MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture

Arun Venkataramani, James F. Kurose
U. Massachusetts Amherst

Dipankar Raychaudhuri, Kiran Nagaraja
Rutgers U.

Suman Banerjee
U. Wisconsin-Madison

Z. Morley Mao
U. Michigan

ABSTRACT

MobilityFirst is a future Internet architecture with mobility and trustworthiness as central design goals. Mobility means that all endpoints – devices, services, content, and networks – should be able to frequently change network attachment points in a seamless manner. Trustworthiness means that the network must be resilient to the presence of a small number of malicious endpoints or network routers. MobilityFirst enhances mobility by cleanly separating names or identifiers from addresses or network locations, and enhances security by representing both in an intrinsically verifiable manner, relying upon a massively scalable, distributed, global name service to bind names and addresses, and to facilitate services including device-to-service, multicast, anycast, and context-aware communication, content retrieval, and more. A key insight emerging from our experience is that a logically centralized global name service can significantly enhance mobility and security and transform network-layer functionality. Recognizing and validating this insight is the key contribution of the MobilityFirst architectural effort.

Categories and Subject Descriptors

C.4.3 [**Computer Systems Organization**]: COMPUTER-COMM. NETWORKS—*Network Architecture and Design*

Keywords

Mobility; security; global name service; network architecture

1. INTRODUCTION

The enormous success of the current Internet as well as our deepened understanding of its strengths and weaknesses have fueled recent research interest in clean-slate designs for a next-generation Internet, informed by our collective experience but unencumbered by concerns of backwards compatibility. Driven by this goal, and by the technology and usage trends shaping today's Internet, we have been working on MobilityFirst, a future Internet architecture with mobility and trustworthiness as central design goals.

To appreciate the need for mobility as a crucial top-level design goal, one need only consider that smartphones alone today far outnumber tethered Internet hosts. Although the Internet has flexed remarkably to accommodate this growth, its core architecture and protocols designed originally for tethered hosts have hardly changed – a state of affairs that is problematic for end users, operators, and application developers and threatens to stymie long-term growth and innovation. From a user's perspective, the Internet fails to satisfy seemingly straightforward expectations: a download does not resume gracefully when moving from home to work; a smartphone VoIP call does not seamlessly switch from WiFi at home to LTE on the road; ad hoc mobile-to-mobile communication can not be easily used to exchange information when the infrastructure network is congested or unavailable. From a performance perspective, the Internet's TCP/IP protocol stack has been widely recognized as being fragile in mobile and wireless network scenarios [8, 19]. From an application developer's perspective, the lack of architectural support for seamless mobility necessitates additional cloud-based infrastructure and redundant application-specific workarounds. Mobile IP[21], a plausible mobility approach, is based on a legacy cellular worldview that users have a single network "home", can be reached only via a contemporaneous IP path, are connected to a single network at any time, require only host-to-host (rather than host-to-content) communication, and move infrequently across networks.

Our position is that the Internet's naming and addressing architecture is in good part responsible for these problems and has serious implications for its trustworthiness as well. The Internet has been widely criticized for conflating identity and location by overloading an IP address to represent both, which complicates mobility (single identity, changing locations) and more so with multihoming (single identity, multiple locations). Conflating identity and location also makes it difficult to verify that an endpoint is indeed at the claimed location making it easy to hijack or spoof addresses today – a problem that will only be exacerbated with frequent mobility of billions of mobile devices across many network addresses per day. More broadly, the Internet's core protocols, designed originally with benign users in mind, are vulnerable to abuse. For example, a single misconfigured router can render large portions of the Internet unreachable. The lack of verifiable endpoint and network identifiers makes it harder to account for adversarial behavior in protocol design and is at the root of a number of security, privacy, and DDoS vulnerabilities.

An important design decision in MobilityFirst that helps achieve the synergistic goals of mobility and security is a clean separation of names and network addresses and the use of a massively scalable global name service (GNS) to dynamically bind names and addresses. In addition to enabling seamless mobility, this separation also enhances security, as it allows names and addresses to be defined as globally unique identifiers (GUIDs) that are verifiably derived from public keys and are robust to hijacking or spoofing. This GUID-based communication assisted by the GNS forms MobilityFirst's "narrow waist" and is sufficiently flex-

ible to accommodate a variety of endpoint principals including interfaces, devices, users, services, content, context-aware descriptors (e.g., "pedestrians on the UMass campus") and location-independent communication primitives such as device-to-device, device-to-service, content retrieval, context-aware delivery, multicast, anycast, and more.

