Network Working Group
Internet Draft
Intended status: Informational

<Initial. Lastname>
<Affiliation>
July 17, 2012

MobilityFirst Future Internet Architecture: Specification of Architecture and Protocol

INFO (REMOVE): Lines prefixed with "INFO (REMOVE):", such as this (also highlighted in RED) are instructions for the author and *MUST* be removed completely prior to publication.

INFO (REMOVE): Use ONLY the paragraph styles provided in this sample template, in particular Heading 1 through Heading 9, Normal, Figure, and styles prefixed with "RFC". Use of other styles or modifying these allowed styles in any way (bold, italics, change font, spacing, numbering, bullets, etc.) will render the output incompatible.

INFO (REMOVE): Lines surrounded by "<>" (also highlighted in YELLOW) are placeholders and *MUST* be replaced by the author. Required boilerplate or sections are shown in BLUE and should not be omitted. Boilerplate shown in GREEN is included by choice.

INFO (REMOVE): Update the Intended status in the header as needed.

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The MobilityFirst project is founded on the premise that the Internet is approaching an historic inflection point, with mobile platforms and applications poised to replace the fixed-host/server model that has dominated the Internet since its inception. This predictable, yet fundamental, shift presents a unique opportunity to design a next generation Internet in which mobile devices, and

applications, and the consequent changes in service, trustworthiness, and management are primary drivers of a new architecture. The major design goals of our proposed architecture are: mobility as the norm with dynamic host and network mobility at scale; robustness with respect to intrinsic properties of wireless medium; trustworthiness in the form of enhanced security and privacy for both mobile networks and wired infrastructure; usability features such as support for context-aware pervasive mobile services, evolvable network services, manageability and economic viability. The design is also informed by technology factors such as radio spectrum scarcity, wired bandwidth abundance, continuing Moore's law improvements to computing, and energy constraints in mobile and sensor devices.

Table of Contents

1. Introduction

1.1. Design goals

The MobilityFirst architecture is centered around two fundamental goals: mobility and trustworthiness. The mechanisms used to realize these high-level goals in MobilityFirst are also mutually reinforcing, i.e., some of the mechanisms used to improve mobility also enhance trustworthiness. To appreciate this point, we begin with a recap of the high-level design goals that drive the design of MobilityFirst (and that are poorly met by the current Internet):

1) Seamless host and network mobility: The architecture should seamlessly support mobile devices as well as networks at scale. Mobility and the presence of wireless links should be considered the norm. In contrast, the current Internet is primarily designed with tethered hosts in mind, e.g., an IP address is used to identify a host/interface as well as its network location. This makes it cumbersome to support mobility (when a host's network location keeps changing) as well as multi-homing (when a host is simultaneously attached to multiple network locations).

2) No single root of trust: The architecture should not have a single root of global trust. In contrast, the current Internet has a single authority (ICANN) that must be trusted in order to reliably translate names to IP addresses.

3) Intentional data receipt: Receivers should have the ability to control incoming traffic and, in particular, be able to refuse unwanted traffic. In contrast, the current Internet largely treats receivers as passive nodes that have little control over the traffic sent to them.

4) Proportional robustness: A small number of compromised nodes must not be able to inflict a disproportionately large impact on the performance or availability of the rest of the nodes.

5) Content addressability: The network should facilitate content retrieval in addition to the ability to send packets to specified destinations.

6) Evolvability: The architecture should allow for rapid deployment of new network services.

1.2. Architecture Summary

MobilityFirst's architecture addresses the above design goals based on the following key components:

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

2) Decentralized name certification service (NCS): MobilityFirst decentralizes trust in name certification, i.e., different independent NCS organizations could attest to the binding between a human-readable name and the corresponding (public key) GUID. It is conceivable that the different organizations may disagree on the GUID corresponding to a name. End-users can choose which NCS(es) to trust and use quorum-based techniques to resolve disagreement between NCSes.

3) Massively scalable global name resolution service (GNRS): The GNRS is one of the most central components of MobilityFirst and is responsible for supporting seamless mobility at scale. The scale we envision is on the order of 10 billion mobile devices moving through about 100 networks each day, which corresponds to an update overhead of ~10 million/sec. In comparison, DNS, by design, relies heavily on caching and takes on the order of several days to update a record. Thus designing a massively scalable distributed GNRS is a key challenge in MobilityFirst.

4) Receiver-driven filtering and traffic engineering: MobilityFirst explicitly enables receivers to refuse unwanted traffic as well engineer incoming traffic in multihomed settings. The underlying mechanisms are based on a combination of capabilities (a deny-by-default or whitelisting approach) and filtering (a blacklisting approach). Clearly, these mechanisms are instrumental to combating distributed denial-of-service (DDoS) attacks. Furthermore, the ability to engineer incoming traffic across multiple interfaces for multihomed devices is valuable in mobile settings where a device may be simultaneously connected to multiple network service

providers that differ significantly in their price, performance, and power consumption characteristics.

5) Protocol design for proportional robustness: The MobilityFirst protocol stack is explicitly designed to ensure graceful degradation in performance in the presence of a fraction of compromised nodes. In particular, the protocols attempt to ensure that a small fraction of compromised nodes do not severely disrupt the performance or availability experienced by the rest of the network. We are particularly focusing on the design of the interdomain routing protocol, storage-aware intradomain routing protocol, and the end-to-end multipath transport protocol to ensure these properties.

6) Management plane for visibility and security: MobilityFirst relies on a management plane for logically centralized decision making. The management plane improves visibility into network usage patterns for operators, enables greater choice for end-users, and is valuable for monitoring and defending against attacks. Within each ISP, logically centralizing decision making via the management plane enables more robustness in the face of attack compared to a fully distributed control plane that is co-mingled with the data plane in today‘s Internet.

7) Content- and context-aware services: The network layer in MobilityFirst is designed to be content-aware, i.e., it actively assists in content retrieval as opposed to simply providing a primitive to send packets to specified destinations in today’s Internet. MobilityFirst achieves this by assigning GUIDs to content. These GUIDs are cryptographically verifiable, e.g., self-certifying hashes of the content, which allows a receiver to easily check the integrity of the content. MobilityFirst also extends the basic device and content GUIDs to more flexible groups of devices or users, e.g., all mobile devices in Central Park; or all taxis in Times Square, etc.

