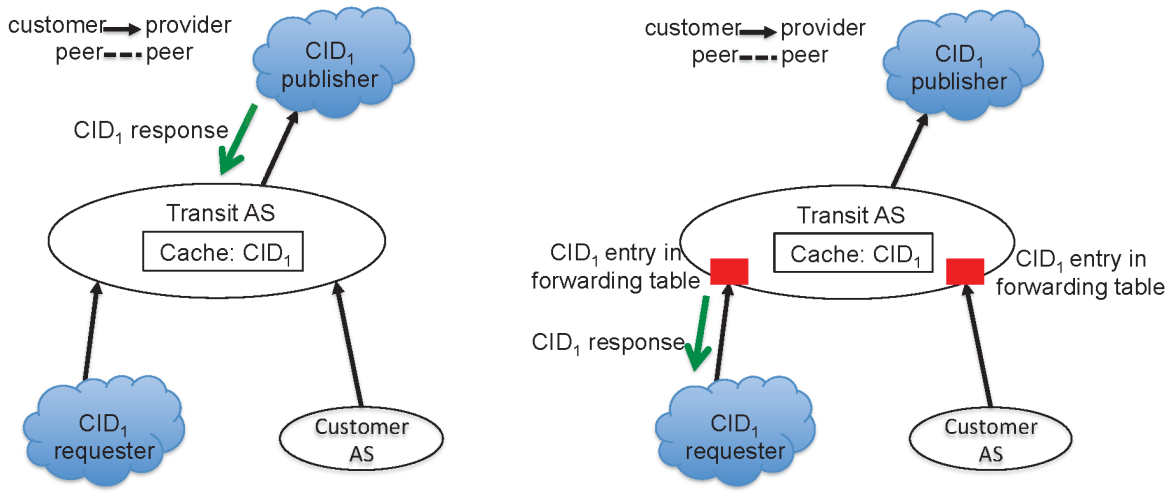
This research was performed by Suk-Bok Lee (postdoc) and PI Peter Steenkiste.

9.4.1 Background

DAG-based addresses, despite their diverse potential uses and advantages, raise a practical question of real deployability in the Internet today. The main problem is that there is a tension between sources' routing flexibility and network owners' transit policies; transit ASes have little control over the traffic traversing their networks, which hinders the route selection based on their own business goals. The focus of this work is to understand the implications of DAG-based on policy-compliant packet forwarding, assuming policies are similar to those used by today's inter domain routing protocol (BGP).

9.4.2 Approach and Findings

We first considered "single-path" source-routing style DAGs, and categorized the cases of taking control over transit ASes' policies. We find that a major conflict of interest occurs when source-routes rule against transit ASes' policies that are not properly incentivized for it. One prominent case is a violation of *valley-free* inter-domain paths. The current BGP's valley-free property is exercised by the economic-driven route export policies; routes learned from providers or peers are not announced to other providers or peers. However, source routing may ask for unannounced routes that would violate such property; especially, such it can happen only at ASes specified in source-routed packets. In addition, even if following valley-free paths, source-routes may override the local ASes' prioritized route decision process, i.e., asking for a different



(a) When to cache relayed CIDs? (b) When to have CID entries in the forwarding table?

Figure 51: Example: mechanisms for a transit AS to implement its policy-based packet processing.

path that is more expensive than the default route selection. Such a route request hinders ASes' business goal of maximizing their revenue. We thus proposed an accounting mechanism for ASes that provide source-routed transit service and bill the sources for traffic that conflicts with their policies. The accounting must be *verifiable* to prevent ASes from cheating (or overcharging) the sources, e.g., false claim for transit service. To this end, we rely on a hop-by-hop process of verifiable voucher collection, and our initial design can verify if source-routes ask for different paths than the ASes' default route selection. We are currently working to improve the mechanism so as to verify that a source-routed path is more expensive than the default one.

We also explored the policy issues of more practical DAGs for content retrieval, i.e., CID as primary intent, with content publisher AD and HID as fallbacks. In this case, transit ASes are given multiple options, mainly whether to serve the CID (primary intent) or to forward the CID request to the fallback. With BGP's bilateral business relationship model between ASes, we considered three mechanisms for transit ASes to implement their policy-based packet processing: (1) *When to cache?* Transit ASes can first employ a caching policy to decide whether to serve a certain CID, e.g., caching relayed contents when profitable. Our result indicates that it is beneficial for ASes to cache the contents whose publisher direction is *provider* link so as to avoid future CID requests via the provider link (See Figure 51(a)); (2) *For a cached CID, when to have its entry in the forwarding table?* Even with a cached CID, ASes can make an indirect decision on serving the CID by (not) having its entry in the table. We analyzed the cases with respect to the business relationship of the CID request ingress links and that of the CID publisher direction links. Our result shows ASes may want to have the CID entry in the forwarding tables at *customer* ingress points while the opposite is true at *provider* ingress points (See Figure 51(b)); (3) *For a CID entry in the forwarding table, when to use?* The above two mechanisms, while fast and pre-configurable, do not cover the complete cases and thus, more expensive runtime checking may be necessary. The checking is determined according to the business relationship of return-path links (i.e., serving CID) and that of fallback links. We presented the results extended based on provider/peer/customer relationship at ingress points, CID publisher directions, return-path and fallback links.

9.5 Privacy and Security with Self-Certifying Identifiers

9.5.1 Background and Goals

The policy and economic group looked at both the technical and policy implications of *self-generated* and *self-certifying* addresses, such as host IDs (HIDs), service IDs (SIDs), and content IDs (CIDs). Whereas IP addresses have traditionally been generated by the network, addresses in XIA are generated by the end device. Moreover, these addresses are derived from the public key in a public-private key pair which makes the addresses self-certifying, e.g. as long as only one end user knows the private key, it is possible to determine that all packets with that source address really do come from the same end user. This form of addressing has important advantages with respect to security, but it can greatly affect both the privacy obtainable by end users and the communications, processing, and storage overheads of routers and networks. For example, if one device keeps the same address for months or years, even as it moves from one network to another, that device can easily be tracked. On the other hand, if it is constantly changing addresses, privacy protection may be better than today's IP networks, but this may greatly decrease network efficiency and undermine some aspects of security. The goals of this work are to (i) understand the tradeoffs between privacy, security, and efficiency, (ii) devise mechanisms for XIA that might better manage these tradeoffs, and (iii) assess the broader implications for privacy policy.

9.5.2 Approach and Findings

Our initial approach is to explore a wide range of design options for a variety of important decisions related to self-generated and self-certifying addresses, including how new addresses are generated, how they are stored, how revocation can occur after compromise, whether and how the generation of new addresses can be limited, and whether and how duplicates can be detected. This step is essential both for clarifying tradeoffs and identifying aspects for which new mechanisms may be needed. To explore policy implications, we will look at which actors have the ability to make decisions that greatly affect end user privacy, and assess the extent to which there is alignment between advancing the self-interest of those actors, protecting the privacy of end users, and perhaps advancing other societal goals such as facilitating wiretapping that has been legally authorized by a court order.

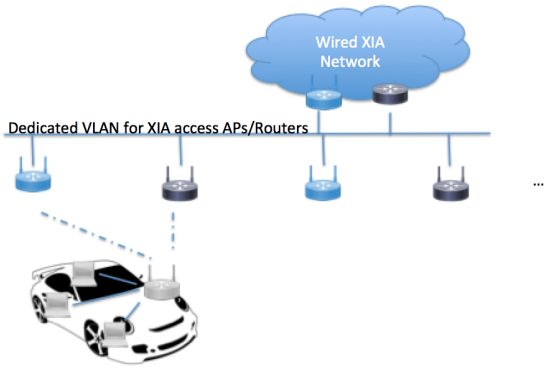


Figure 52: Architecture vehicular testbed

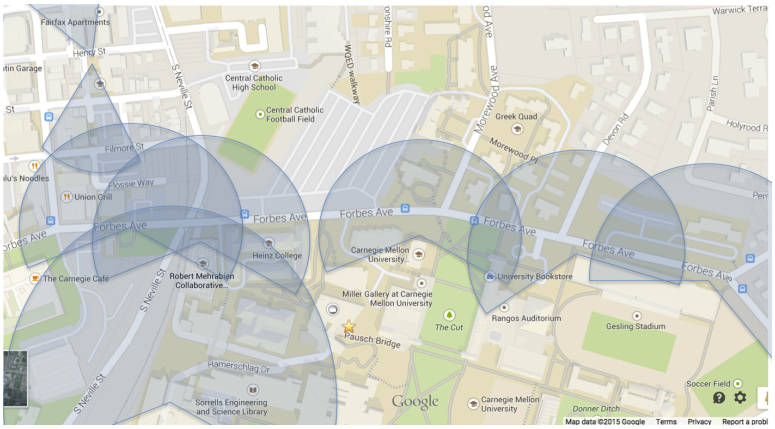


Figure 53: Map with initial RSU locations

10 Vehicular Use Case

10.1 Vehicular Testbed

Figure 52 shows the architecture of our vehicular testbed. It will consist of 6-10 DSRC base stations (typically referred to as "Road Side Units" or RSUs), and a mobile network of 4-6 vehicles. The vehicles will be equipped with a DSRC client device ("On Board Unit" or OBU) that connects to RSUs within range for Internet connectivity (Figure 52). Each OBU is connected to one or more attached client devices and routers that communicate through it, creating an on-board mobile network. The RSUs, servers, OBUs, and other on-board devices support Ethernet and DSRC as native layer-2 protocols, and will run XIP natively as the network protocol. The RSUs function as base stations and XIP routers, and the OBUs function as clients and routers.

The RSUs are being deployed at the edge of the CMU campus. They will be mounted on the roofs of CMU-owned buildings and are intended to provide coverage of the major road that passes along the campus. The locations were picked so that the geographic coverage of the RSUs will result in both intermittent / disconnected operation and overlapping multiple-provider coverage. Figure 53 shows the locations of planned and in-progress installations, along with rough sketches of the estimated coverage areas. Based on our experience with this initial testbed, additional RSUs will be installed on additional buildings.