A key architectural insight that has emerged through our design and implementation effort is that *a logically centralized global name service can significantly enhance mobility, security, and rich network-layer functionality.* We had originally viewed the GNS as primarily a distributed resolution infrastructure (similar in spirit to a next-generation DNS) to enhance mobility. However, we have since come to realize that, architecturally, a fast, secure, logically centralized GNS can transform how network-layer functionality is implemented (as detailed in §2). We have prototyped and evaluated key MobilityFirst elements in a combination of wide-area testbeds, and have also conducted evaluations driven by simulations, measurements, and theoretical analyses; these and more realistic field trial plans are outlined in §3.

2. MOBILITYFIRST ARCHITECTURE

In this section, we first describe the key elements of MobilityFirst that show how the GNS is critical to enhancing mobility, security, and several other network-assisted functions, including routing, context-aware group communication, and content retrieval. This functional overview sets the stage for the requirements and the design of the GNS itself (§2.3).

2.1 Basic communication abstraction

The core primitive enabled by MobilityFirst is communication with *location-independent* and *verifiable* names. For example, an application can invoke `connect(service_name)` or `get(content_name)` or `message(group_name)` to communicate with the named arguments without worrying about their (changing) location(s) or impersonation by malicious entities. We begin with an overview of naming and addressing to help appreciate this communication abstraction.

2.1.1 Naming and addressing

A *name* in MobilityFirst is a *globally unique identifier* (GUID) that can be used to identify a variety of *principals* such as an interface, a device, a service, a human end-user, content, or (recursively) a collection of GUIDs.

Self-certifying identifiers. A GUID is self-certifying, i.e., any principal can authenticate another principal claiming a GUID without the need for third-party certification. A self-certifying GUID is derived simply as a one-way hash of a public key, so a GUID can be authenticated using a simple, bilateral challenge-response procedure that does not require an external certification authority.

The bilateral challenge-response works as follows. Suppose a principal X (say, a router) wants to authenticate another principal Y (say, a destination), i.e., X wants to verify that Y is indeed the rightful owner of the GUID Y. Then, X issues a challenge by sending a random nonce n to Y. Y is expected to respond with $[K^+, K^-(n)]$, where K^+ is a public key and $K^-(n)$ is the nonce encrypted using the corresponding private key. Upon receiving the response, X first checks that $H(K^+) = Y$, where $H(.)$ is a well-known one-way hash function, and then checks that $K^+(K^-(n)) = n$. If both checks pass, then X has authenticated Y.

In addition to a GUID, it is convenient to assign a principal an optional human-readable name (e.g., "John Smith's cell phone") or an inexact intent (e.g., a set of search keywords or other abstract descriptions). To this end, a name certificate binds the human-readable name or intent to a public key. Endpoints wanting to securely communicate using human-readable names must thus first obtain a certificate from a trusted a certification authority.

Network addresses. A *network address* (NA) is a self-certifying identifier for a *network*, i.e., an autonomous collection of interconnected devices that act as intermediate forwarders of traffic sourced by or destined to GUIDs attached to any device in the collection. An NA most naturally corresponds to an autonomous system in today's parlance, but could also be used to identify finer-grained collections such as a subnet or one or more base stations or coarser-grained collections such as an Internet Service Provider. A GUID is said to be attached to an NA if it is directly connected to one or more forwarding devices in the NA.

2.1.2 End-to-end communication

The GNS enables end-to-end communication by mapping endpoint identifiers (GUIDs or human-readable names) to a flexible set of attributes including but not limited to their network addresses. Thus, the GNS subsumes a *name certification service* that resolves a human-readable name to a GUID and a *name resolution service* that resolves a GUID to its attributes (with more details deferred to §2.3).

To contact a GUID, a sending endpoint queries the GNS to obtain an NA corresponding to a GUID (much like it queries DNS to obtain an IP address for a domain name) before sending the first packet to the destination. The tuple [GUID, NA] is a *routable* destination identifier carried in packet headers. Senders can also send a packet addressed just to a GUID, thereby implicitly delegating to the first-hop router the task of querying the name service for an NA.

End-to-end packet forwarding is accomplished in two steps, first by an internetwork and then by an intranetwork routing protocol. The internetwork routing protocol is responsible for delivering packets to the destination NA in the packet header (oblivious of the destination GUID). Once the packet reaches the destination NA, an intranetwork routing protocol involving routers in NA is responsible for delivering the packet to the GUID. As in today's interdomain Internet, each NA can independently choose its intranetwork routing protocol. As GUIDs can not encode any information about network location, the intranetwork routing protocol must be capable of routing on flat identifiers.

2.2 Enhanced network functions

Next, we describe several important functional components of MobilityFirst–(1) endpoint mobility, (2) scalable routing, (3) context-awareness, (4) content retrieval, and (5) evolvability–that are enabled or assisted by the GNS.