8) Computing and storage layer: Experience with the current Internet shows that it is imperative to design for evolvability. To this end, MobilityFirst routers explicitly support a computing and storage layer that enable rapid introduction of new, and possibly niche, services while minimally impacting the performance of the large majority of existing users.

In addition to the mobility-trustworthiness synergy indicated in the components above, there are several other examples. The NCS and GNRS can be used to improve privacy as well as attack resistance. Access control mechanisms in the NCS and GNRS allow users to hide their

current network location or expose it to a whitelisted set of trusted users only. Users desiring even greater privacy can (by buying into a paid service) rapidly create pseudonym GUIDs that are valid only for a limited time. Furthermore, the ability to mask one's network location naturally improves the ability to defend against DoS attacks.

It is also envisioned that economics-based mechanisms will be valuable both for supporting new mobile services as well as for improving accountability of resource consumption. For example, multicast primitives such as send this emergency message to all vehicles in Times Square can be better supported if resource usage can be carefully accounted for. Accountability dissuades potential attackers and aids in post-hoc forensic analysis to identify perpetrators.

2. Conventions used in this document

INFO (REMOVE): Include this section only if needed. Suggested wording.

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

3. MobilityFirst Network Protocol

The MobilityFirst network protocol that enables end-to-end communication can be summarized as below:

1. A network entity is associated with a human-readable name, by its human owner or corporation for example.
2. The human-readable name may then be registered along with other distinguishing attributes with one of several NCS providers to obtain a GUID for the entity.

3. With a GUID and valid network-use authorization (obtained through out-of-band channels), the entity may attach to a network (possibly multiple networks at a time - multi-homing) at a topologically addressable port defined by the network service provider. This establishes the entities attachment point(s) at a given time.
4. The network element at the service provider (wireless access point, base station, gateway router, etc.) will publish the association of GUID with one or more distinct network addresses (NAs) to the GNRS, with due authorization from the entity at the time of attachment.
5. A source network entity (*Source*) with intent of communicating with a destination network entity (*Destination*) will obtain the Destination's GUID through global or local resolvers run by a NCS or other third parties.
6. *Source* will use a network service API to specify its credentials (e.g., GUID), communication intent (such as transport type, delivery type, and request for in-network services), the message to send, and the *Destination*'s GUID.
7. The communication intent is captured by the protocol using a service identifier (SID) along with additional parameters to qualify the service. The SID passed with the message may be an aggregated value when multiple services are requested.
8. A message destined to a GUID may first be resolved to an up-to-date NA (or a set of NAs) using the GNRS. It is early binding when the NA is resolved at-source, and late binding when the resolution happens along the way.
9. The message is progressed through the network towards *Destination*'s NA in a hop-by-hop manner. The hop protocol is executed on suitably segmented blocks of the original message, with each block self-contained with complete routing information. The Hop protocol guarantees reliable transportation at each hop.
10. Routing elements within the network provide requested services (SID) for each routable block, including allocating any compute and storage resources to fulfill service requests and enable efficient end-to-end delivery. Services may be provided on a best-effort basis with due notifications to *Source* when unable to accommodate the delivery request in entirety.
11. To support successful delivery to a mobile *Destination*, the NA for in-flight blocks may either be progressively resolved or even re-resolved upon failure to find the *Destination* at the previously resolved NA.
12. Upon delivery to *Destination*, message integrity and originator's authenticity may be verified using the *Source*'s GUID and any signatures or similar mutually agreed upon mechanisms.

13. End-to-end signaling for reliability or flow control maybe implemented through extended functionality at transport or higher layers.

In the following sections, we outline architecture and protocol details that make up the above protocol.

3.1. Clean Name-Address Separation

MobilityFirst employs a clean separation of names of entities (network-attached objects) from their network addresses (points of attachment). It goes further than current TCP/IP Internet practices, however, in implementing this clean separation. Both IPv4 and IPv6 addresses combine functions of a network identifier with the topological address or locator of the network object. This introduces difficulties under mobility, and makes impossible communication continuity without redirection techniques (e.g., home agents in Mobile IP) that are often inefficient and present scalability challenges.

3.1.1. Name Certification Services (NCS)

An NCS will perform the GUID assignment for a network entity and maintain the mapping of the human readable name and associated attributes of the network entity to the assigned GUID. Existence of multiple NCS instances is expected and each may be domain specific. For example, a conglomerate of automobile producers may run an NCS that register and certify identities of automobiles equipped with communication devices. Though coordination among NCSs could guarantee uniqueness of assigned GUID, we do not postulate a framework or speculate on the possible organization of these services to avoid collisions. We believe, however, that the size of the GUID will render the probability of a collision to be insignificant. In the event of a collision, we expect GUID resolution services and other higher-level validation services to identify and bring to notice for out-of-band arbitration.

3.1.2. Network Identifier: GUID

In MobilityFirst, the network name of an entity is a globally unique identifier - GUID. As noted, an entity can be a host, sensor, service, application, content, etc. Further, to address several security-related issues seen with IP addresses such as hijacking and spoofing, the GUID is required to be a public key. The GUID is part of the packet header to enable self-certification and easy verification of sender authenticity.

Here are some possible sizes for a GUID based on currently accepted public-key cryptography standards:

1. RSA public key: 1024 - 4096 bits (2048, 3072 are current recommendations)

2. DSA public key: 1024 - 3072 bits (2048, 3072 are current recommendations)

3. ECC/ECDSA: 256 bits (for 128-bit security level)

These sizes are potentially a significant overhead when passed along with each packet - even when we consider jumbo MTUs. It is also a concern when considering efficient forwarding structures for a GUID-based routing fabric. However, two alleviating approaches are being considered. First, the routing header with a GUID need not accompany each packet/frame, and instead encapsulates a PDU (or chunk/block) whose size can be as high as a few hundred megabytes. The large PDU is fragmented into frames by the link data transport during transport to the next hop in the path, and re-aggregated there prior to handing to the routing layer to decide the subsequent hop.