The testbed will use the existing CMU campus network infrastructure to connect the RSUs to a set of

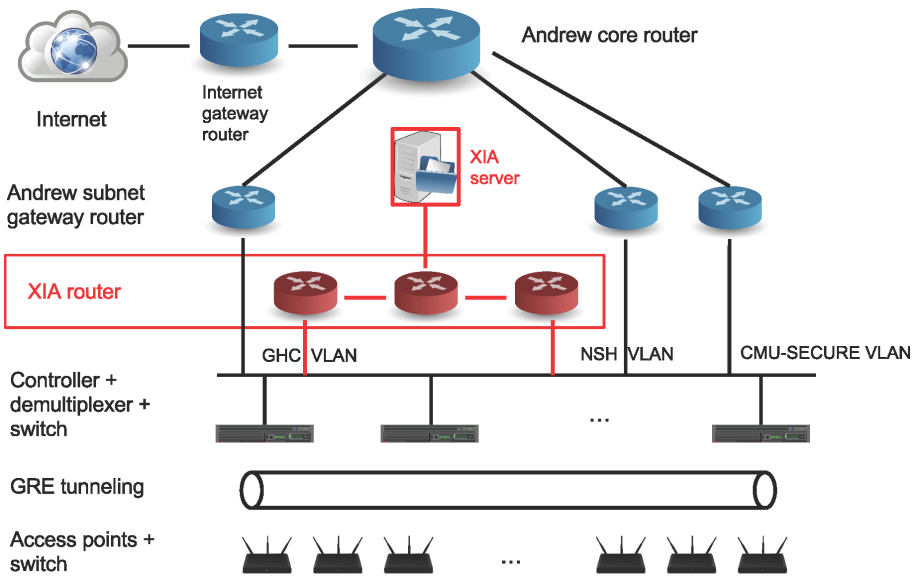


Figure 54: On-campus mobility testbed

servers and the rest of the Internet. These connections will be supported by a dedicated layer-2 Virtual LAN (VLAN), isolating testbed traffic and avoiding problems with IP-centric firewalls. Though they all share a common VLAN, the RSUs can be logically partitioned into separate layer-3 networks (indicated by different color routers in the picture) to study multi-homing, handoffs, and load balancing in XIA natively. We may augment the DSRC wireless connection with WiFi and/or cellular connections for some mixed-technology experiments.

We also deployed a small on-campus mobility testbed that we are using for low-speed mobility experiments. The testbed leverages the existing on-campus wireless network, called Wireless Andrew, which is supported across campus. Wireless Andrew uses a set of VLANs to direct all wireless traffic to a centralized Aruba controller, which supports internet connectivity with different level of security based on the Wifi SSID used by the mobile client. The wireless andrew group added an XIA VLAN which is advertised under a new set of SSID. Those VLANs isolate XIA traffic from other (IP) wireless traffic, similar to the set up described for the vehicular testbed (Figure 54). The current testbed uses two SSIDs for XIA, which we use for testing of our mobility code. Additional SSIDs can be added as needed.

10.2 Network Joining

"Network joining" means the process by which a host identifies, connects to, and makes the necessary arrangements to use, an existing network.

The traditional network joining process might look something like this: "First the end user identifies the Ethernet jack which she intends to use. A network administrator then ensures that corresponding switch port is enabled, and assigned to an appropriate VLAN based on the host's intended role. The host is then physically connected and the interface is activated. The NIC and switch perform layer-1 auto-negotiation to determine rate, duplex mode, and cable configuration. After this is complete, the host determines that the link is up and initiates DHCP. Based on the combination of VLAN and the host NIC's Ethernet address, the host is assigned an IP address and name, and configuration parameters are sent to the host. The combination of IP address and VLAN determine whether the host is allowed to access the global Internet and/or private internal network facilities."

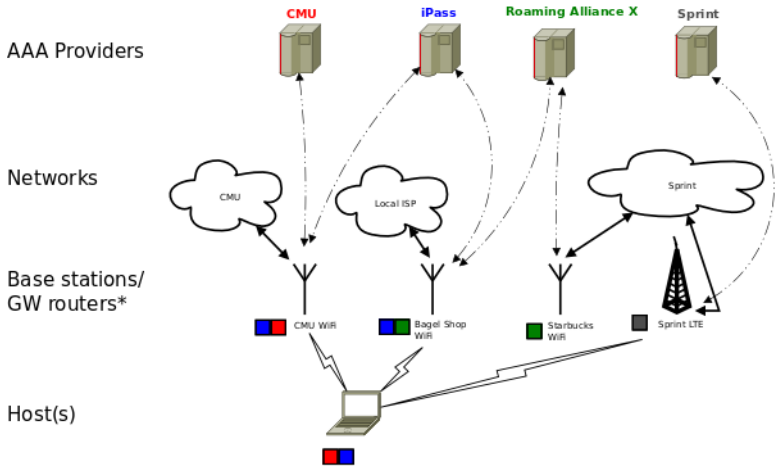


Figure 55: Mobile architecture

While this approach might be appropriate for an Ethernet in an IPv4-based enterprise network, there are several reasons why present and future networks call for a different design:

1. *The advent of public and semi-public networks,* especially wireless ones (e.g. municipal or coffee shop Wi-Fi). This implies both (a) the impracticality of involving a human administrator, and (b) limited mutual trust and familiarity between the network operator and user.

2. *The diversity of network access technologies and providers.* A contemporary device is likely to support at least one, if not multiple, WLAN and WWAN technology, and at least one wired LAN technology for larger devices. In addition to the technology diversity, there may often be many available networks with different performance, cost, and trust characteristics so devices need to make informed decisions about what network to use for each traffic flow.

3. *Small-cell mobility.* The demand for bandwidth on mobile platforms is increasing, and it seems very unlikely that ”macrocell” WWANs can provide the spectrum efficiency necessary to support future traffic loads. As a result, we are likely to see growth in the use of shorter range, low power, high rate, cells. This implies very frequent handoffs between cells.

The XIA project is defining an efficient network joining protocols that is based on an architectural model with four kinds of entity, as shown in Figure 55:

1. *Client*: A device attempting to attach to an existing network. This could be a single host, or a gateway router for a mobile network.

2. *Base station*: A set of functions supporting layer-2 and layer-3 network connectivity, including a ”gateway router” facing the client.

3. *Network*: A network infrastructure that includes at least one base station for wireless connectivity and layer-3 devices that provide access to the Internet.

4. *Provider*: The entity providing authentication, authorization, and accounting for one or more networks. The AAA provider(s) may not be directly connected to the base station.

The goals of the network joining protocol are (1) determining the service to be provided; (2) mutual authentication of the node and the network; authorization of network access, and (3) configuration of the connection. Moreover, we want the protocol to be efficient, e.g., in terms of roundtrip times. The definition and implementation of the protocol is ongoing work.

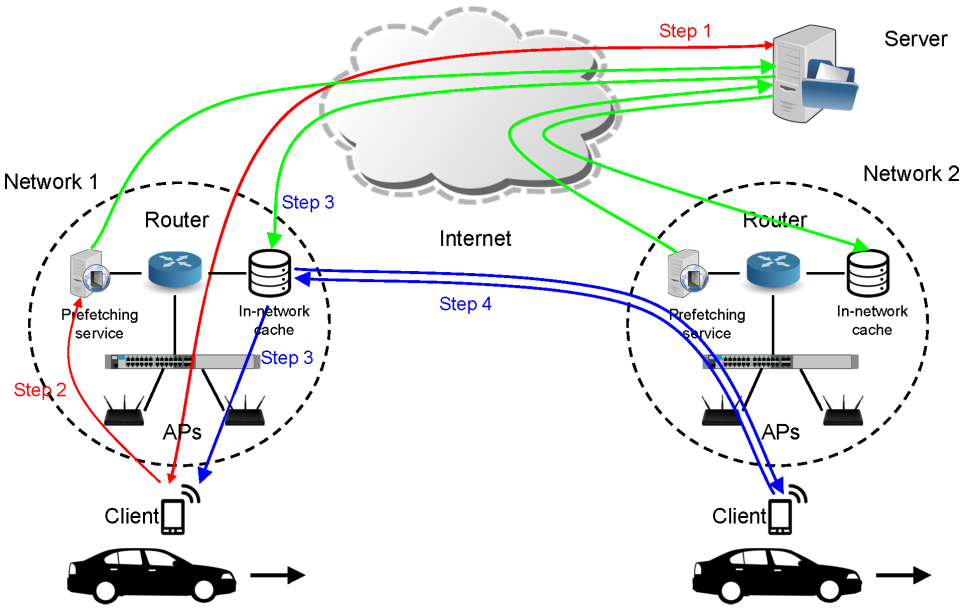


Figure 56: Vehicular content retrieval.

10.3 Leveraging caching and in-path services

In this reporting period, we explored how XIA can support mobility-based applications. This work is being performed Chenren Xu, Eric Anderson, and Peter Steenkiste.

Background - Client mobility is a fundamental challenge when accessing the current Internet, especially in vehicular networks. A first challenge is that client device may have to change its network attachment point every few seconds. Layer-2 mobility is significantly easier than layer-3 mobility, both because the change is not visible at the network layer, reducing network reconfiguration, and because authentication and authorization can be optimized. Unfortunately, it restricts the choices for connectivity since the user and device are limited to a single operators and limited set of base stations. This can reduce performance and increase cost. Another challenge is that connectivity will unavoidably be intermittent as a result of coverage gaps and occasionally slow handoffs. This does not only affect performance but may also disrupt communication sessions at the transport and application layer.

Previous work on content access in vehicular networks has focused on the current Internet [37, 30, 40]. The techniques used include the use of mobility prediction to prefetch content onto APs that are expected to be used, streamlining session establishment, separation the end-to-end path into a wired and wireless segment, and the use of caching near the edge of the network. All these solutions effectively rely on overlays, making these solutions complex and hard to deploy. In this project, we will explore how XIA data plane features to optimize bandwidth and latency at the network layer by hiding the performance impact of handoffs and disconnections as much as possible. We specifically focus on the use of CIDs and SIDs, since they provide easy and efficient access to caching and in-path services in the access network.

Approach and findings - The key idea is to download content as a sequence of chunks, identified and addressed by CIDs, instead of the traditional approach of establishing an end-to-end communication session. This has a number of potential benefits. First, it breaks long transfers, which are unlikely to complete, into a sequence of shorter ones. Second, by caching chunks as they are "in flight" between the client and the server, we can reduce the impact of disruptions due to handoffs and disconnections. For example, after a disruption, content can be transferred from the cache instead of the server as illustated in Figure 56. Finally, the cache effectively decouples the data transfer over the wireless access link from the transfer over the Internet, which

benefits both transfers by reducing the roundtrip time (wireless segment) and creating a more stable path (Internet segment). Note that we have explored some of the ideas in earlier work [37] but XIA CIDs and SIDs enable an implementation directly on the network layer, rather than based on an overlay.