2.2.1 Handling endpoint mobility

The GNS-driven end-to-end communication as above implicitly enables only *pre-lookup mobility*, which suffices if endpoints rarely change network addresses. However, when mobility is the norm, three other types of mobility, as shown in Figure 1, must be handled. *Connect-time mobility* is when a destination B moves after the initiator A's query but before before a connection has been mutually established through

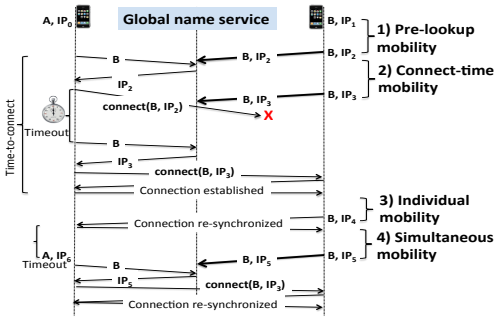

Figure 1: Four kinds of mobility (1) Pre-lookup, (2) Connect-time, (3) Individual, (4) Simultaneous, three of which require a global name service.

a three-way handshake. *Individual mobility* refers to either (but not both) of the endpoints moving after a connection has been established. *Simultaneous mobility* is when an endpoint moves mid-connection after the other endpoint has moved but before it could re-synchronize the connection.

Three of the four types of mobility have to rely on the GNS; only individual mobility can be handled in a purely bilateral manner. We also note that although prior work on connection migration [25, 6] often considers simultaneous mobility to be a rare case, it can be pretty common in disruption-tolerant mobile app scenarios and does not require both endpoints to change their addresses "simultaneously", e.g., when a mobile user stops watching a video on her phone using the cellular network and then resumes watching it after a few hours using a WiFi network, by which time the virtual machines hosting the video server in the cloud has been migrated for load balancing.

2.2.2 Scalable routing and network mobility

The internetwork routing protocol enables reachability to NAs much like the current Internet enables reachability to IP prefixes. Thus, the number of forwarding table entries in a core router is commensurate to the total number of NAs. As the number of NAs may grow significantly over time (e.g., home networks, vehicular networks, body area networks, etc.), the internetwork routing protocol is designed to support a small number of levels of hierarchy so as to trade off packet header space against forwarding table size. Our candidate internetwork routing design, *core-edge* routing supports a two-level hierarchy with networks explicitly designated as *core* or *edge* networks.

A core network router only maintains forwarding entries for other core networks and a small number of their "customer" edge networks. An edge network router maintain forwarding entries only for a small number of their "provider" core networks and edge networks in their vicinity. The name service enables the two-level interdomain routing protocol by resolving a GUID to a two-tuple $[X, T]$ (instead of a single NA), where X is the most downstream core network enroute to GUID and T is the terminal network to which the GUID is attached. An edge network need not be directly connected to a core network, however, it must ensure that at least one core network agrees to maintain forwarding state for it.

Like endpoint mobility, the GNS also facilitates *network mobility* wherein a network as a whole moves across locations, e.g., when vehicular or body-area networks physically move and connect to different access networks. The details of network mobility as well as other candidate internetwork routing protocols (e.g., edge-aware interdomain routing or EIR) are described in longer papers [28, 29].

2.2.3 Context-aware communication

A key primitive enabled by the GNS in MobilityFirst is context-aware communication, or the ability to communicate with endpoint principles identified by sophisticated, attribute-based descriptions. Unlike the current Internet that mainly provides a primitive to send data to an IP address, context-aware communication allows an endpoint to send (receive) data to (from) a dynamic set of endpoints without explicitly managing or even knowing the membership of the set. For example, in MobilityFirst, an application can invoke `send(message, "Taxis near TimesSquare")` or even bind a socket to such attribute-based descriptors.

Multicast is a simple instance of context-aware delivery. A multicast GUID (MID) has the same format as a regular GUID and the resolved output of the name service has the same format as multi-homed network address. However, the name resolution and routinf differ as follows. The name service maintains a membership set for each MID that consists of all GUIDs subscribed to the multicast group. Each member GUID i in MID subscribes to the group via a single home, NA_i. The name service resolves a MID by returning the union of all NA_i's having at least one GUID subscribed to the MID. By default, the sender is responsible for sending data addressed to $[MID, NA_i]$ for each of the returned NA_i's. When packets arrive at a destination NA_i, the NA_i is responsible for resolving the MID to the subset of member GUIDs attached to its network and forwarding a copy to each member relying on the intranetwork routing protocol.

Context-aware delivery generalizes multicast to groups based on attribute-based descriptors. The GNS's support for maintaining flexible attributes, not just network addresses, using an extensible key-value interface is key to enabling context-aware delivery. To enable geo-casting, for example, each potential member must maintain its geolocation ([lat,long]) attribute and enable read privileges for potential senders. In order to send a geocast message, the GNS creates on-demand a context-aware MID whose members are all GUIDs matching the specified geolocation. The subsequent message delivery is identical to multicast as described above. Context-aware MIDs can also be created using more sophisticated comparison and logical operators, e.g., `"type='taxi' and [lat,long] within [target_lat, target_long, radius]"` to send a geofenced message as in the example above (with more details in §2.3 and §3).