Hash of GUID: A second approach is to pass only a compressed value such as a hash that still preserves uniqueness properties. For example, a 160-bit (20 byte) hash of the public key is still large enough to significantly alleviate any chance of collision. While this could significantly reduce overheads, especially for small PDUs, and reduce routing costs, implications on security properties afforded by passing the entire GUID needs to be considered. Furthermore, the hash representation is to done consistently across the set of services that interact with the routing fabric and also utilize the identity of the network endpoint.

3.1.3. Network Address (Locator):

Each attached network object is associated with one or more topological addresses usable by the routing fabric. The format of this address depends on the network architecture and in the general form can be nested to as many levels as there are levels in the network hierarchy. For example in a two-level network structure with global networks (corresponding to administrative or trust domains) each with its own internal routing structure, the network address is made up of a network domain identifier and local routing identifier.

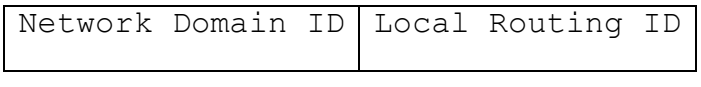


Figure 3.1.3: General structure of a topological address

The network domain ID is required to be a public key for the same reasons that a host's GUID is a public key, to address spoofing and hijacking concerns seen with IP addresses. The format of the local address is flexible, however, and may be determined entirely by the architecture and routing protocols deployed at the local domain. It can be a flat identifier such as the GUID, or a structured IPv6 address. Some example local routing identifiers and their corresponding sizes:

Local Routing ID	Format/Size
GUID - compressed or hash of public key	E.g., SHA-1 of key => 160 bits
Unique Local IPv6 Unicast Address	RFC 4193 => 128 bits

Table 3.1.3: General structure of a topological address

3.1.4. Multi-Homed Entities and Groups

Multi-homing, where an entity is simultaneously connected to multiple ports within single network or ports on different networks, is naturally available in MobilityFirst. A multi-homed entity can maintain a single GUID while having multiple NAs at a time. Protocol allows for source or network driven use of multiple delivery points for reliability or higher performance through bonding of interfaces/ports. Alternatively, a multi-homed node may establish and announce different GUIDs for each NA, with different delivery intents.

Groups of entities may similarly aggregate themselves under a single GUID to participate in group data delivery services supported by MobilityFirst network protocols, such as multicast and anycast. The process of group formation and management of group GUIDs itself is outside the scope of the architecture. Application services establishing a joint interest may interact with NCS and GNRS services to establish group identities and manage group membership.

3.2. Reliable Hop-by-Hop Block Transport

In the MobilityFirst network, data blocks are transported in a hop-by-hop manner reliably from one router to another. A data block can be large, variable in size and may even extend to an entire file (i.e., a few hundred megabytes or up to a gigabyte). Larger data blocks reduce the overhead of control signaling involved, but also require larger buffers. Since data may traverse network segments that differ in resource characteristics, the block sizes can also be negotiated when crossing such segment boundaries.

The hop transport is a key component in enabling efficient delivery in the presence of unreliable access networks where mobile hosts commonly experience disconnection and/or variable link quality. As data is progressed reliably hop-by-hop, storage at intermediate routers allow for temporary pause and subsequent continuation upon improved path conditions. This significantly reduces retransmissions and keeps end-to-end control signaling to a minimum. This is in contrast to the current TCP/IP network architecture where reliability is possible only through burdensome end-to-end signaling and consequently incurs severe underperformance under variable path conditions.

Finally, it is not necessary for a data block to be ‘aggregated, routed and segmented’ at each hop on the path. Alternatively, some hops may be ‘bypassed’ by tunneling packets to a downstream router when it is deemed unnecessary to buffer at intermediate points. This addresses to an extent concerns of ‘overbuffering’ within the network core where link/path qualities are normally stable.

3.3. Dynamic Name Resolution

As hosts move in physical space and associate with different access points (APs, BSSs, etc.) the topological address of the host changes. The network-association protocol updates the GUID-to-NA mapping in GNRS to reflect this. Since the GNRS is accessible to all network entities (hosts and routers), the up-to-date mapping of a host can be accessed at all times. Though, the end-point originating a message may prefer to establish the binding (i.e., resolving an NA for a destination GUID)in a certain manner. MobilityFirst allows for early- and late-binding approaches.

3.3.1. Early Binding

The host protocol stack includes GUID-services as a sub-layer of the network layer. Early binding happens at the sender host when the GUID service executes a GNRS lookup on the destination GUID to populate the destination NA field of the routing header. This is similar to the

binding in TCP/IP protocol stacks. The benefit of early binding, of course, is that there are no further resolutions of destination GUID. This, however, may potentially result in delivery failure if the destination host moves when the data is in-flight.

3.3.2. Late Binding

Alternatively, a sender host may request the network to perform the binding of the destinations NA and specify only the destination's GUID. Routers along the path progressively bind the NA to the given GUID, where progressive here refers to exploiting any inclusive relationship the destination host's network may participate in. For example, if the destination is attached to subnet A within network N1, a router outside of N1 may bind the data packet to N1. Then, on entering N1 the packet is subsequently bound to subnet A. Local resolutions are performed on local versions of the GNRS or alternate mechanisms suitable for local scale.

Under local mobility of destination (within network or subnet), progressive binding removes possibility of delivery failures. When host moves outside of a network, late binding implies re-binding the packet to the up-to-date NA. This is done at the failure router by re-resolving the GUID through a GNRS lookup.

3.3.3. Hybrid Name-Address Routing

With network elements able to dynamically resolve the network address of a destination, routing in MobilityFirst can proceed with either the GUID or the pre-resolved NA. Sources may provide just the destination GUID or a resolved NA within the network header. Furthermore, in a local scope with scale permitting the routing may proceed entirely using just the flat GUID of the host.

3.4. Storage-Aware and Edge-Aware Routing

MobilityFirst proposes to two key principles in the design of routing protocols for the future Internet. First, we harness the ready availability of storage and compute resources in the future (i.e., inexpensive) to address delivery challenges to wireless and mobile access networks. Routers will actively hold (active implies upstream routers will hold rather than at downstream portion of path) data packets when it is determined that access links and paths to mobile hosts are facing intermittent low qualities. The decision to store or forward is determined by analyzing link/path qualities observed from the vantage point of the router to the destination network or node. If near-term quality is determined to be equal or better than those observed over a large window, then the router forwards packets to the

destination. If not, the packets are temporarily held until better qualities return. Multiple metrics to estimate path/link qualities are under investigation with the expected transmission time (ETT) determined to be a useful indicator. Furthermore, routers advertise the status of storage resources to other nodes in the network to enable a cooperative management of network-wide resources.