10.4 Multihoming

The policy and economic group began work on technical and economic issues raised by multihoming. The research is done by PhD student Nandi Zhang, supervised by Marvin Sirbu and John Peha.

Background and Goals - One benefit of the XIA architecture is its ability to support multihoming. Enterprise networks frequently connect to multiple ISPs, both for greater performance and better resilience to failure. The need is even greater in vehicular networks, since a car may establish and then lose many cellular, Wi-Fi and DSRC connections as it travels. Giving end users fine-grained control over which carrier serves them and when may also change the nature of competition among carriers. The goal of this work is to define and evaluate protocols and algorithms for multihoming within the XIA architecture, assess whether XIA is better able to provide multihoming than today's IP networks, and explore their economic and policy implications.

Approach and Findings - We have devised a novel protocol that allows devices to migrate individual flows from one network to another. Our focus is on a mobile network in which devices in the car connect to the Internet through a mobile router, but the results should apply to individual devices. In a mobile network, the end device interacts with the car router to determine which network currently best meets the needs of an application, and moves its traffic to that network. The protocol takes advantage of XIA's self-certifying identifiers for security and DAGs for route improvements. We plan to further assess its advantages and disadvantages when compared to approaches developed for IPv4 and IPv6, and explore the broader economic and policy implications of intensifying competition among wireless providers by reducing switching costs in this way.

The use of DAGs as addresses in XIA provides immediate benefit for multihoming, as fallback can be used to expose alternative paths to an endpoint, without burdening the core routing tables or introducing additional middleboxes as occurs in today's BGP- and NAT-based multihoming solutions. DAGs feature identifier-locator separation, which facilitate mobility as well as multihoming, since a multihomed host or network often needs to shift flows between its available links. XID's self-certifying nature allows endpoints to verify the ownership of an identifier without relying on a PKI. A flow-based migration protocol allows each application to use a different service provider that best meets its needs, improving allocative efficiency. The same migration protocol can be used for flow-by-flow load balancing, unlike Mobile IP [94] and Network Mobility (NEMO) [31].

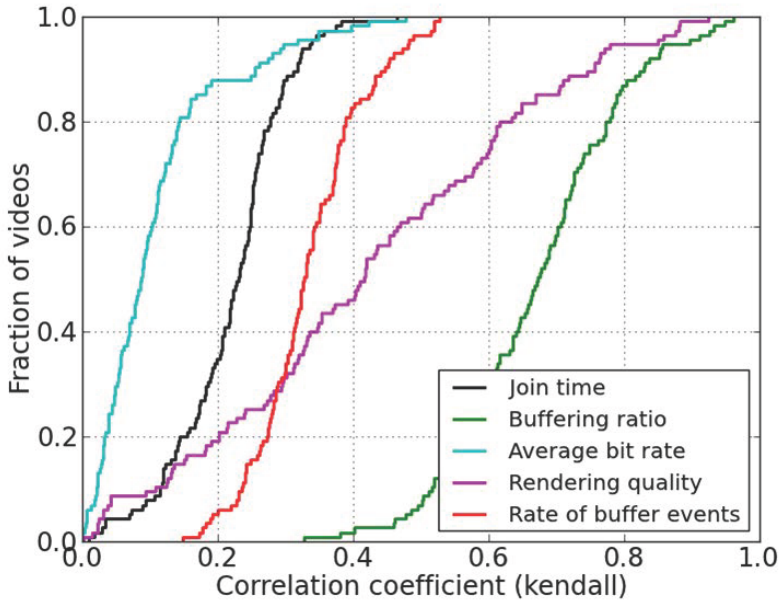


Figure 57: Impact of five quality metrics on viewer engagement

11 Video Use Case

11.1 Rate control for video streaming

This research is performed by CMU PhD student Junchen Jiang, advised by PI Zhang.

11.1.1 Background

Video traffic has grown significantly faster than the overall Internet traffic for the last five years. Already, more than 60% bytes transmitted on the Internet are video bytes. Still, we are only at the very beginning of the great migration of video consumption from traditional media (e.g., over-the-air broadcasts, cable, and satellite TV) to the Internet. In the next five years, the total amount of Internet video traffic will increase by a thousand fold, significantly dwarfing all other types of Internet traffic.

The questions of whether and how the future (and current) Internet should support video were extensively discussed and debated in the research community during the 1990's, resulting in a lot of research in network quality of service and connection-oriented video transport protocols. With the benefit of hindsight, we now have a much better understanding on why these solutions were not adopted and what key assumptions behind these solutions are wrong.

We are pursuing research in video delivery over the Internet in two directions. First, as the distribution of the video over the Internet becomes main- stream and its consumption moves from the computer to the TV screen, user expectation for high quality is constantly increasing. In this context, it is crucial to understand when and how video quality affects user engagement and how to best invest their resources to optimize video quality. We uses a unique dataset that spans different content types, including short video on demand (VoD), long VoD, and live content from popular video content providers. Using client-side instrumentation, we measure quality metrics such as the join time, buffering ratio, average bitrate, rendering quality, and rate of buffering events. A better understanding of how various aspects of video delivery impact user engagement will help identify appropriate network support.

Second, the growth in video is accompanied by a key technology trend: namely the shift from connection-oriented video transport protocols (e.g.,Windows Media, QuickTime, RTMP) to HTTP-based adaptive streaming protocols such as SmoothStreaming, HLS, HDS, and the emerging standard DASH. With a HTTP-based adaptive streaming protocol, a video player can, at the granularity of seconds, dynamically adjust the video bit rate (thus the video resolution) based on the available network bandwidth or CPU capacity. The available bit rates usually span a wide range range:from as low as 300Kbps to as high as 6Mbps with today's premium video sites. There is a tradeoff between the video quality and the network load: the higher the bit rate, the higher video quality, but also the higher load on the network. In the next few years, video traffic transported on top of HTTP streaming protocols is expected to dominate the Internet traffic. Therefore, the design of robust algorithms on top of HTTP streaming protocols is importantnot only for the performance of Internet video applications, but also the stability and performance of th overall Internet.If we were to draw an analogy to the early days of the Internet, the design of a robust TCP was critical to prevent "congestion collapse". We are potentially at a similar juncture today with respect to the design of the adaptation logic on top of HTTP streaming protocols.

11.1.2 Approach and Findings

We have quantified user engagement both at a per-video (or view) level and a per-user (or viewer) level [32]. Figure reffig:video shows for example the impact of five quality metrics on user engagement (defined as viewing time) for long videos viewed on demand. We found that the buffering ratio, the percentage of time spent in buffering, has the largest impact on the user engagement. This is true across all types of content, but the magnitude of this impact depends on the content type, with live content being the most impacted. For example, a 1% increase in buffering ratio can reduce user engagement by more than three minutes for a 90-minute live video event. We also found that the average bitrate plays a significantly more important role in the case of live content than VoD content.

We have proposed an abstract player model that is general enough to capture the characteristics of existing bitrate adaptation algorithms. This abstract model helps us identify the root cause of several undesirable interactions that arise as a consequence of overlaying the video bitrate adaptation over TCP that lead to poor efficiency, fairness, and stability. Building on these insights, we develop a set of techniques that can systematically guide the tradeoffs between stability, fairness and efficiency and thus lead to a general framework of video adaptation. We pick one concrete instance from this design space and demonstrate that it significantly outperforms all commercial players on all three key metrics across a range of experimental scenarios.

11.2 Video Quality Control

"Network aware" applications have a been active research topic for a long time. The idea is that applications can learn about the state of the network and optimizing their performance by changing how they use the network. However, given the basic functionality offered by IP and the minimal interface it offers to applications, opportunities for adaptation are limited, so a lot of research ended up focusing on network monitoring tools and simple applications like server selection. There are more potential opportunities when the network service layer is considered. For example, CDNs are now a critical component of the Internet infrastructure from the application's perspective. However, using them effectively is challenging, considering teh minimal network architectural support offered by the traditional IP architecture.

Architectures like XIA introduces new abstractions and mechanisms to enable (a) the architecture to natively support advanced service layers; (b) advanced services to optimize for specific requirements of certain applications — for example, given the unique performance requirements and the large traffic volume (growing from today's 70% Internet traffic to more than 90% in the near future), it is extremely beneficially to consider distribution services for video specifically (not for "general" content); (c) rich form of inter-layer

interactions: service-aware applications, application-aware services, network-aware services etc. Examples include not just selection of the server or CDN, but bit rate selection, path selection, etc. It may also be possible to learn about network conditions directly, e.g. from congestion control information, e.g., [52], or from network monitoring services operated by network operators, rather than having to use active probing tools.

To start exploring how we can effectively support rich network-aware applications over XIA, we decided to focus on video streaming, which accounts for a very high percentage of the traffic on the Internet. In this section we report on initial results in developing a control plane for optimizing video streaming. In the next section we look at what quality metrics at application level can be used to guide adapation and optimization at runtime for both application layers and service layers. We also present the results of a study looking at the design of in-network caches for mobile video.

Background - We continued our explorations on how to improve video quality in the Internet. This research was performed by Junchen Jiang, Vyeas Sekar, and Hui Zhang [62].

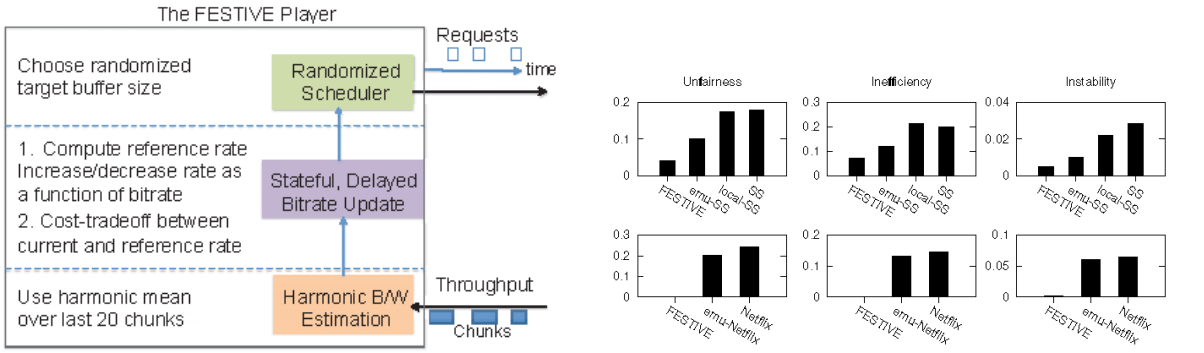
In creating Future Internet Architectures, such as XIA, we are interested in understanding what architectural support should be included to improve the quality of content delivery. To this end, we explored how different metrics affect perceived video quality [32]. We found that buffering ratio and percentage of time spent in buffering has the largest impact on user engagement. Next we expanded our focus to understand how to increase the video quality seen by users, especially for HTTP-based adaptive protocols, such as SmoothStreaming, HLS, HLS, and the emerging standard, DASH.