2.2.4 Content retrieval and storage-aware routing

Content in MobilityFirst is a first-class endpoint principal and is named using self-certifying GUIDs like other endpoints but with some differences. First, unlike interface or device GUIDs, a self-certifying content GUID is computed as a one-way hash of the content itself, allowing any entity to verify its integrity. Second, the GNS does not store state for all content GUIDs as that would impose a prohibitively high overhead on any GNS provider; instead, a routable content address is encoded as a two-tupe [PID, CID], wherein CID is the content GUID and PID is the publisher's GUID. The PID could either be a core or a terminal network itself or be attached to one and the GNS and/or the interdomain routing protocol help route to the PID that should then serve the

CID. Endpoints can know of the [PID, CID] tuple to request through several different means including hyperlinks in web pages, or by connecting to a publisher's web service by resolving a human-readable name like "www.nytimes.com". The GNS's support for indirection (§2.3) is convenient to translate all requests to PID_1 to PID_2 instead (similar in spirit to CNAME aliasing today) in order for a content producer PID_1 to delegate content distribution to a CDN PID_2.

Storage-aware routing. Storage-aware routers support opportunistic caching and retrieval of content that helps contain demand for popular content close to the edge. A storage-aware router can intercept a request for a CID if it has a copy of that content stored locally and serve the content to the requesting endpoint. Note that a storage-aware router or an edge network can unilaterally enable support for opportunistic content caching and retrieval without coordinating with the publisher or other routers or networks, as the requesting endpoint can verify the content based on the CID alone. We also emphasize that not all routers need be storage-aware; indeed recent studies have shown that opportunistic edge caching alone suffices [24, 13] to achieve much of the benefit of path-caching advocated by more content-centric network architectures [2].

Block transport. Storage-aware routing combines well with *block transport*, MobilityFirst's network-assisted transport alternative to today's end-to-end TCP. Block transport transfers blocks, or large chunks of contiguous data (possibly entire files), as opposed to small packets in a per-hop-reliable manner, which significantly enhances performance, fairness, and disruption-tolerance in mobile settings. A detailed design and implementation of a block transport protocol, Hop, is described here [19]. MobilityFirst generalizes Hop to work with *segments* (i.e., a continuous set of links with storage-aware routers at each end) instead of single-link hops.

2.2.5 Evolvable compute layer

MobilityFirst supports a computing layer for evolvability, wherein network service providers either themselves provide value-added services to their end-users, or enable an open platform where third-party services (e.g., transcoding, onion routing, etc.) validated by the network provider can register and be co-deployed at designated provider PoPs. Compute plane services so deployed can either be located explicitly by endpoints or network intermediaries through their GUID registered in the GNS or opportunistically when an on-path router supports the requested service. By enabling deployment of in-network compute services in a virtualized environment, MobilityFirst allows for new network services to be quickly rolled out without impacting the fast forwarding path or legacy traffic that does not need the new services[11].

2.3 GNS requirements and design overview

A natural question raised by the central role of the GNS in MobilityFirst is: can we build a practical GNS that can deliver on the expectations? Much of the expected functionality boils down to one key distributed systems challenge: *ensuring that any endpoint or router gets the look and feel of a high-availability name service that is nearby (≈ few milliseconds) and rapidly returns up-to-date responses.*

A more complete set of design goals is as follows.

(1) Time-to-connect: The design must ensure low latencies for name lookups to return up-to-date values, which determines the *time to connect* to a destination when the value being queried for is a network address.

(2) Resource cost: The design must ensure low replication cost. A naive way to minimize lookup latencies is to replicate every name record at every available location, however high mobility implies high update rates, so the cost of pushing each update to every replica would be prohibitive.

(3) High availability: The design must be resilient to failures including disasters impacting an entire datacenter, and (by consequence) also prevent crippling load hotspots.

(4) Security: The design must be robust to malicious user behavior such as hijacking or corrupting name records. The design must support flexible access control policies to ensure the desired privacy of name records.

(5) Federation: The design must allow different name service providers to co-exist and users to freely choose providers.

(6) Extensibility: The design must be extensible to a rich set of attributes associated with a name and resolution policies to enable new group communication primitives.