The second key routing principle is to extend the intra-domain sharing of network state to beyond network domain boundaries. Similar to pathlet routing proposal from Godfrey et al, the objective here is to provide more in-depth information about the internal structure, resource and traffic conditions of a network to enable upstream networks and routers to make informed choices in deciding transit and delivery paths. Inter-domain extensions to our storage-aware protocol represent the network as a collection of virtual 'aggregation' nodes we call *a-Node*s, which are interconnected by *v-Link*s. This abstracted graph representation, annotated with capacity, availability and traffic information, is shared with other neighboring networks using what we term a 'telescoping' approach to address scalability concerns in the control plane. Telescoping involved summarization and/or filtering of control data along both time and space dimensions to limit control traffic to manageable levels. The precise composition of these dissemination packets, the specific telescoping algorithms, and the tradeoffs thereof are currently under investigation and await formalization.

3.5. In-Network Delivery Services

MobilityFirst requires that certain services be natively implemented by the network to support proposed architectural features. For example, routers are to implement lookup of locator(s) for a GUID to enable dynamic name-address binding. The following are the basic delivery services proposed for baseline deployment:

- Unicast
- Multicast
- Anycast
- Stream or Real time
- Delay Tolerant
- Content Request
- Content Response

- Compute Layer Processing
- Acknowledge on Store
- Acknowledge on Delivery

3.6. Service-Oriented Routing Header

Based on the above requirements, we propose the following packet/header format for MobilityFirst:

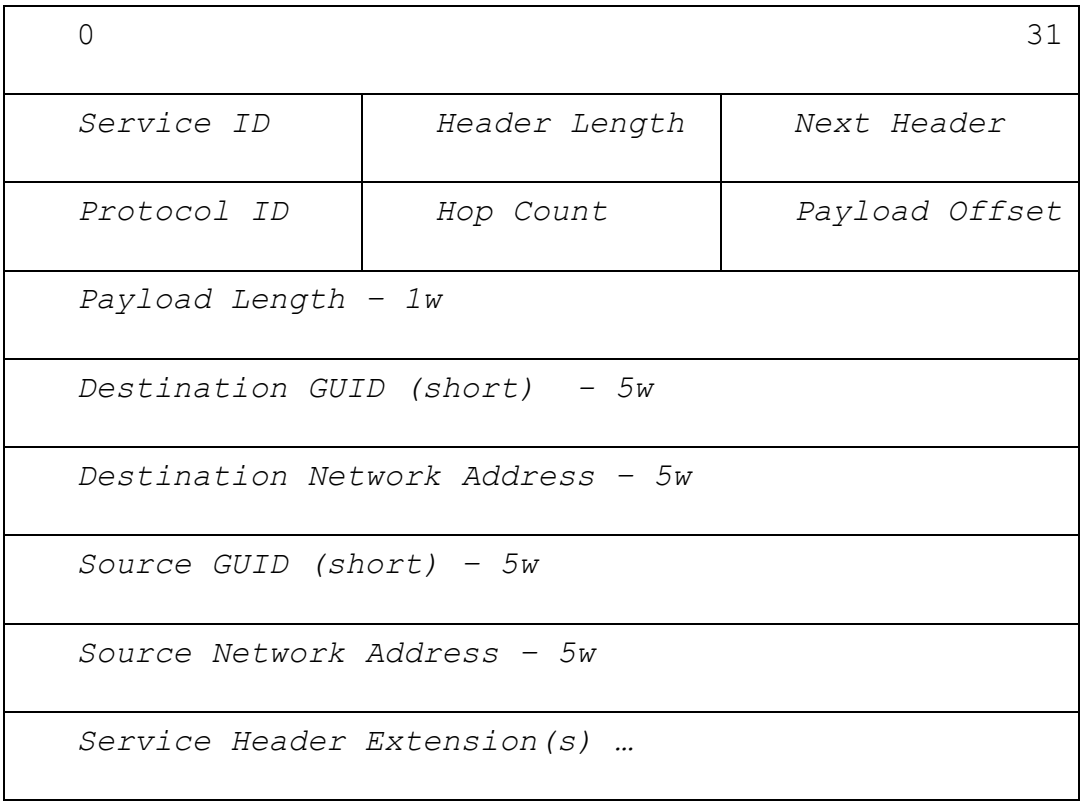


Figure 3.6: MobilityFirst Network Header

Where the fields are defined as follows:

Service ID	Identifies the specific processing or delivery service (or set of services) to be applied to this packet

Header Length	Length of header excluding any extension headers.
Next Header	Offset of the next service header. '0' indicates no more headers
Protocol ID	Demux for protocol running above MobilityFirst routing
Hop Count	Maximum number of hops after which the packet is dropped. This is decremented by 1 at each hop.
Payload Offset	Offset where the payload begins
Payload Length	Length of payload
Destination GUID	Destination endpoint identifier. When short, it's the compressed form of destination GUID.
Destination Network Address	Topological address for destination. This can be blank when it's not yet been resolved.
Source GUID	Source endpoint identifier. When short, it's the compressed form of the source GUID
Source Network Address	Topological address for destination. Optional and can be left blank.
Service Header Extension	Optional to define additional packet-handling services and corresponding parameters
Payload	Data or begin of header of next protocol above MobilityFirst routing

3.6.1. Service Header Extension

When additional parameters are required to be passed for specific packet processing or delivery services, header extensions can be

utilized. The general format of an Service Header Extension is shown below.