With a HTTP-based adaptive streaming protocol, a video player can, at the granularity of seconds, dynamically adjust the video bit rate (thus the video resolution) based on the available network bandwidth or CPU capacity. The available bit rates usually span a wide range range: from as low as 300Kbps to as high as 6Mbps with today's premium video sites. There is a tradeoff between the video quality and the network load: the higher the bit rate, the higher video quality, but also the higher load on the network. In the next few years, video traffic transported on top of HTTP streaming protocols is expected to dominate the Internet traffic. As such, a focus on robust algorithms for HTTP-streaming protocols is a must, not only for the performance of Internet video applications, but also for the stability and performance of the entire Internet.

In this work, we focus on two key questions. First, do existing streaming algorithms exhibit important properties, such as efficiency (i.e., do they use up as much bandwidth as is available to them?), fairness (i.e., do they divide available resources equally among competing clients?), and stability (i.e., do they avoid frequent shifts in bitrate, which are likely to annoy users?). These properties are similar to those provided by TCP, but similar algorithms cannot be used because HTTP-streaming algorithms reside at the application level and, as such, have access to only coarse-level information about the network. Second, what changes are necessary to to guarantee the properties stated above?

Approach and Findings - To evaluate whether existing streaming algorithms provide efficiency, fairness, and stability, we compared SmoothStreaming, Akamai, and Adobe's existing video players. We found that all failed to satisfy these properties to various extents. We then considered where to implement functionality needed to provide these properties. For example, we considered the advantages of re-architecting the transport layer for video players so as to bestow them with detailed information about the network. Ultimately, we concluded that application-level algorithms are sufficient. To demonstrate this to be the case, we created one such algorithm called FESTIVE.

We found that a key reason existing streaming algorithms do not achieve efficiency and fairness is that they cannot easily see the true network state. For example, existing algorithms may mis-estimate available network bandwidth because the periodic chunk-based download mechanisms used by many of them may unintentionally be synchronized (or anti-synchronized with other competing traffic. To alleviate this problem, we designed FESTIVE's periodic chunk-based download mechanism with added jitter. But, even with added jitter, an unfair initial allocation of bandwidth can lead to prolonged unfairness. For example, a video player with an initially high bandwidth allocation may continue to see high available bandwidth because its



(a) Architecture

(b) Comparison to other video players. Details about the metrics can be found in Jiang et al. [62]

Figure 58: Overview and evaluation of FESTIVE

wire occupancy is higher than other competing players. To help, we designed FESTIVE with a mechanism that aggressively ramps up the allocation of video players that have low bandwidth allocations. To provide stability, we introduced a control parameter that delays bitrate changes if too many have been observed in a recent time window. In our paper [62], we provide proofs showing that FESTIVE guarantees the desired properties. Figure 58(a) shows a diagram of a FESTIVE-based video player.

We compared our FESTIVE-based video player to existing video players. We found that a set of FESTIVE-based players suffer from less unfairness, inefficiency, and instability compared to other video players (see Figure 58(b)). Though this is a promising start, we do not expect all video players on the web to use FESTIVE. For future work, we plan to investigate whether efficiency, fairness, and stability can be maintained in the face of heterogeneity. We also plan to investigate how specific network support can help improving the optimization process.

11.3 Video Control Plane

We designed a control plane for video. The results of this research will feed into the video distribution network deployment that is part of the XIA follow on project (FIA XP program).

Background - The motivation of video control plane is the observation that today's video delivery system faces many choices on each control knob, and there is a huge diversity in both time and space, among the performance of different choices. In this context, we identify that many of today's video quality issues are caused by the absence of centralized intelligence. For instance, client-driven CDN selection is often suboptimal, due to the lack of knowledge pertaining to which CDN performs the best in different areas.

Video control plane is a centralized control layer on top of the existing video delivery system, as shown in Figure 59. The vision of a video control plane is to improve video quality by making real-time decisions for different components on the path from video origin to client screen. It can perform control over different components, such as on client-side players, ISPs, and/or CDNs. To make optimal decisions, the control plane also has to actively monitor the real-time performance of the delivery system. For instance, since our prototype has the ability to instrument client-side video players, our performance monitoring collects updates of real-time quality information from the clients, including standard quality metrics (such as buffering ratio), and session tags (such as ASN and player name). In other words, a key design aspect of the video control plane is the definition of interfaces between the various actors supporting both information exchange and control.

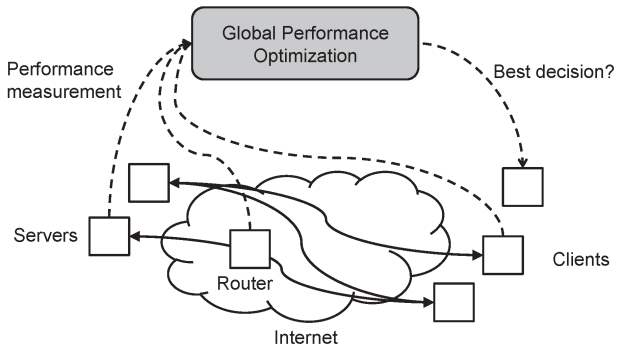


Figure 59: Global video control plane

The core technique of such a video control plane is to transform the huge amount of measurements into decisions. One approach is that, by data mining the measurements, the control plane can identify useful knowledge of the current states, such as potential congestion, hot spots, and failures, and make decisions to avoid these bad cases. Alternatively, the control plane can model the problem as blackbox and take a data-driven approach – making the decisions that appear to be the best, with little reasoning. To make the transform from updates to decision efficiently in real-time, we have to leverage the emerging big data platforms, and specifically, the streaming data processing tools, such as the Spark/Hadoop ecosystem.

We now present our research finding in five related areas. Several of the research efforts leveraged traces of client-side video quality provided by Conviva for experiments. Conviva also provided platforms for testing and deployment of core decision making algorithms.

Bottleneck inference of video quality issues - This project aims to understand the nature of video quality problems, and see what components have to be improved and if simple approaches can yield comparable benefits. The work is a first attempt to shed some light on the structure of video quality issues.

Using the measurements from 300 million video sessions over a two-week period, we identify recurrent problems using a hierarchical clustering approach over the space of client/session attributes (e.g., CDN, AS, connectivity). Our key findings are (1) a small number (2%) of critical clusters account for 83% of join failure problems (44–84% for other metrics); (2) many problem events (50%) persist for at least 2 hours; (3) a majority of these problems (e.g., 60% of join failures, 30–60% for other metrics) are related to content provider, CDN, or client ISP issues. Building on these insights, we evaluate the potential improvement by focusing on addressing these recurrent problems and find that fixing just 1% of these clusters can reduce the number of problematic sessions by 55% for join failures (15%–40% for other metrics).

Core algorithms for decision making - This project focuses on the core algorithms that make decisions for client-side players based on massive updates from the clients. In designing the algorithm, we make two arguments. First, we argue that being able to accurately predict the outcomes of the available choices (e.g., would a streaming video client be able to sustain a particular bitrate?) can greatly help to achieve better quality. Second, we argue that to accurately predict the outcome of a given choice, we need to leverage the information available from other sessions, streams or connections.

Our evaluation focuses on the quality prediction algorithms and how they impact the resulting quality improvement. We use the trace of client session quality of 800 sessions over a duration of a month. Our key findings are: our prediction algorithm outperforms baseline algorithms by 30% to 50%, including averaging over a fixed spatial granular (e.g., by ASN) or over a fixed temporal granular (e.g., last 1 hour) as prediction. We also find our algorithm achieves noticeable improvement on four metrics, e.g., for buffering ratio, by 1.5% globally and as much as 4.8% for some popular site.

A cooperative architecture for application quality control - There is a growing recognition among researchers, industry practitioners, and service providers of the need to optimize user-perceived applica-

tion experience, including video quality experience. However, network infrastructure owners (i.e., ISPs) have traditionally been left out of this equation. In parallel, application providers have to deploy complex workarounds that reverse engineer the networks impact on application-level metrics. This project makes a case for EONA, a new network paradigm where application providers and network providers can collaborate meaningfully to improve application experience. To this end, EONA introduces two new *information sharing* interfaces between the application providers and network providers.

So far in this project, we have focused on the use cases of EONA and recipe for its interface design. In the next stage, we will focus on using EONA to improve video quality.

System design of video control plane - This topic focuses on designing the system of a video control plane, called C3, that is highly scalable, available, and resilient. As a centralized control platform, there are natural parallels between C3 and corresponding efforts in the SDN literature. Moreover, we have to achieve more strict requirements than SDN along three aspects: finer-grained monitoring and decision making, handling client-side diversity, and larger scale. At the same time, C3 has more tolerance for responsiveness and consistency than SDN because of the "application-level resilience" of video streaming. C3 has been built and operated for many years by Conviva, and we have learned a lot from their deployment experience and evolution.

In designing C3 to meet the requirements, we make two key choices, which turn out to be critical. First, we make an explicit choice to make the *client-side functionality minimal*. This decision dramatically simplifies deployment for new content providers, accelerates testing and integration, and also enables independent evolution of both client-side platforms and the control plane logic. Second, we consciously exploit *application-level resilience*. For instance, we can tradeoff a small increase in the staleness of the control decisions for significantly improved scalability and resilience.