2.3.1 Auspice GNS design overview

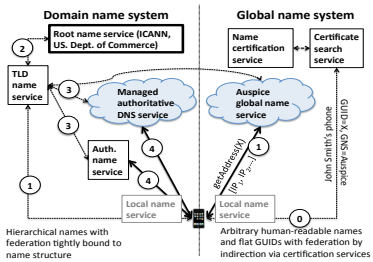

Figure 2: DNS vs. GNS: Auspice can be deployed similar to a managed DNS provider today (left) so as to provide name resolution service for its customer GUIDs (right). Name certification services (say Verisign) bind a human-readable name to a GUID and its GNS provider, and certificate search services (say Google) can help index and distribute certificates from all certification services. Solid (dotted) lines represent frequent (infrequent) query paths for a given mobile destination. Except for the tightly controlled DNS root service, all services above are designed to be purveyed competitively.

To address the above goals, we have developed Auspice, a GNS decribed below that is designed as a massively geo-distributed key-value store for name resolution. We have also designed and implemented an alternative in-network GNS, DMap [30] (along with the associated service API [9]), that provides an in-network DHT scheme to map a self-certifying identifier to a fixed number of resolver locations, with clients choosing the closest mapped resolver. Below, we explain the Auspice design in detail.

Geo-distribution is essential to the latency and availability goals while the key-value design enables extensibility in Auspice. Each *name record* in Auspice is associated with a globally unique identifier (GUID) that is also the record's primary key. A name record contains an associative array of key-value pairs as shown, wherein each key K_i is a string and the value V_i may be a string, a primitive type, or recursively a key-value pair.

GUID $\mid K_1, V_1 \mid K_2, V_2 \mid \cdots$

Loosely speaking, the human-readable alias is analogous to a DNS domain name and a name record to a zone file, but with the following important differences from DNS.

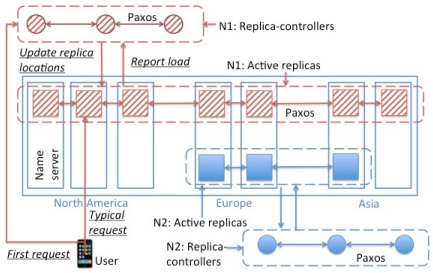

Figure 3: Geo-distributed name servers in Auspice. Replica-controllers decide placement of active replicas and active replicas handle requests from end-users. N1 is a globally popular name and is replicated globally; name N2 is popular in select regions and is replicated in those regions.

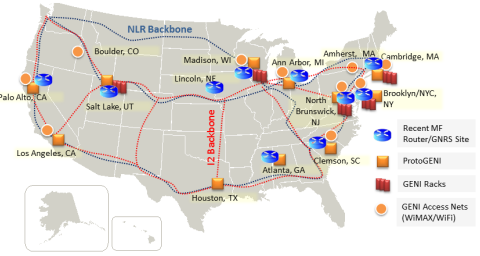

Figure 4: GENI testbed, showing deployment of MobilityFirst naming and routing prototypes.

Certification. As shown in Fig. 2, to initiate communication with a destination Y, an endpoint X must first obtain a certificate of the form $[JohnSmith1956{:}Phone1, Y, P]_{K^-}$ that binds the human-readable alias to the GUID Y and its GNS provider P, and is signed by the private key K^- of an NCS that X trusts. A certificate search service can help index certificates from different NCSes, and help X find a certificate from a trusted NCS as well as find the human-readable alias based on keyword searches.

Federated trust. Unlike ICANN and root DNS servers that respectively act as a single name adjudication authority and root of trust, the design above decentralizes trust across different NCS providers and potentially allows for endpoints to use quorum-based approaches to resolve name conflicts (or conflicting certificates). More importantly, the federated design above allows for endpoints to select arbitrary human-readable names and NCS providers unlike DNS that restricts domain names to be hierarchical and federation and the DNSSEC keychain to strictly follow the name structure. An inevitable restriction of the above design is that two endpoints can only communicate securely if they share a trusted NCS provider, a design we argue is preferable to and a generalization of the single root of trust model.

Extensibility. The design above cleanly separates the GNS provider's resource-intensive responsibility of name resolution under high mobility from the slow-changing certification process. The key-value API enables an extensible name record representation to enhance functionality while ensuring security and privacy. By default, each top-level key has associated read and write ACLs that could either be a blacklist or whitelist of GUIDs that respectively have read or write access. Thus, users can choose to enable geocast capability (as in §2.2.3) only to emergency personnel or potential customers using such ACLs.

Automated geo-replication. The key to achieving the first three goals in Auspice is a *demand-aware replication* engine that automatically decides for each GUID the number and locations of replicas of its name record based on its lookup rate, update rate (mobility or attribute changes), geo-distributed pattern of lookup demand, and global system capacity. The placement engine consists of replica controllers (themselves replicated using Paxos) that proactively create replicas (not passive cached copies) of name records close to their pockets of demand so as to optimize time-to-connect latency and resource cost. Figure 3 illustrates its design with more details deferred to the Auspice paper [23].