Service ID	*Header Length*	*Next Header*
Custom Service Fields		

Figure 3.6.1: General format of a service header extension

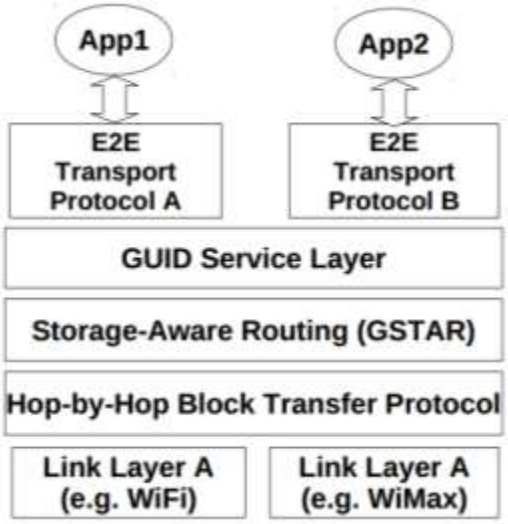
The format for the service specific fields can be defined by the individual service, and has no obvious restrictions other than maybe a maximum limit on the overall length of the header extension (TBD). Sample services headers for common use cases are shown in Appendix A.1.

SID Encoding

When the packet invokes more than one service, it would be desirable to avoid tagging on extension headers for each service. Since several of the above basic services do not require parameters beyond those that are already part of the normal header, we go for a simple and compact 'bit-position' encoding of the SID field - each bit of the field one of these basic services. A 16-bit allocation for this field allows for 15 basic services while saving 1 bit for encoding identifiers of non-basic services. SIDs of non-basic services then follows the pattern 0x8xxx. The encoding of identifiers for basic and other services using this scheme are shown in Appendix 1.B.

4. Protocol Stack

Figure 4: Layering in MobilityFirst Protocol Stack (host protocol



stack shown)

Figure 4 shows the layers of MobilityFirst protocol stack. The stack matches quite closely the current Internet stack layering with notable deviations. Layers in-between the transport and link layers shown above make up the traditional network layer of the stack. The IP-based thin waist of the Internet stack is replaced by a GUID-based network layer that forms the new thin waist. The hop-by-hop data transport shown in the figure is more akin to the link data transport of the traditional link layer. In-network elements (i.e., elements other than end-hosts such as routers) implement layers network and below, while end-hosts in addition implement transport and higher functionality, including a network service API for endapplications. The network API enables a high-level messaging interface with GUID-based end-point addressing with the ability to distinctly (or in combination) request MobilityFirst in-network services.

4.1. Transport Layer

The transport layer is responsible for taking a message and segmenting it into large data blocks we refer to as chunks or PDUs. A chunk represents an autonomous data unit that can be routed through the network, and contains the header with authoritative routing information - the destination GUID. A chunk can be as large as a few hundred megabytes, but the size can also be negotiable with the next-hop or even the final recipient of the message - to accommodate resource differences. As in traditional sockets, applications can choose among multiple supported transport protocols through options in the messaging interface.

4.2. Network Layer

GUID Services

A main function of this sub-layer is to provide lookup services for resolving the NA for packet with a destination GUID. It initiates a resolution by contacting local GNRS agents (host daemon or LAN/gateway service) to perform the GNRS lookup operation. In a router element it also provides resolution for GUID-addressed in-network compute services (e.g., content cache, context/mobility services, etc.) registered with the forwarding plane and requested by the data packet.

A second important function for this layer is to announce network reachability for each endpoint. Objects interested in establishing network presence can indicate this intent to the stack through the

network API. The GUID service layer initiates an association protocol with the network gateway (e.g., access point, BSS) for each such object which results in a network attached object whose identity and location (<GUID, NA>) are published to the GNRS. Note that the association message is duplicated by the manager on each connected network interface known to the network layer. Layer also manages life-cycle an attached object, sending periodic keep-alives and a disassociate notification session termination or managed disconnection.

In contrast to current Internet stack's use of transport layer ports, MobilityFirst can use a GUID to both identify an application and its reachability at the network level. Note that this is in addition to any GUID(s) assigned to the host or device and decouples an application's identity from its current host, enabling easy migration. However, if limited in number of GUIDs, an application label can be employed to distinguish a specific instance or endpoint. The stack assigns and maintains a demux identifier - appID - for each such endpoint. An appID is added to an outgoing message and establishes the corresponding endpoint for incoming messages.

Storage-Aware Routing

The primary function of this layer is to determine the routes when multiple interfaces exist. In the current IP-stack, this decision is made based on routing tables and usually the traffic goes to the Internet through the interface configured as default gateway.

In MobilityFirst, decision of which interface to use for the incoming and outgoing traffic is made at this layer based on context information instead of routing tables. Here, the context information includes application performance, battery usage and monthly data plan capacity. In addition, the DTN-style ad hoc routing is supported by the network layer. Thus, client stack would work in environments lacking in infrastructure such as access points and base stations, and still proceed to progress packets towards destinations in a multi-hop manner.

MobilityFirst routers implement a generalized storage-aware routing protocol (GSTAR) that seamlessly adapts across different networks with varied degrees of connectivity including, wired, wireless and even DTN-type networks where partitioning and disconnections are common. It exploits storage available at each router to overcome disconnections and intermittent link quality variation, especially in mobile wireless or congestion scenarios. If quality of next-hop link or path is unacceptable, the chunk is held in a local store until quality improves. Chunks are also held in the store during host disconnections.

4.3. Link Layer

The MobilityFirst link layer is made up of two sub-layers. First, is the traditional link layer, and second, is hop-by-hop block data transfer layer. The functionality of the hop layer is to:

1. Take the chunk/block from upper layer and fragment it into data packets suitable for PHY layer
2. Implement control signaling for reliable transfer of all packets from/to node at other end of link
3. On the receive side, to receive and aggregate all data packets belonging to a chunk from the upstream node and deliver to the network layer

A chunk ready for transfer is fragmented and transmitted as MTU-size frames to the next hop node. The corresponding link layer at the next hop aggregates the entire chunk before passing it to the routing layer to be routed on either the GUID or NA. A chunk that fails to transfer to the next hop is handed back to routing layer for rerouting or temporary storage.