Hybrid control for CDNs - This effort has focused on the design of a control plane, called VDN, that achieves quality at scale by enabling dynamic fine-grained control over live video delivery. Specifically, VDN has three key goals:

- **Quality:** It must optimize for application-specific quality metrics (e.g., bitrate, buffering ratio, etc.). Note that more throughput does not always mean higher bitrate and may even reduce other metrics. We would like this use the results of our work on deriving a video QoE metric.
- **Scalability:** It must support the workload of today's largest CDNs by scaling to thousands of live video streams and millions of users, using thousands of CDN distribution clusters.
- **Real-time response:** It must respond to user- and network-induced events with low latency (less than a second), while provisioning for individual videos on-the-fly.

We found that to meet the quality requirements of the VDN design, we needed to leverage SDN-like traffic engineering to optimize the delivery topology and avoid adverse interactions between different video delivery streams. Based on this observation, we created a central controller that uses network measurements to compute a distribution tree that satisfies user demands and optimizes the CDNs resource costs.

However, we found that a fully centralized design did not meet the real-time response requirements. A purely centralized approach cannot react to small-scale, real-time variations that occur very fast. The critical path in central decision making has a high latency. In contrast, a distributed control systems would allow each edge cluster to monitor and react to small scale changes. As a result, we decided to create a *hybrid* control plane. The central controller provides guidance that optimizes the overall system behavior. For example, it informs each node how much of the incoming video it should retrieve from different sources. However, VDN allows each node to dynamically selects the upstream clusters that it wishes to use at any moment. If the downlink bandwidth from upstream node X slightly decreases, it can temporarily direct more requests to Y while adhering to the global optimization decision (e.g., the 80% to 20% split between X and Y) over a slightly longer period of time.

VDN's local control complements the global decision making even under a failure. For example, if a delivery link between two nodes fails, central control takes a long time to generate a new set of decisions (i.e., delivery trees). VDN allows a node to override the global decision by choosing not to support higher bitrates and/or find another source of the video stream that currently has the highest remaining capacity.

We have begun to evaluate our system against a trace of real video sessions from Conviva. Our initial results show that this design should support a CDN with 1,000 distribution clusters, and allow us to react to network and viewership changes at a timescale of milliseconds.

11.4 Weak cooperation among application and infrastructure entities

We have also started to explore the design of richer interfaces to enable cooperation among application-level entities, which are responsible for creating and delivering content on the Internet. This work is being performed by Matt Mukerjee, Raja Sambasivan, Ilker Nadi Bozkurt, Bruce Maggs, Srini Seshan, Hui Zhang, and Peter Steenkiste.

Background - As described previously, the Internet is composed of application-level entities, which create and distribute content (e.g., video on demand (VoD), live video, images, large datasets), and infrastructure-level entities, which route traffic from sources to destinations. Despite the fact that application-level entities are responsible for originating a large fraction of the Internet's traffic, they do not cooperate among each other. While the lack of coordination among brokers, and among CDNs, should not be a surprise, brokers and CDNs do not coordinate either, even though they solve similar and closely coupled resource management problems, as discussed in the C3 and VDN findings. This negatively impacts clients' quality of experience and results in lost revenue [28].

One example of lack of cooperation is evidenced by the fact that both CDNs and content brokers independently deploy extensive measurement infrastructures to identify the best CDN cluster for a client. However, because both have slightly different objectives and constraints and because the visibility provided by their infrastructures are different, they may end up making different choices. A broker may choose a CDN expecting it will use a certain cluster, but the CDN may choose to use a cluster that results in worse quality of experiences because of internal constraints, such as cost or current load, or because its measurement infrastructure predicted a different cluster than the brokers' infrastructure.

In this work, we aim to identify interfaces that enable application-level entities to cooperate while also keeping sensitive information secret and how they can effectively use XIA network control interfaces for resource management.

Initial approach & drivers for a design - As a first step, we have identified categories of problems that application-level entities involved in video on demand (VoD) experience due to lack of cooperation; these are in addition to the networking related issues identified in the network control plane section. These include:

1. *Unintended use of expensive clusters*: Content brokers try to pick the CDN (e.g., Akamai or level 3) that will deliver the best performance for a client. They do so without knowledge of the CDNs' costs for using its clusters (e.g., due to transit pricing). In certain situations, this might result in brokers only picking a CDN for clients that are expensive for it to serve.

2. *Mismatches in predictions* As described in the example above, a broker might choose a CDN based on performance predictions enabled by its measurement infrastructure. However, CDNs may choose to use a completely different cluster because internal constraints, such as cost, or because the measurement infrastructures CDNs use have different granularity data than brokers' infrastructures. For example, CDNs typically only have visibility to the DNS server a client uses, whereas brokers have visibility into the application itself.

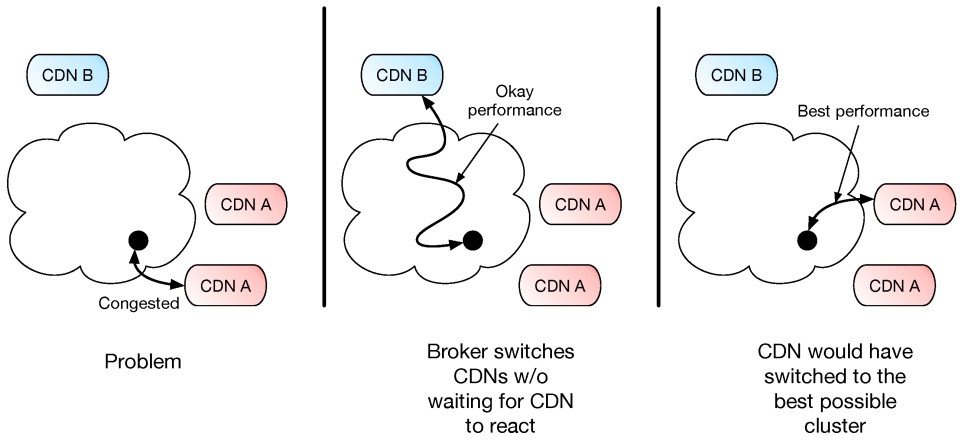


Figure 60: Example case where conflicting timescales of decision making can yield suboptimal results

3. *Conflicting time-scales of decision making*: A broker may decide to switch CDNs because of poor performance. However, if left to its own means, the original CDN might eventually have decided to switch clusters by itself. The new cluster chosen by the original CDN may exhibit better performance than the new cluster chosen by the broker (see Figure 60).

Based on the above categories, we have identified a sytem design that allows brokers to query CDNs, asking them for a short list of clusters they will likely use for a client. This list would be annotated with a probability that encodes the CDNs' likelihood of using the stated clusters. This probability could encode CDNs private constraints and goals (i.e., cost of using a cluster) w/o explicitly revealing them to brokers. Brokers could combine the query results with their independent performance measurements to choose CDNs. An accounting mechanism could be used to verify that CDNs report probability accurately. We believe this design allows brokers and CDNs to retain their existing autonomy while avoiding the problems described above. Note that ths approach to coordination is similar in style to the proposed network control interface, which would allow networks to expose multiple paths to edge networks, annotated with information related to resource management.

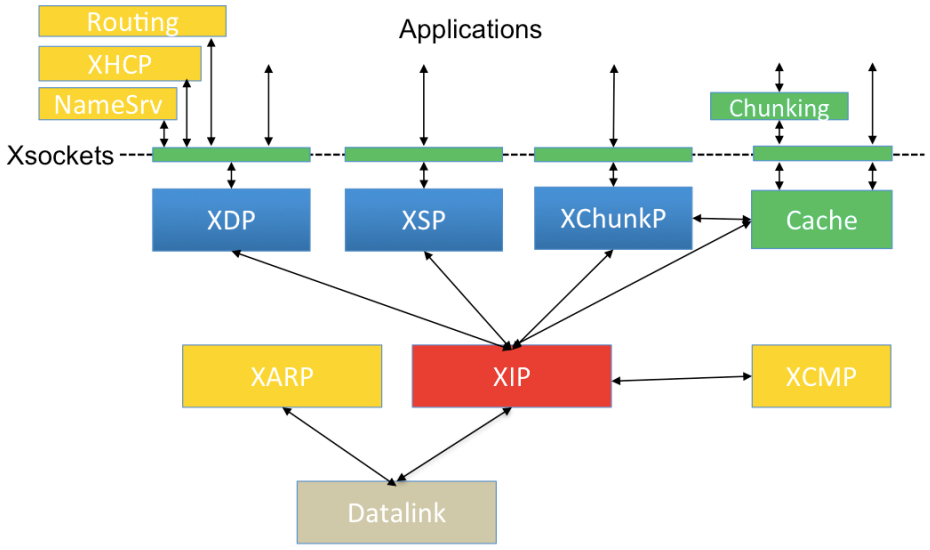


Figure 61: XIA Protocol Stack

12 XIA Prototypes

We have implemented two complete prototype networks based on the XIA architecture, one using Click and a native Linux implementation. We also have a prototype of Scion that is integrated in XIA but can also run as overlay over IP.

12.1 XIA Prototype

Our first prototype is based on Click [73] and includes all components needed to run a complete XIP-based network. It includes the XIA network protocol stack, socket API for XIA, and network bootstrap/support services (e.g., routing, initial host configuration, name service). The XIA prototype modules are shown in Figure 61. We also provide instructions and scripts for running experiments on GENI and in virtual machine environments. The XIA prototype is available as Apache-licensed open source on GitHub (http://www.github.com/xia-project/xia-core). All the related documentation including the information on the building and using the XIA prototype (e.g., how to run XIA on local Linux machines or over the GENI testbeds, a collection of simple C and Python programs using the XIA APIs, running a sample web demo, and a walkthrough on creating new XIA applications, etc) is is available on the XIA wiki: `http://www.xia.cs.cmu.edu/wiki`.

The central component of the prototype is the XIA forwarding engine (i.e., XIP module in the figure) which we described in Section 1.3.4. On top of XIP, we implemented three different types of transport protocols: XDP, XSP, and XChunkP. XDP (similar to UDP) is unreliable connectionless protocol, while XSP (similar to TCP) supports reliable connection-oriented communication. XChunkP is a reliable chunk-delivery protocol for content retrieval in the XIA network, i.e. it transfers chuncks between caches and clients. All three transport protocols are fairly basic. We plan to replace them with optimized versions later in the project.