2.4 Trustworthiness discussion

Our position is that easily verifiable identifiers go a long way towards addressing the Internet's security vulnerabilities. The GUID-based communication abstraction is MobilityFirst's thin waist. Unlike IP addresses, GUIDs and NAs can not be hijacked or spoofed and can be verified in a purely bilateral manner. The GNS itself further fortifies security and privacy through the use of flexible ACLs and support for pseudonyms for anonymous online transactions. The massive replication and federation of name certification as well as name resolution in the GNS implies a very high resource cost for an adversary to exhaust GNS resources through DDoS attacks.

The support for in-network content storage and retrieval means that static (named) content is resistant to flash floods. Intrinsic support for disruption-tolerance through block transport means that end-to-end communication will continue despite the absence of a contemporaneously connected path. At the interdomain level, self-certifying NAs achieve similar security guarantees as S-BGP but without relying on a PKI in the common case [5]. Finally, verifiable identifiers simplify the deployment of many proposals based on filtering and/or capabilities to alleviate network-layer DDoS attacks in MobilityFirst compared to the current Internet.

3. PROTOTYPING AND EVALUATION

To evaluate the MobilityFirst architecture, we have built multiple prototypes of its key components including the global name service, routing and forwarding engine, a virtualized compute layer, and an endhost socket library. Although our effort has been driven by a clean-slate design process, we believe it is important to identify a realistic deployment path for the architecture to be successfully adopted, so we have emphasized incremental deployability of key components in our prototyping and field trial efforts.

Auspice and msocket. We have implemented a full-featured prototype of Auspice as described above and have been internally using an Amazon EC2 deployment for research purposes for many months now [23]. We have also developed msocket, an endpoint socket library that interoperates with Auspice and is similar to the BSD API but with (1) intrinsic support for mobility and multihoming (e.g., maintaining a persistent connection despite moving across different vertical networks), (2) multipath transport, (3) mobile-to-mobile communication despite the presence of address-translating or (unidirectional) firewalling middleboxes, and (4) context-aware application development, e.g., an API that allows an app to invoke `msocket.bind(lat, long, radius)` for geofenced messaging.

Auspice and msocket can also be used on top of today's

Internet (with IP addresses instead of self-certifying NAs) and offer the first scalable solution for seamlessly handling all four types of mobility (§2.2.1). Some evaluation highlights (refer [23, 31] for details) are as follows:
(1) Auspice and msocket enable recovery from both individual and simultaneous mobility in ≈2 RTTs after both endpoints are back online.
(2) Compared to best-of-breed, commercial DNS providers today, Auspice yields significant cost and/or performance gains even for today's read-dominated (hardly mobile) names.
(3) Under high mobility, with a geo-distributed deployment of just hundred nodes, Auspice can resolve queries within median and 95th percentile latencies of 20ms and 80ms resp., outperforming state-of-the-art alternatives based on DHT replication by up to an order of magnitude. For a broader evaluation of endpoint and content mobility approaches in MobilityFirst as well as alternate architectures, see [14]

Forwarding and routing. We have implemented two prototypes of the MobilityFirst forwarding plane, the first based on a more traditional distributed control plane using Click, and a second based on a logically centralized control plane using an SDN-capable router using OpenFlow and OpenDayLight. Both prototypes also implement hop-by-hop segmented transport [19] and storage-aware routing protocols [20, 27] that are robust to changes in network conditions. On a commodity AMD quad-core 2.3GHz machine with 4GB RAM, we have been able to achieve line rates for GUID,NA forwarding of 700Mbps and close to 1Gbps respectively with the Click and SDN based routers. As expected, the performance is much slower with the SDN-based router when late binding via a GNS lookup is involved, and techniques to optimize the controller switch interface to better support late binding are part of ongoing work. Additionally, we have implemented the core-edge interdomain routing protocol integrated with a scalable, intradomain GUID-routing protocol based on Seattle [17], and the edge-aware interdomain routing protocol mentioned in §2.2.2.

Field trials. For a holistic validation of the MobilityFirst architecture, we also have three distinct real-world trials planned for the next phase of the MobilityFirst project –a mobile data services trial with a wireless ISP (5Nines) in Madison, WI; a content production and delivery network trial involving several public broadcasting stations in PA connected by a green-fields optical network called PennREN; and a public service weather emergency notification system (CASA) with end-users in the Dallas/Fort Worth area. These field trials are expected to provide further validation of the MobilityFirst protocol stack and and real-world insights into its usability for developing advanced mobile, content, context and cloud applications.