4.4. Network Service API

The following are the basic methods supported by the network API for application end points to declare their identity, their communication intent including transport, security and delivery options, and request any in-network services supported by the extensible service architecture.

open (src-GUID, profile-opts)	This sets up self-identity and customization of the stack incl. transport and security options applicable for the session (i.e., until a close). Profile options are passed in the URL parameter passing style. A handle representing the created network endpoint is returned for invoking other methods
attach (handle, GUIDs)	This method is to announce network reachability for specified GUID(s). The network layer initiates an association request for each GUID in turn attaching it to the network and publishing an entry to the GNRS

send (handle, dst-GUID, message, len, svc-flags, svc-opts)	Applications send data as messages - application PDUs. There is no limit on the size of the message, except as limited by system resources. The dst-GUID may be any network attached entity including a host, group or context. The svc-flags parameter defines the set of network services requested in delivering the message to the destination. Some options include: MULTICAST, ANYCAST, CONTENT CACHE, MULTIPATH, DTN and REALTIME. svc-opts define custom arguments to chosen services in URL parameter passing style.
recv (handle, buffer, len, GUID-set)	Applications can receive messages by passing pre-allocated message buffers to the above API. The optional GUID-set parameter contains the set of GUIDs that the application intends to limit receipt from. On receipt of a valid message (and duly loaded into the buffer), a receipt descriptor with message details is returned to the application.
get (handle, content-GUID, buffer, svc-flags, svcopts)	Content-centric applications may exploit native network support for content discovery and retrieval by content by its GUID. If svc-flags parameter includes ANYCAST then the content retrieval is attempted from the closest source (from among available replicas).
close (handle)	This clears any state set up for the application within the stack and the network including the network attachment state that is removed by initiating a disassociation request for the set of associated GUIDs.

The MobilityFirst architecture aims to integrate a global name resolution service (GNRS) as a basic network-layer service which can be efficiently accessed both by end-user devices and in-network routers, base stations and access points. This concept is illustrated in Figure 1 which shows the layering of functionality in the proposed MobilityFirst architecture. In this approach, a human-readable name such as ”Sue’s laptop" is mapped to a GUID through one of many possible application level services deployed by the network provider or independent third-party providers. The GUID is then assigned to the mobile device (or other network-connected object) and entered into the network-level GNRS service shown in the figure. The GNRS is a distributed network service is responsible for maintaining the current bindings between the GUID and network address(es) (NA’s). Mobile devices (or routers at their point of attachment) update the GNRS with current NA values resulting in a table entry such as <GUID: NA1, NA2, NA3, optional properties>. The technical problem addressed here is that of realizing a scalable GNRS service with ~10 Billion GUID entries (i.e. network-attached objects) with lookup latencies fast enough to support anticipated mobility speeds and application usage patterns. We emphasize that since the GNRS can also be queried by the in-network routers for dynamic GUID:NA resolution for in-transit packets, low latency in the query response procedure is a critical requirement for the design.

6.1. Service Interface

GNRS provides three name resolution primitives: Insert, Query, and Update.

6.1.1. Insert (and Update) GUID-to-Locator Mapping

Mobile devices or routers at their point of attachment invoke an Insert command when a new GUID first appears in the following format:

```
INSERT(GUID, NA1,NA2,..., [OPTIONS])
```

When the devices have any changes to their network attachment points, an Update is called:

```
UPDATE(GUID, NA1,NA2,..., [OPTIONS])
```

6.1.2. Query Locator(s) for a GUID

Mapping of a GUID of interest can be queried by a network entity through a Query with the following format:

```
QUERY(GUID,[OPTIONS])
```

GNRS returns the GUID to locator mapping which can then be used in the network layer for routing purposes.

6.2. GNRS Protocol

The GNRS service is accessed by both end-hosts as well as network elements such as routers, gateways, and access points. The base protocol uses lightweight, connectionless messaging and relies on client-side retries for robustness. We also propose a secure version of the protocol for deployments and use cases that require it.

6.2.1. Hierarchical Organization

6.2.2. Security Considerations

6.3. Scale and Performance Considerations

+ Low Latency: Since mobility is directly handled using dynamic identifier to locator mapping, latency requirements are much stricter in host-based schemes.

+ Low Staleness: Fast mobility support also requires that the identifier-locator mappings be updated at a time-scale smaller than the inter-query time.

+ Storage Scalability: Since flat identifiers would lead to substantially more number of identifier to locator entries, the mapping scheme needs to scale to the order of billions of entries instead of thousands

6.4. Realization 1: DMap: A Shared Hosting Scheme using Single-hop In-network Hashing

In DMap, each GUID→NA mapping is stored in a set of ASs. Each GUID is directly hashed to existing network addresses and its mapping is stored within the ASs corresponding to these network addresses.

To perform the mapping service for a given GUID, DMap applies K(K > 1) hashing functions onto it to produce a list of K network addresses, which are IP addresses in today's Internet, and stores the GUID→NA mapping in the ASs that announce those network addresses. By doing so, DMap spreads the GUID→NA mappings amongst ASs, such that an AS will host mappings of other ASs, as well as have its mappings hosted by others. A key advantage of this shared hosting approach is that it allows the hosting ASs to be deterministically and locally derived from the identifier by any network entity. DMap is simple yet efficient. It leverages the routing infrastructure to reach the hosting AS in a single overlay hop; it does not require a home agent, unlike mobile IP and existing cellular networks. Further, the potential shortcoming of the direct mapping scheme, lack of locality, is addressed by having multiple copies of the mappings that are stored in multiple locations. We further improve the design by including a local copy of the mapping within the AS that the GUID is residing in (this AS may change as the host moves).

Let us suppose host X, with GUID Gx, is attached to NA, Nx. X first sends out a GUID Insert request, which is captured by the border gateway router in its AS. The border gateway router then applies a

predefined consistent hash function on Gx and maps it to a value IPx in the IP space. Based upon the IP-prefix announcements in the BGP table, the border gateway router sends the Gx -> Nx mapping to the AS that owns IPx. Later, suppose host Y wishes to look up the current locator for GUID Gx. Y sends out a GUID Lookup request. After the request reaches Y's border gateway router, the border gateway runs the same hash function to identify the AS that stores the mapping. Every time when X changes its association and connects to a different AS, it needs to update its mapping by sending out a GUID Update request. Update requests are processed similarly as insert and lookup requests. Using the above approach, a GUID's mapping is hashed to a random AS, without considering the locality between the GUID and its lookup requests. This lack of locality may potentially lead to unnecessarily long lookup latencies. Thus, instead of storing a mapping at only one AS, we consider having K replicas of the same mapping stored at K random ASs. Having K replicas can significantly reduce the lookup latency as the requesting node can choose the closest replica (e.g., based upon the hop count between itself and the hosting ASs). Meanwhile, it will not have a big impact on the update latency as we can update the replicas in parallel. With K mapping replicas, the lookup latency becomes the shortest latency among the K ASs, while the update latency becomes the largest among the K ASs.