The XIA prototype also facilitates content retrieval by allowing the content requests to be served by any intermediate routers (or the original publisher) who hold a copy of the content. In the prototype, content cache module is responsible for such in-network caching at routers and it is also used for publishing and serving contents at end-hosts. The content cache also performs chunk fragmentation (i.e., a single content

chunk may be divided into multiple packets, each of which is individually delivered via routers to the requesting host; those packets share the same CID information). If a packetized content chunk is relayed via routers, each packet of the chunk is partially stored in the cache, and reassembled into the original chunk once all the packets are passed through the router.

XIA Sockets (Xsockets) are designed to be as similar as possible to the standard socket API to make porting code as easy as possible. The Xsocket API includes calls for both unreliable datagram and reliable streaming based communication; These calls are very similar to the calls used over the current Internet; the primary difference is that they use a different address type, XIA instead of INET. We also have calls for retrieving contents chunks using the CID principal type and for creating chunks and making them available through the chunk cache. We have also develop simple example applications, such as a web proxy, to illustrate the Xsocket API.

In addition, the XIP protocol stack has two supporting modules: XARP and XCMP. XARP is the XIA version of address resolution protocol, used for resolving target XIDs into link layer addresses (e.g., mac addresses). Its implementation is very similar to that of ARP. XCMP is XIA's version of ICMP. It can be used for error reporting and for very basic diagnostic functions. It is basis for XIA's version of ping and traceroute.

In order to allow users to run larger scale experiments over diverse network topologies, the prototype XIA network also provides network-wide bootstrap functions including host configuration, name resolution, and routing. These functions are configured automatically when the network topology is created, minimizing the configuration effort required from users. Host configuration is supported using XHCP. It is similar to DHCP, although there are some differences, e.g. in contrast to the current Internet, XIA hosts generate their own HID. Name resolution is based on a centralized name server. The released version is a custom implementation that translates hierarchical names into DAGs. We have also modified the BIND DNS implementation to support the XIA address type; we plan to include it in the next release.

Since its initial release in 2012, we have added a lot of functionality to the prototype, include SDN based intra-domain routing for HIDs, CIDs, and SIDs, inter-domain routing for NIDs and SID anycast, mobility support, AIP style checking of source-addresses, external caching support for scalability, reliable chunk transfers, a compatibility library for sockets (described below), and multi-homing support.

12.2 Native Linux Implementation

We have also built a native XIA stack for Linux that will simplify use by early adopters (Section 4). It is also available on GitHub (repo: `XIA-for-Linux`). The native implementation of the XIA protocol stack in Linux provides a full implementation and integration of the BSD socket interface running with XIP as the internetwork layer protocol, as well as forwarding and routing support for host, content, and autonomous domain principals (HIDs, CIDs, and ADs respectively), and basic UDP-style transport furnished by the XDP transport layer protocol. We have also added support for new principal types to support service discovery using Serval [92] (part of the Nebula project) and multicast based on Pursuit. We also implemented an XIA version of Wireshark that works on both the native Linux and Click-based prototypes.

The current codebase sports also state-of-the-art technologies such as LXC (a lightweight form of virtualization), and RCU (a lockless synchronization mechanism), as well as innovation with XIA principals implemented as loadable kernel modules (LKMs). To the best of our knowledge, Linux XIA is the first network stack whose binary image that can be extended or reduced, on-the-fly, according to the needs of network administrators.

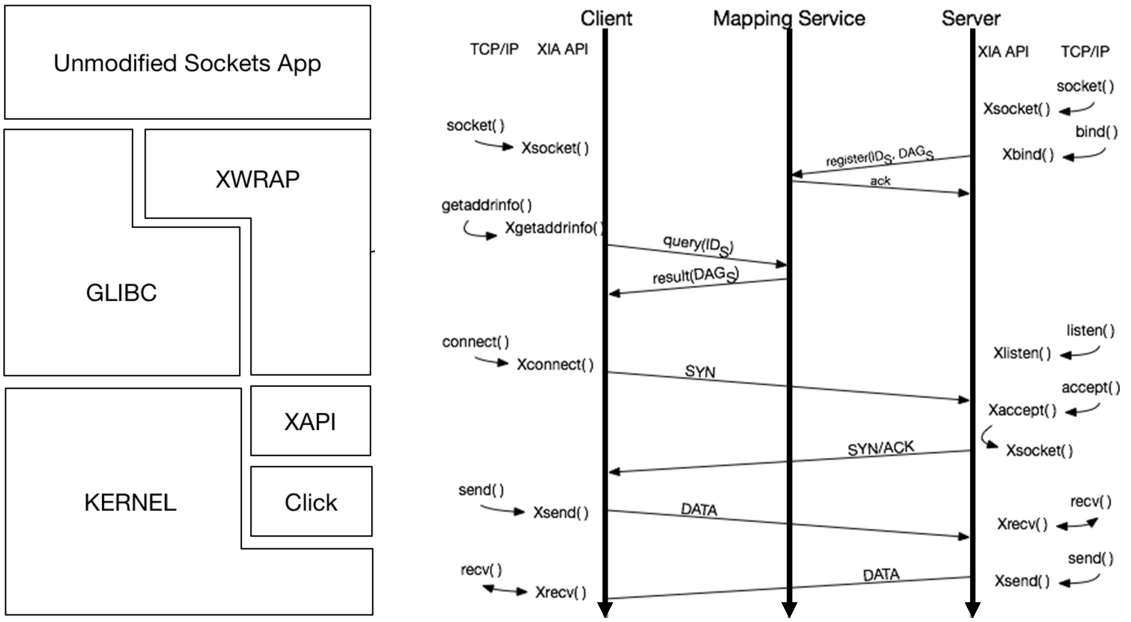


Figure 62: Compatibility library concept (left) and example (right).

12.3 Scion Prototype

We have also several prototypes of SCION (Section 1.7). The first implementation is based on Click and includes all the modules needed to bootstrap and select paths in a operational network: a protocol for path discovery, a trust management infrastructure, a path server, path selection in client networks, and path-based forwarding mechanisms. In order to decouple the development of the XIA dataplane and SCION prototypes, we chose to target the first SCION implementation as an overlay of today's Internet.

We also developed a high-speed in-kernel SCION router implementation with fast packet processing and cryptographic operations. We also ran experiments across multiple international sites connected using IP tunnels.

Finally, we have re-implemented the SCION prototype in Python, sacrificing performance for dramatically improved code readability. The goal is to enable other research teams to pick up the SCION code and rapidly understand and extend it to add additional functionality. Similar to the original implementation, it include all the modules to bootstrap a network and select paths in a operational network: a protocol for path discovery, a trust management infrastructure, a path server, path selection in client networks, and path-based forwarding mechanisms. The Python re-implementation allowed us to improve the robustness of the server infrastructure.

12.4 Compatibiity Library

We report on our effort to simplify porting IP applications to XIA networks. This work was done by Daniel Barrett, Nitin Gupta, and Peter Steenkiste at CMU.

Background and Goals - An important goal of the XIA project is to demonstrate XIA's viability as a network architecture by having several network deployments. However, such deployments will only contribute to the research agenda if they are used, which requires useful applications, i.e., applications people are interested in using. While writing XIA applications is fairly intuitive, developing practical applications such as web servers, browsers, video players, etc. is very time consuming and has very little, if any, research value, so it is important to minimize how much student time (and research funding) is spent on this.

Early on in the project, we decide to explore the use of a *compatibility library* to reduce the effort of porting existing IP/socket applications to XIA, as described in earlier reports. This approach was motivated by the design of the Xsocket interface, which is itself based on the design of the XIA protocol stack (Figure 61). The Xsocket library has two classes of calls. A first class consists of calls that manage sockets and that support UDP and TCP-style (XDP and XSP in XIA) communication. For these calls, there are IP socket calls that are more or less equivalent. The second class calls relate the use of CIDs (left side of the figure), which support a fundamentally different communication paradigm. When using TCP and UDP like protocols, communication is initiated by the sender, which sends (i.e., pushes) data, but with CIDs, communication is initiated by the receiver, which gets (i.e., pulls) data. There is no equivalent for CID-based communication in IP.

This suggest a two step approach to porting IP applciations to XIA. First, any IP socket calls in the application are translated automatically into the equivalent Xsocket calls. Figure 62(left) illustrates this: the compatibility library (XWRAP) intercepts and maps networking related socket calls to Xsocket calls, while calls related to file IO are mapped to GLIBC. In a second optimization step, applications that can benefit from using CIDs for specific communication sessions need to be modified by a programmer. Specifically, any calls for the communication sessions of interest must be replace XIA call used for CIDs.

We implemented this approach and our experience with simple applications was fairly positive. While some manual changes are necessary since several socket calls expose the IP address structure, common send and receive calls could be translated automatically. However, porting large applications turned out to still be very time consuming for two reasons. First, industrial strength application use a large very number of different functions for communication (well over a hundred), many of which are hard to map. Second, those applications also tend to use a lot of sockets, so a lot of manual changes are needed to replace any code that exposes the IP address format. We decided that while our general approach was sound, we needed a different type of compatibility library.

Approach and Findings - The new compatibility library is based on a simple observation: the vast majority of applications do not care about the address they use for communication, i.e., they simply use the address that is returned by the name server. This leads to the following simple design: in order to be able to run many, if not all, IP applications unmodified, we allow them to continue to use IP addresses. However, we do not use the IP addresses as network addresses when running over XIA. Instead we treat them as a 32 bit value that is combined with the IP port number to form an endpoint identifier, which uniquely identifies the socket. This endpoint identifier is then transparently translated into an XIA address (which unique identifies the corresponding Xsocket) by the compatibility library as needed. This mapping is somewhat tricky since it must be done consistently throughout the network. This is easy for client side addresses, since they are generated and used only by the client device. Addresses for services, however, are generated by the service and used by clients, making a consistent mapping harder. Logically, one can think of the compatibility library as acting as a NAT for each device, translating between its (IP address, port number) and an XIA address. All network communication is based on XIP.

Let us use the example in Figure 62(right) to explain how the library works. The figure shows an IP client (left) contacting an IP service (right). Starting at the left-top, the server creates a socket and binds it to its IP address. These calls are translated into the equivalent XIA calls, but in addition, the bind() call creates a DAG for the service and register the binding between the server's (IP address, port number) pair and XIA DAG with a mapping service (middle of the figure). This service is not part of the XIA architecture but is a utility that helps legacy applications run natively over XIA. The service then registers its IP "address" with DNS and executes a listen call, waiting for incoming connection requests.