4. RELATED WORK

MobilityFirst synthesizes ideas from a large body of prior work so as to realize a holistic Internetwork architecture with mobility and security as central design goals. Our novel, high-level contribution is to recognize and validate the insight that a logically centralized global name service can significantly enhance mobility, security, and a number of rich network-layer functions, which to our knowledge has not been done before. Below, we compare MobilityFirst to a few other representative network architectures.

Handling mobility. Existing approaches to handle mobility can be broadly classified in three categories: (1) indirection, (2) global name resolution, (3) name-based routing. Indirection approaches, e.g., MobileIP [21], LISP[4], i3[26], GSM, route to a fixed network address, the home address, and a home agent router tunnels all data packets to the mobile's current location. Indirection schemes enable seamless mobility of one or both endpoints at any time and are oblivious to non-mobile endpoints, but as a consequence have to route all data through the home agent exacerbating path inflation. Global-name-resolution-based approaches, e.g., HIP[16], LNA[7], XIA[15] rely on a logically centralized service such as DNS or Auspice[23] that resolves an endpoint identifier to its network location(s). This approach requires a lookup to the name service at connection initiation time and in order to handle simultaneous (but not individual) mid-session mobility, but does not suffer from data path inflation. Pure name-based routing approaches, e.g., ROFL[10], TRIAD[12], NDN[2], eschew network locators and route directly on flat or structured names. This approach in theory allows any mobility to be completely seamless to endpoints, but in practice can induce outage times commensurate to convergence delays for routing unless they additionally rely on indirection or global name resolution.

Compared to HIP, an *endpoint stack* that pioneered self-certifying host identifiers, MobilityFirst is a *network architecture* with a number of additional conceptual and functional elements. MobilityFirst uses self-certifying identifiers not only for endpoints but also for network addresses and content (similar in spirit to AIP[5] and XIA). MobilityFirst features such as network mobility, disruption-tolerant transport, in-network content storage and retrieval, context-aware communication, and compute layer all require network-layer support that is absent in an endpoint architecture. A narrow view might compare HIP to a stripped-down version of MobilityFirst interoperable with today's routers,. But that view would miss the architectural centrality of MobilityFirst's GNS, which is philosophically distinct from HIP that relies on the DNS, a three-way UPDATE signaling mechanism to handle mobility of endpoints, and "rendezvous points" to handle the rare case of simultaneous mobility.

Like MobilityFirst, the XIA future Internet architecture [15] also advocates self-certifying identifiers and network locations and both architectures use them to represent a variety of principals. For evolvability, XIA represents addresses as a directed acyclic graph of self-certifying identifiers, wherein DAG paths correspond to possible "source routes" to reach the destination. In comparison, MobilityFirst is designed with more explicit support for mobility-centric services while keeping addresses and packet headers simple. Extensible functionality comes from the support for indirection, grouping, attribute-based lookups ivia the GNS and late binding at routers.

Both MobilityFirst and NDN [2] leverage opportunistic caching and retrieval of content via storage-aware routers. Neither requires every router to be storage-capable, which aligns well with recent studies [13, 24] showing that much of the benefit of on-path caching can be obtained by just caching close to the edge. However, an important difference is that, unlike MobilityFirst, NDN begins by eschewing network locators (like IP addresses or NAs) altogether opting instead to route directly over a hierarchically organized content name space. Our position is that while the approach is well suited to consumption of content that is highly aggregatable and moves infrequently, it faces scalability challenges

under high mobility. Enabling device-to-device or service-to-device communication with billions of devices moving across tens of network locations a day necessitates augmenting a pure name-based approach with indirection or a global name service, as also recognized by the NDN effort [3].

DNS vs. GNS. Until the early 80s, the Internet relied on a centrally maintained `HOSTS.TXT` text file for name resolution. The DNS arose in response to the rapidly increasing size of the file and the cost of distributing it. Mockapetris and Dunlap point to TTL-based caching to reduce load and response times as a key strength, noting that "the XEROX system [Grapevine [22]] was then ... the most sophisticated name service in existence, but it was not clear that its heavy use of replication, light use of caching ... were appropriate". We have since come a full circle, turning to active replication in Auspice in order to address the challenges of mobility, a concern that wasn't particularly pressing in the 80s. Compared to classical systems like Grapevine or ClearingHouse, Auspice supports automated, demand-aware placement of replicas and, through its support for context-aware delivery, is a step towards addressing some of the challenges to which Lampson alludes on representing "descriptive names" [18].

5. CONCLUSIONS

We overviewed MobilityFirst, a future Internet architecture with mobility and security as central design goals. MobilityFirst enables seamless mobility through a clean separation of names and addresses and enhances security by representing them using intrinsically verifiable identifiers. A key component of MobilityFirst that makes this approach practical is a geo-distributed, massively scalable global name service that rapidly resolves names to flexible attributes. More importantly, the deeper insight that has emerged from this effort is that a logically centralized global name service can significantly enhance not only mobility and security but a number of other network-layer functions. Our main contribution is to recognize and validate this insight, an endeavor that is made possible because of our improved understanding of fielding massive-scale distributed systems and necessitated by the explosive growth of mobile applications. The MobilityFirst effort is still ongoing and we welcome feedback from the community. More details are available at [1], including an app developer portal at `http://gns.name`.