6.5. Realization 2: Locality-Aware Distributed Name Resolution Service - UMass Amherst Scheme

7. Generalized Edge-Aware Routing

The MobilityFirst routing component is designed around the observation that there is a fundamental paradigm shift towards mobile communication. Therefore, mobile devices and their associated applications must be treated as first-class Internet citizens, and have built-in support at the network layer. In this section, we will first briefly explore the challenges of mobility and present a set of guiding principles used in our routing techniques. Next, we present GSTAR, a generalized storage-aware routing system, and show how both global- and local-scale routing can support the challenges brought about by mobility.

Once the PDU has been delivered to the destination network, local storage-aware routing techniques can be used to deliver to the final host.

MobilityFirst uses a two pronged approach for intra-domain routing that is capable of quickly responding to link quality changes for nearby nodes as well as remaining robust in the face of disconnection and network partitioning. At a high level, individual routers maintain two types of topology information, one useful for responding to fine-grained changes to links and nodes within the router's current partition, and one useful for responding to course-grain changes to connection probability for all nodes in the network.

The intra-partition graph is formed by collecting topology messages that are periodically flooded by all nodes in the network. These topology messages contain time sensitive information about the link quality for each of the node's 1-hop neighbors. These are transmitted per interface, allowing for in-network multi-homing. Since the messages are flooded, and hence immediately broadcasted and dropped, they will not traverse across partition boundaries. This allows all nodes in the network to have an up-to-date view of the current link qualities within its current partition. In addition to storing current link qualities, all routers maintain a history of link quality information received in the past, which, as will be shown, is useful for routing decisions. If control messages have not been received from a particular node for some period of time, and hence its long term link qualities have become low, a router may assume that node has left the partition and remove it from the graph.

The DTN graph is formed by collecting topology messages that are periodically epidemically disseminated by all nodes in the network. Epidemic dissemination, where control messages are carried by

intermediate nodes, is a common technique used in delay-tolerant networking, and allows messages to cross partition boundaries. In essence, these messages are not immediately dropped, but rather carried for a long period of time such that if a node moves from one partition to another, it can ferry messages between the two. These topology messages contain time insensitive information about connection probabilities between the source node and all other nodes in the network. This graph allows a node in the network to be aware of the general connectivity patterns of all nodes, even those outside of its current partition.

These two graphs can then be used together to help route messages to their destinations. For a given message, a router first checks its intra-partition graph for the destination. If the destination

exists, then the router will then choose the best path from multiple ones by considering the short term link qualities for nearby hops and the long term link qualities for hops further away. Given that path, if the short term link quality for the next hop is much greater than the long term link quality for the next hop, then the router should immediately forward the data to take advantage of the abnormally good link. On the other hand, if the short term link quality is abnormally bad, then the router should store the message and re-evaluate later. A pictorial example of the routing decision graph, using the estimated transmission time (ETT) as the routing metric, can be seen in Figure ?

Figure ? - Intra-Partition Routing Decision

Figure ? - DTN-graph Routing

If the destination is not found in the intra-partition graph, the router tries to make progress along the DTN graph, which contains a general overview of connectivity throughout the network. The router will compute all shortest paths according to that graph, and attempt to forward a replica of the message to all 1-hop neighbors along those paths. In essence, this DTN-style approach attempts to make use of readily available storage to bridge partitions in the network. An example is given in Figure ?, where node S forwards a replica of a message destined for node D to both nodes A and B. Note that this message is not copied to node C, as this would not progress the message. It is important to note that this framework is flexible enough to incorporate many existing DTN routing protocols. For instance, the protocol-specific metric being used to capture connection probability can be used as the MobilityFirst DTN ‘link-quality’ metric that is epidemically disseminated to form the DTN graph. One example is to utilize the average availability metric, described in, to capture the historical percentage of time two nodes were connected.

Furthermore, replication rates can be determined by the DTN protocol in question

MobilityFirst is an ideal platform for pervasive computing because of following reasons:

A. MF is a computing grid: every node of MF offers storage and computing capacities to GUID identified networked objects (resources).
B. MF is a content distribution network (CDN): MF network layer offers a hop-by-hop transport that distributes the data of GUID identified networked objects, with native multicasting and anycast supports.
C. MF provides a secured framework based on public key: trust can be built between two parties based on their GUIDs. One way trust (e.g. a consumer trusts a bank) needs no end-to-end hand shake protocol. GUID is self-certifying.
D. MF provides a simple economic model: GUID identified network objects can trade their network resources (data and/or services) based on the trust built upon GUIDs.

MobilityFirst, as a pervasive computing platform, can enable M2M applications to share their data or middleware services as GUID identified network resources at MF core network layer, breaking the boundaries of isolated information islands built by traditional vertical market M2M applications.

Data and services owned by one M2M application, with GUIDs as identities, can be distributed and/or executed on MF routers and are directly accessed by other user applications. MobilityFirst is a true Internet of Things (IoT) architecture.

11.1. Naming Considerations

In MobilityFirst architecture, physical objects in M2M applications, identified through sensors (or tags, actuators), are assigned with GUIDs.

Figure 2. GUID structure

As shown in Figure 2, a GUID consists of a public key of the owner of the object. The generation of the public key must follow the suggestion based on the specification in section ??.

As an extension, GUID may contain a suffix of sequence number differentiating multiple networked objects belong to one owner. A suggested size of the suffix is 2-4 bytes (subject to further debate).

The name assignment process, including publishing, updating and lookup, must follow the specification in section ??.

Sensor Data

Sensor data can be considered as very low rate streaming media. It has data values with time stamps. Therefore, in general, a data block from a sensor can be expressed in following format.