A client that wants to communicate with the service first creates a sockets, which is translated to an XIA call, and then does a name lookup with DNS. DNS will return the IP address for the service, which getaddrinfo() translates into and XIA address by contacting the mapping service. The client then calls the connect call, which is translated into two separate calls. First it calls Xconnect to establish the XIA

connection. Second, it allocates an ephemeral port number which it associates with the socket and also uses to create an endpoint identifier. It also enters the translation between the endpoint identifier and the XIA address associated with the Xsocket in a local mapping database. When the server accepts the connection request and creates a new socket, it similarly adds its mapping to a local mapping database. From then on, both the client and server can use their local mapping database to identify the Xsocket associated with the socket used in any communication related calls, such as send, receiver, and closing connections.

This version of the compatibility library has a number of advantages. First, it should allow us to run most IP applications over XIA with no modifications. Second, these applications will automatically get many of the benefits offered by XIA, e.g., intrinsic security, anycast support for replicated services (SIDs), mobility support, etc. Third, applications can freely mix connections that use IP sockets and the compatibility library, and native XIA Xsockets (e.g., for using CIDs). Applications may want to use native Xsockets for specific connections for a number of reasons. For example, they may want to use CIDs, or they may want to inspect or modify the DAG that is used as the address for the source or destination, e.g., to hide CIDs for privacy reasons or to insert SID for in-path services.

Status and future work - We have implemented this new compatibility library design and have successfully used it for several applications without needing to make any changes to the code. The largest applications so far are Firefox and Linphone. Firefox is a industrial strength web browser. Linphone is video conferencing tool in which each party acts both as a client and a server. We will continue to port applications, which may require us to add support for more calls (e.g., to support fork). We will also develop support for applications that want to manipulate DAGs or use CIDs.

References

[1] University of Oregon Route Views Project. http://www.routeviews.org.

[2] J. Abley, Afilias Canada, and K. Lindqvist. Rfc 4786 - operation of anycast services, Dec 2006.

[3] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Edward A. Fox, and Stephen Williams. Removal policies in network caches for world-wide web documents. In *Proc. SIGCOMM*, 1996.

[4] A. Acquisti, L. Brandimarte, and G. Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221), 2015.

[5] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. `http://named-data.net/techreport/TR005-ndnsim.pdf`, 2012.

[6] P. Agyapong and M. Sirbu. The economic implications of edge-directed routing: A network operator's perspective. In *4th International Conference on the Evolving Internet*, Venice, Italy, June 24-29 2012.

[7] Patrick Agyapong and Marvin Sirbu. Economic Incentives in Content-Centric Networking: Implications for Protocol Design and Public Policy. In *Proceedings of the 39th Research Conference on Communications, Information and Internet Policy (TPRC 2011)*, Arlington, VA, September 2011.

[8] Patrick Agyapong and Marvin Sirbu. Economic Incentives in Content-Centric Networking: Implications for Protocol Design and Public Policy. *Submitted to IEEE Communications Magazine Special Issue on CCNs*, December 2012.

[9] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David Andersen, John Byers, Srinivasan Seshan, and Peter Steenkiste. Xia: An architecture for an evolvable and trustworthy internet. In *The Tenth ACM Workshop on Hot Topics in Networks (HotNets-X)*, Cambridge, MA, November 2011. ACM.

[10] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David Andersen, John Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: An Architecture for an Evolvable and Trustworthy Internet, February 2011. Technical Report CMU-CS-11-100, Department of Computer Science, Carnegie Mellon University.

[11] David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable Internet Protocol (AIP). In *Proc. ACM SIGCOMM*, Seattle, WA, August 2008.

[12] Katerina Argyraki and David R. Cheriton. Network capabilities: The good, the bad and the ugly. In *Proceedings of ACM HotNets*, 2005.

[13] A. Bakre and B. R. Badrinath. Implementation and Performance Evaluation of Indirect-TCP. *IEEE Trans. on Computers*, 46(3), March 1997.

[14] Athula Balachandran, Vyas Sekar, Aditya Akella, and Srinivasan Seshan. Analyzing the potential benefits of cdn augmentation strategies for internet video workloads. In *Proc. IMC*, 2013.

[15] David Basin, Cas Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. APKI: Attack Resilient Public-Key Infrastructure. In *Proceedings of the ACM Conference on Computer and Communications Security*, November 2014.

[16] L Bisti, E Mingozzi, and G Stea. Interdomain path computation for PCE-assisted Traffic Engineering. In *Proceedings of the 2009 International Conference on Broadband Communications, Networks, and Systems*, pages 1–9, 2009.

[17] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. In *RFC 2475*, December 1998.

[18] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations, 2001.

[19] Peter Bright. Can a DDoS break the Internet? `http://arstechnica.com/security/2013/04/can-a-ddos-break-the-internet-sure-just-not-all-of-it/`, 2013.

[20] Bob Briscoe, Arnaud Jacquet, Toby Moncaster, and Alan Smith. Re-ecn: The motivation for adding congestion accountability to tcp/ip. In *Internet Engineering Task Force*, October 2010.

[21] CAIDA. skitter. `http://www.caida.org/tools/measurement/skitter/`.

[22] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. In *Proc. IFIP Networking*, 2012.

[23] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, July 2003.

[24] Cisco. Cisco forecast. `http://goo.gl/hHzW4`.

[25] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms. In *Proc. ACM SIGCOMM*, Baltimore, MD, August 1992.

[26] David D. Clark, Craig Partridge, Robert T. Braden, Bruce Davie, Sally Floyd, Van Jacobson, Dina Katabi, Greg Minshall, K. K. Ramakrishnan, Timothy Roscoe, Ion Stoica, John Wroclawski, and Lixia Zhang. Making the world (of communications) a different place. *SIGCOMM Comput. Commun. Rev.*, 35:91–96, July 2005.

[27] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's internet. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 347–356, New York, NY, USA, 2002. ACM.

[28] Conviva. Conviva Viewer Experience Report, February 2013.

[29] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. In *ACM SIGCOMM Computer Communication Review*, July 1998.

[30] Pralhad Deshpande, Anand Kashyap, Chul Sung, and Samir R Das. Predictive methods for improved vehicular wifi access. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 263–276. ACM, 2009.

[31] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Rfc 3963 - network mobility (nemo) basic support protocol, Jan 2005.

[32] Florin Dobrian, Asad Awan, Dilip Joesph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, and Hui Zhang. Understanding the impact of video quality on user engagement. In *SIGCOMM*, August 2011.

[33] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proc. SIGCOMM*, 2011.

[34] Fahad Dogar. *Architecting for Diversity at the Edge: Supporting Rich Network Services over an Unbundled Transport*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, May 2012.

[35] Fahad Dogar and Peter Steenkiste. M2: Using Visible Middleboxes to serve Pro-Active Mobile-Hosts. In *The Third International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'08)*, Seattle, August 2008. collocated with ACM SIGCOMM'08.

[36] Fahad Dogar, Peter Steenkiste, and Konstantina Papagiannaki. Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices. In *The 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys 2010)*, San Francisco, CA, June 2010. ACM.

[37] Fahad R. Dogar and Peter Steenkiste. Architecting for edge diversity: supporting rich services over an unbundled transport. In *CoNEXT '12*, pages 13–24, New York, NY, USA, 2012. ACM.

[38] Nandita Dukkipati. Rate control protocol (rcp): Congestion control to make flows complete quickly. In *Ph.D. Thesis, Department of Electrical Engineering, Stanford University*, October 2007.

[39] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: vehicular content delivery using wifi. In *MobiCom '08*, pages 199–210, New York, NY, USA, 2008. ACM.

[40] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *14th ACM MOBICOM*, San Francisco, CA, September 2008.

[41] Suyong Eum, Kiyohide Nakauchi, Masayuki Murata, Yozo Shoji, and Nozomu Nishinaga. Catt: Potential based routing with content caching for icn. In *Proc. ICN*, 2012.

[42] Bin Fan, David G. Andersen, and Michael Kaminsky. MemC3: Compact and concurrent memcache with dumber caching and smarter hashing. In *Proc. 10th USENIX NSDI*, Lombard, IL, April 2013.

[43] Bin Fan, David G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than Bloom. In *Proceedings of CoNext 2014*, December 2014.

[44] Bin Fan, Dong Zhou, Hyeontaek Lim, Michael Kaminsky, and David G. Andersen. When cycles are cheap, some tables can be huge. In *Proc. HotOS XIV*, Santa Ana Pueblo, NM, May 2013.

[45] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, K.C. Ng, Vyas Sekar, and Scott Shenker. Less pain, most of the gain: Incrementally deployable ICN. In *Proc. SIGCOMM*, 2013.

[46] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, August 2000.

[47] Bryan Ford and Janardhan Iyengar. Breaking up the transport logjam. In *ACM Hotnets*, New York, NY, USA, 2008. ACM.

[48] Alexander Gladisch, Robil Daher, and Djamshid Tavangarian. Survey on Mobility and Multihoming in the Future Internet. *Wireless Personal Communication*, October 2012.

[49] Albert Greenberg, Gísli Hjálmtýsson, David A Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review*, 35(5):41–54, October 2005.

[50] Dongsu Han, Ashok Anand, Aditya Akella, and Srinivasan Seshan. Rpt: Re-architecting loss protection for content-aware networks. In *The 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, April 2012.

[51] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, Wenfei Wu, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: Efficient Support for Evolvable Internetworking. In *Proceedings of the 9th Usenix Symposium on Networked Systems Design and Implementation*, NSDI 2012, 2012.

[52] Dongsu Han, Robert Grandl, Aditya Akella, and Srinivasan Seshan. FCP: A Flexible Transport Framework for Accommodating Diversity. In *Proceedings of Sigcomm 2013*, New York, NY, USA, 2013. ACM.

[53] Sangjin Han et al. Packetshader: A GPU-accelerated software router. In *SIGCOMM*, 2010.

[54] Sangjin Han, Keon Jang, KyoungSoo Park, and Sue Moon. PacketShader: a GPU-accelerated software router. In *Proc. ACM SIGCOMM*, New Delhi, India, August 2010.

[55] T. Hardie. Rfc 3258 - distributing authoritative name servers via shared unicast addresses, Apr 2002.

[56] Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Soo Bum Lee, Xin Zhang, Sangjae Yoo, Virgil Gligor, and Adrian Perrig. STRIDE: Sanctuary Trail – Refuge from Internet DDoS Entrapment. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, May 2013.