Acknowledgement. This research was supported in part by the US National Science Foundation awards CNS-1040781, CNS-1040735, CNS-1039657, and CNS-1040648. We thank the rest of the MobilityFirst team and participants at NSF-FIA meetings for shaping the ideas in this paper.

6. REFERENCES

[1] MobilityFirst:. `http://mobilityfirst.cs.umass.edu/`.
[2] Named data networking. `http://www.named-data.net/`.
[3] Personal communication at FIA meetings. `http://www.nets-fia.net/Meetings/Fall13/Agenda-fall-13.html`.
[4] The Locator/ID Separation Protocol (LISP). RFC 6830.
[5] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). ACM SIGCOMM, 2008.
[6] M. Arye, E. Nordstrom, R. Kiefer, J. Rexford, and M. J. Freedman. A Formally-Verified Migration Protocol For Mobile, Multi-Homed Hosts. In *ICNP*, 2012.
[7] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *ACM SIGCOMM*, 2004.
[8] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN Routing As a Resource Allocation Problem. In *ACM SIGCOMM*, 2007.
[9] F. Bronzino, K. Nagaraja, I. Seskar, and D. Raychaudhuri. Network service abstractions for a mobility-centric future internet architecture. In *ACM MobiArch Workshop*, 2013.
[10] M. Caesar and T. Condie et al. ROFL: Routing on Flat Labels. In *ACM SIGCOMM.*, 2006.
[11] Y. Chen, B. Liu, Y. Chen, A. Li, X. Yang, and J. Bi. PacketCloud: an Open Platform for Elastic In-network Services. In *ACM MobiArch Workshop*, 2013.
[12] D. R. Cheriton and M. Gritter. An architecture for content routing support in the internet. In *USENIX USITS*, 2001.
[13] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, and A. Ghodsi et al. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *ACM SIGCOMM*, 2013.
[14] Z. Gao, A. Venkataramani, and J. Kurose. Towards a Quantitative Comparison of Location-Independent Network Architectures. In *ACM SIGCOMM*, 2014.
[15] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient Support for Evolvable Internetworking. In *USENIX NSDI*, 2012.
[16] P. Jokela, P. Nikander, J. Melen, J. Ylitalo, and J. Wall. Host Identity Protocol. In *Wireless World Research*, 2004.
[17] C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. In *ACM SIGCOMM*, 2008.
[18] B. W. Lampson. Designing a global name service. In *ACM PODC*, 1986.
[19] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched Networks: A New Paradigm for Wireless Transport. In *USENIX NSDI*, 2009.
[20] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. GSTAR: Generalized Storage-aware Routing for Mobilityfirst in the Future Mobile Internet. In *MobiArch Workshop*, 2011.
[21] C. E. Perkins. Mobile IP. *IEEE Comm.*, May 1997.
[22] M. D. Schroeder, A. D. Birrell, and R. M. Needham. Experience with Grapevine: the growth of a distributed system. *ACM Trans. Comput. Syst.*, February 1984.
[23] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav. A Global Name Service for a Highly Mobile Internetwork. In *ACM SIGCOMM*, 2014.
[24] A. Sharma, A. Venkataramani, and R. Sitaraman. Distributing Content Simplifies ISP Traffic Engineering. In *ACM SIGMETRICS*, 2013.
[25] A. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *ACM MOBICOM*, August 2000.
[26] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *ACM SIGCOMM'02*.
[27] X. Tie, A. Venkataramani, and A. Balasubramanian. R3: Robust Replication Routing in Wireless Networks with Diverse Connectivity Characteristics. In *MOBICOM*, 2011.
[28] A. Venkataramani, X. Tie, A. Sharma, D. Westbrook, H. Uppal, J. Kurose, and D. Raychaudhuri. Design Guidelines for a Global Name Service for a Mobility-Centric, Trustworthy Internetwork. *COMSNETS*, 2013.
[29] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri. EIR: Edge-aware Interdomain Routing Protocol for the Future Mobile Internet. Technical report, WINLAB, 2013.
[30] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri. DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet. In *IEEE ICDCS*, 2012.
[31] A. Yadav, A. Venkataramani, A. Sharma, and E. Cecchet. msocket: System Support for Developing Seamlessly Mobile,Multipath, and Middlebox-Agnostic Applications. Technical report, UMass SCS, 2013. `http://web.cs.umass.edu/publication/`.