Figure 2. GUID identified Sensor Data

11.2. End-point Resource Considerations and Function Offloading

12. MF Use Case 3: Support for Context Applications

Context is an aspect or quality of some object, function or relation (hereafter referred to as the 'subject') that provides additional information or inference about it that is not present or represented in the subject itself, but is pertinent to and indicative of its functionality, status or role to such a degree that it would benefit the subject to be able to either sense, react to, or be considered alongside this additional, contextual information. Contextual information is also not entirely representable by or in the subject. Otherwise, it would be a feature or a function element of that subject.

Existentially, context is:

1. a property of relations between subjects
2. defined dynamically as subjects change
3. momentary for a given measurement and environment
4. instantaneously created when subjects interact

Due to the exigencies of discrete computation context needs to be quantified as:

1. information relating to some detectable quality of a subject
2. a set of relations defined before they occur
3. a set of static relations whose results change
4. a quantifying relation separate from the process that causes it

Context Services within the FIA are meant to be used to instantiate an arbitrary network overlay so that any discrete function on quantifiable data can supply a target for a data operation or meta-operation. Context Services supply an interface to create the constraining functions, create and maintain data endpoints, and compute constraining functions using data in order to determine a network target to act upon.

12.1. Representation

12.2. Architecture

We propose that context services in MobilityFirst may be realized by three alternate architectures, which differ in the extent of support afforded by the network.

12.2.1. Arch 1: Direct client-driven context

- client queries CRS for a list of GUIDs for compatible context servers
- client contacts context server directly and sends data to it

The client uses the context service to find the context server it requires, and then conducts all other interactions with the server itself. In this case the initial message that the client sends needs to be sent to the correct server, so the client needs to know its address. Rather than having direct access to the context resolution service, what the client needs is a way to instruct the network to not deliver its message, but to send back to it the addresses of the intended recipients. The context service computes the context query the client sends, however instead of forwarding the data, if the client sets the lookup option, the context service will instead send the addresses of the recipients to the client - allowing it to contact the context server(s) directly.

12.2.2. Arch 2: Direct service-driven context

- client either:
  - dispatches a send to a target context query
  - dispatches a send to a target context GUID

- context service:
  - computes the context query and forwards the data, on behalf of the client, to the matching GUID(s)
  - computes the context query the context GUID references, and forwards the data, on behalf of the client, or the matching GUID(s)

In this model, the client sends a data operation targeted at a context request or GUID bound to a context request to the context service, which computes the request and carries out the request on behalf of the client. In this case the client is entirely insulated from the computation of the context. In the case of a context GUID, the client need not be aware it is making a context request.

12.2.3. Arch 3: Indirect network-driven context

- client registers a persistent context subscription with the context service by issuing a get data request on a context query or context GUID
- context server dispatches a send data to a blank target, but specifies a contextual description of the data
- untargeted data is relayed to the context service, which resolves the description, finds which clients have matching queries outstanding, and issues a send data to them on the server's behalf

In this model, the client knows a-priori how to describe the type of context it wants, but needs to make a less rigid request. If the client wants to receive data over time rather than enact a specific data operation, or set a time-based or standing request for a type of context that is currently unavailable, but may be available at some point in the future, it needs the capability to submit a specification that can be filled at any time. Contextual data producers can instrument their data for these type of clients by posting send data requests with a blank target, but with a contextual description of the data. The network would forward these requests to the context service, which can compute the clients' requests and determine if any match the incoming descriptions. If the descriptions of any match, a data operation to each matching client is started with the expectation that it will be a DTN-type stored delivery.

12.3. Operations

Two operations, each with two modes:

12.3.1. Send data:

Requires target specification and data payload. Target specification can be a GUID, context GUID, or an alias to context description of payload (SID).

mode 1: actuation: look up target and relay data to it

mode 2: information/query: look up target and get all addresses that the data would have been sent to

12.3.2. Get data:

Requires target specification and data specification, where a target can be a GUID, context GUID, or an alias to context description of payload (SID), and the data specification can be a contextual description of the data.

mode 1: actuation: look up target and send data request to it

mode 2: information/query: look up target and get all addresses that the request would have been sent to

12.4. End-to-End Protocol

13. Discussion of Open Issues

14. Conclusions

<Add any conclusions>

15. References

INFO (REMOVE): Authors can use either the auto-numbered references OR the named references; typically, these would not be mixed in a single document. This template includes both examples for illustration of the two variations. Named references are preferred (e.g., [RFC2119] or [Fab1999].

15.1. Normative References

INFO (REMOVE): Normative refs are references to standards documents **required to understand this doc. These are usually Standards-**

track and BCP RFCs, or external (IEEE, ANSI, etc.) standards, but may include other publications.

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

15.2. Informative References

INFO (REMOVE): Informative refs are those that are not standards or standards not required to understand this doc. These are usually informative RFCs, internet-drafts (avoid if possible), and other external documents.

[3] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[Fab1999] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

16. Acknowledgments

<Add any acknowledgements>

INFO (REMOVE): The author of this template would appreciate if you would keep the following line in your final IDs and RFCs:

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. MobilityFirst Packet Header Definitions

A.1. Service Header Extensions for Common Services

PKI Cryptography Extension Header

Service ID = [0x8000]	*Header Length*	*Next Header*
Source GUID = [PEM encoded Source Public Key]		

Source Routing Extension Header

Service ID = [0x8001]	*Header Length*	*Next Header*
Next Destination Offset		
Intermediate destination 1 (e.g., GUID or NA)		
…		
Intermediate destination N		

A.2. Hexadecimal SID Values for Proposed Services

Service	Hexadecimal value (16-bit encoding)
Unicast	0x0000
Multicast	0x0001

Anycast	0x0002
Block Transfer	0x0004
Stream or Real time	0x0008
Delay Tolerant	0x0010
Acknowledge on Store	0x0020
Acknowledge on Delivery	0x0040
Content Request	0x0080
Content Response	0x0100
Compute Layer Processing	0x0200
Public-Key Cryptography	0x8000
Source Routing	0x8001

Authors' Addresses

<Firstname> <Lastname>
<Affiliation>
<Address>

Phone: <optional>
Email: <Your email address>

<Firstname> <Lastname>
<Affiliation>
<Address>

Phone: <optional>
Email: <Your email address>