[57] Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Adrian Perrig, Akira Yamada, Samuel C. Nelson, Marco Gruteser, and Wei Meng. LAP: Lightweight anonymity and privacy. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2012.

[58] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. SIGCOMM*, 2014.

[59] IETF HTTPbis Working Group. Http/2. `http://http2.github.io/`.

[60] V. Jacobson, D. K. Smetters, N. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. Braynard. VoCCN: voice over content-centric networks. In *Proceedings of the 2009 Workshop on Re-architecting the Internet (ReArch 2009)*, December 2009. Colocated with ACM CoNext 2009.

[61] M. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, and K. Park. Kargus: A highly scalable Software-based Intrustion Detection System. In *ACM CCS*, October 2012.

[62] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 97–108, New York, NY, USA, 2012. ACM.

[63] Anuj Kalia, Michael Kaminsky, and David G. Andersen. Using RDMA efficiently for key-value services. In *Proc. SIGCOMM*, 2014.

[64] Anuj Kalia, Dong Zhou, Michael Kaminsky, and David Andersen. Raising the bar for using gpus in software packet processing. In *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation*, Oakland, CA, May 2015.

[65] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02*, August 2002.

[66] Dina Katabi and John Wroclawski. A framework for scalable global ip-anycast (gia). In *ACM SIGCOMM*, pages 3–15, 2000.

[67] Frank Kelly. Charging and rate control for elastic traffic. In *European Transactions on Telecommunications*, 1997.

[68] Frank Kelly and Gaurav Raina. Explicit congestion control: charging, fairness and admission management. In *Cambridge University Press*, July 2010.

[69] Frank Kelly, Gaurav Raina, and Thomas Voice. Stability and fairness of explicit congestion control with small buffers. In *SIGCOMM Computer Communication Review*, 2008.

[70] S. Kent, C. Lynn, and K. Seo. Design and analysis of the secure border gateway protocol (s-bgp). In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings*, volume 1, pages 18–33 vol.1, 2000.

[71] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Ja, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM*, August 2014.

[72] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure. In *Proceedings of the International World Wide Web Conference (WWW)*, May 2013.

[73] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[74] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *ACM IMC*, 2012.

[75] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta algorithms for hierarchical web caches. In *Proc. ICPCC*, 2004.

[76] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.*, 2006.

[77] Ed A. Li. Rtp payload format for generic forward error correction. In *RFC 5109*, December 2007.

[78] Xiaozhou Li, David G. Andersen, Michael Kaminsky, and Michael J. Freedman. Algorithmic improvements for fast concurrent cuckoo hashing. In *Proceedings of EuroSys 2014*, April 2104.

[79] Hyeontaek Lim, David G. Andersen, and Michael Kaminsky. Practical batch-updatable external hashing with sorting. In *Proc. Meeting on Algorithm Engineering and Experiments (ALENEX)*, January 2013.

[80] Hyeontaek Lim, Bin Fan, David G. Andersen, and Michael Kaminsky. SILT: A memory-efficient, high-performance key-value store. In *Proc. 23rd ACM Symposium on Operating Systems Principles (SOSP)*, Cascais, Portugal, October 2011.

[81] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proc. SIGCOMM*, 2012.

[82] Michel Machado. *Linux XIA: An Interoperable Meta Network Architecture*. PhD thesis, Department of Computer Science, Boston University, May 2014.

[83] Michel Machado, Cody Doucette, and John W. Byers. Understanding internet video viewing behavior in the wild. In *11th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '15)*, May 2015.

[84] Ratul Mahajan, David Wetherall, and Thomas Anderson. Mutually controlled routing with independent ISPs. In *Proceedings of the 4th USENIX conference on Networked systems design & implementation*. USENIX Association, April 2007.

[85] John Markoff and Nicole Perlroth. Firm Is Accused of Sending Spam and Fight Jams Internet. `http://www.nytimes.com/2013/03/27/technology/internet/online-dispute-becomes-internet-snarling-attack.html`, 2013.

[86] Microsoft. Teredo overview, 2007.

[87] Zhongxing Ming, Mingwei Xu, and Dan Wang. Age-based cooperative caching in information-centric networks. In *Proc. INFOCOM WORKSHOPS*, 2012.

[88] Greg Minshall, Yasushi Saito, Jeffrey C. Mogul, and Ben Verghese. Application performance pitfalls and tcp's nagle algorithm. In *SIGMETRICS Performance Evaluation Review*, 2000.

[89] Matthew Mukerjee, Srini Seshan, and Peter Steenkiste. Understanding tradeoffs in incremental deployment of new network architectures. In *9th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2013)*. ACM, December 2013.

[90] David Naylor, Matthew K. Mukerjee, and Peter Steenkiste. Balancing accountability and privacy in the network. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 75–86, New York, NY, USA, 2014. ACM.

[91] Arbor Networks. Infrastructure security survey. `http://www.arbornetworks.com/sp_security_report.php`.

[92] Erik Nordstrom, David Shue, Prem Gopalan, Rob Kiefer, Matvey Arye, Steven Ko, Jennifer Rexford, and Michael J. Freedman. Serval: An end-host stack for service-centric networking. In *Proc. 9th Symposium on Networked Systems Design and Implementation*, NSDI '12, 2012.

[93] C. Partridge, T. Mendez, and W. Milliken. Rfc 1546 - host anycasting service, Nov 1993.

[94] Charles E. Perkins and David B. Johnson. Mobility Support in IPv6. In *ACM MobiCom*, November 1996.

[95] Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, and Arvind Krishnamurthy. One tunnel is (often) enough. In *Proceedings of the 2014 Conference on Applications, Technologies, Architectures, and Protocols for COmputer Communications*. ACM, August 2014.

[96] Lucian Popa, Ali Ghodsi, and Ion Stoica. HTTP as the Narrow Waist of the Future Internet. In *Proceedings of ACM HotNets 10*, Monterey, CA, October 2010. ACM.

[97] PPTV. `http://www.pptv.com/`.

[98] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proc. ICN*, 2012.

[99] Ioannis Psaras, Richard G. Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. Modelling and evaluation of ccn-caching trees. In *Proc. NETWORKING*, 2011.

[100] Barath Raghavan, Tadayoshi Kohno, Alex C. Snoeren, and David Wetherall. Enlisting ISPs to improve online privacy: IP address mixing by default. In *Proceedings of PETS '09*, pages 143–163, 2009.

[101] Y. Rekhter, T. Li, and S. Hares. Rfc 4271 - a border gateway protocol 4 (bgp-4), Jan 2006.

[102] G. Rossini and D. Rossi. A dive into the caching performance of content centric networking. In *Proc. CAMAD*, 2012.

[103] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[104] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984.

[105] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri. Mobilityfirst future internet architecture project. In *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, pages 1–3, New York, NY, USA, 2011. ACM.

[106] Ankit Singla, Balakrishnan Chandrasekaran, P. Brighten Godfrey, and Bruce Maggs. The Internet at the Speed of Light. In *Proc. HotNets*, 2014.

[107] Won So, Ashok Narayanan, David Oran, and Mark Stapp. Named Data Networking on a Router: Forwarding at 20Gbps and Beyond. In *Proceedings of ACM SIGCOMM*, SIGCOMM'13, 2013.

[108] Ahren Studer and Adrian Perrig. The coremelt attack. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, September 2009.

[109] Yi Sun, Seyed K. Fayaz, Yang Guo, Vyas Sekar, Yun Jin, Mohamed-Ali Kaafar, and Steve Uhlig. Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads . In *Proc. CoNext*, 2014.

[110] Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. PoliCert: Secure and Flexible TLS Certificate Management. In *Proceedings of th ACM Conference on Computer and Communications Security*, November 2014.

[111] The Chromium Projects. Spdy. `http://www.chromium.org/spdy`.

[112] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel. A trace-driven analysis of caching in content-centric networks. In *Proc. ICCCN*, 2012.

[113] Bruno Vavala, Nuno Neves, and Peter Steenkiste. Concatenating PALs Robustly for Verifiable Trusted Computations, 2013. Under submission.

[114] Arun Venkataramani, Ravi Kokku, , and Mike Dahlin. Tcp nice: A mechanism for background transfers. In *Proceedings of the 5th Symposium on Operating Systems Design (OSDI)*, December 2002.

[115] Chong Wang and John W. Byers. Incentivizing efficient content placement in a global content oriented network. In *CS Department, Boston University, Technical Report BUCS-TR-2012-012*, June 2012.

[116] Yi Wang, Ting Zhang, Kunyang Peng, Qunfeng Dong, bin Liue, Wei Meng, Huicheng Dai, Xin Tian, Hao Wu, and Di Yang. Wire speed name lookup: A gup-based appraoch. In *NSDI'13*. USENIX Association, 2013.

[117] Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie. Optimal cache allocation for content-centric networking. In *Proc. ICNP*, 2013.

[118] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *Proc. 8th USENIX NSDI*, Boston, MA, April 2011.

[119] XIA Security Architecture. http://www.cs.cmu.edu/afs/cs/project/xia/www/Documents/xia-security-arch.pdf.

[120] Wen Xu and Jennifer Rexford. Miro: Multi-path interdomain routing. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 171–182, New York, NY, USA, 2006. ACM.

[121] Hong Yan, David A. Maltz, T. S. Eugene Ng, Hemant Gogineni, Hui Zhang, and Zheng Cai. Tesseract: A 4D Network Control Plane. In *Proc. 4th USENIX NSDI*, Cambridge, MA, April 2007.

[122] Z. Zhu and R. Wakikawa and L. Zhang. A Survey of Mobility Support in the Internet, July 2011. IETF RFC 6301.

[123] Fuyuan Zhang, Limin JIa, Cristina Basescu, Tiffany Hyun-Jin Kim, Yih-Chun Hu, and Adrian Perrig. Mechanized Network Origin and Path Authenticity Proofs. In *Proccedings of the ACM Conference on Computer and Communications Security*, November 2014.

[124] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *IEEE Symposium on Security and Privacy (Oakland)*, Oakland, CA, May 2011.

[125] Dong Zhou, Bin Fan, Hyeontaek Lim, David G. Andersen, and Michael Kaminsky. Scalable, High Performance Ethernet Forwarding with CuckooSwitch. In *Proc. 9th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, December 2013.