Architecture Summary Document

MobilityFirst: A Robust and Trustworthy Mobility-Centric Architecture for the Future Internet*

Contact: D. Raychaudhuri, ray@winlab.rutgers.edu
WINLAB, Rutgers University

Abstract:

The MobilityFirst project was started in Sept 2010 as a collaborative, multi-institutional research initiative under the NSF FIA (Future Internet Architecture) program. The overall goal of the project is to design and validate a clean-slate mobility-centric and trustworthy architecture for the future Internet. The scope of research includes specification of the proposed new network architecture, detailed design and verification of key protocol components, analysis of economic and policy aspects, evaluation of network security and privacy, system-level prototyping and validation of the network as a whole, and "real-world" testbed deployments for evaluation by application developers and end-users.

This document provides an overview of the MobilityFirst architecture as of July 2012, and is intended as a supplement to the annual progress report. After briefly outlining the original design goals of the project, we provide a discussion of the main architectural concepts behind the network design, identifying key features such as separation of names from addresses, public-key based globally unique identifiers (GUIDs) for named objects, global name resolution service (GNRS) for dynamic binding of names to addresses, storage-aware routing and late binding, content- and context-aware services, optional in-network compute layer, and so on. The building blocks of the protocol are then identified in terms of two distinct layers, a GUID-based meta-services layer and a network address (NA) based core transport services layer, and each major protocol module is described briefly. This is followed by an outline of the MobilityFirst protocol stack as a whole along with an explanation of how the protocol works at end-user devices and inside network routers. Examples of specific advanced services supported by the protocol stack, including multi-homing, mobility with disconnection, and content retrieval/caching are given for illustration. This is followed by a summary of the MobilityFirst protocol spec as currently defined is given, including some details on packet formats, syntax and semantics. In conclusion, we provide an outline of the protocol layers in the MobilityFirst architecture and discuss the features of the network service API.

Draft v1.0, July 2012

* Research supported by NSF Future Internet Architecture (FIA) grant CNS-1040735

1. High-Level Architecture & Design

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 some of the high-level design goals that drive the design of MobilityFirst (and that are not adequately 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 require a single global root of 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 Concepts:

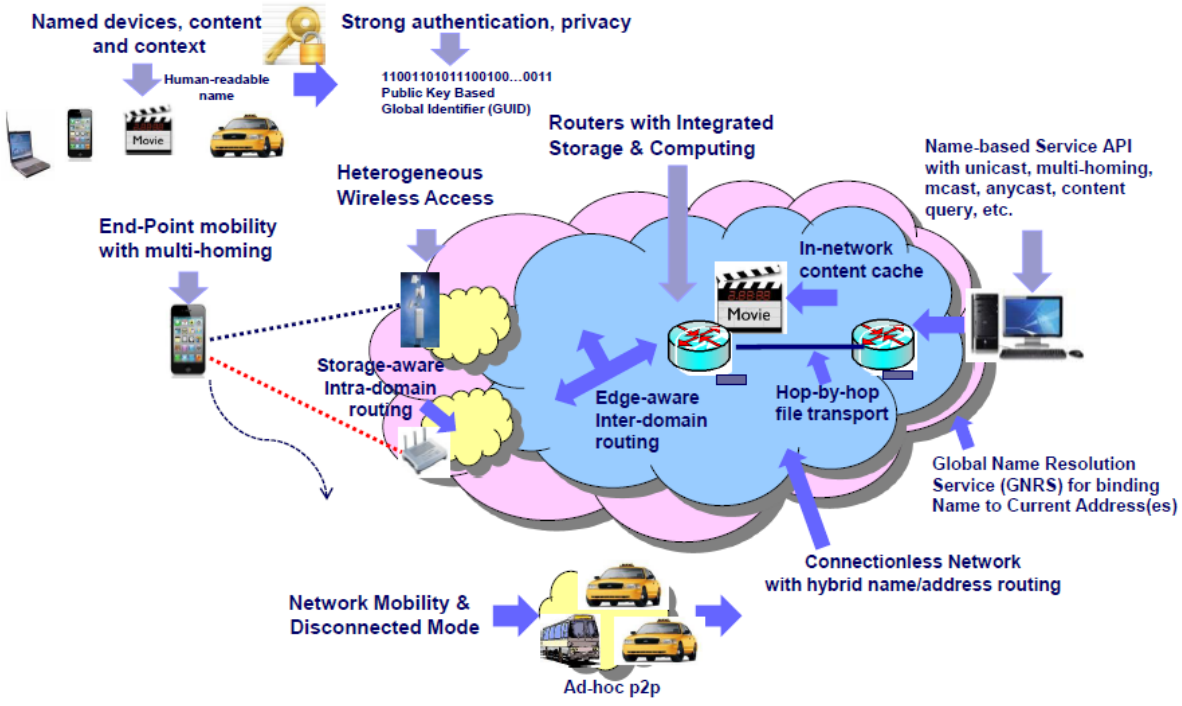

Fig. 1.2.1 Major design features of MobilityFirst architecture

The architectural research conducted in this project started with the abstract requirements outlined above and then considered high-level protocol design approaches towards realizing these goals. The

MobilityFirst team met on a regular basis during the first year of the project to discuss architectural ideas and design trade-offs, leading to a gradual convergence towards an overall system concept and a related set of protocol elements. The architecture which emerged from these discussions (see Fig 1.2.1) is centered around a new name-based service layer which services as the "narrow-waist" of the protocol – this name-based services layer makes it possible to build advanced mobility-centric services in a flexible manner while also improving security and privacy properties. The name-based service layer uses the concept of "flat" globally unique identifiers (GUIDs) for network attached objects, a single abstraction which covers a broad range of communicating objects from a simple device such as a smartphone, a person, a group of devices/people, content or even context. As shown in the figure, named objects are assigned a secure public key GUID by a name certification service at the upper layers of the protocol stack. Network services are defined by the source and destination GUID along with a service identifier (SID) used to specify the delivery mode such as multicast, anycast, multi-homing, content retrieval or context-based message delivery. A hybrid name/address based routing scheme is used for scalability, employing a fast global name resolution service (GNRS) to dynamically bind the GUID to a current set of network addresses (NAs). The GNRS is a central feature of the architecture, enabling on-the-fly binding of names to routable addresses as needed for dynamic mobility, disconnection or cloud migration scenarios. Actual delivery of data through the network is based on hop-by-hop transfer of large files with in-network storage to deal with link quality variations and disconnections due to mobility. The corresponding intra- and inter-domain routing protocols used in the network have new features such as edge network awareness and late binding capabilities needed to achieve the design goals. The overall philosophy of the design is thus back to the basics of packet switching with hop-by-hop routing of entire data files with a minimum of in-network state – the packets themselves carry destination and service identifiers that can be dynamically bound to routable addresses during transit through the network.

Some of the major design features of the architecture are discussed further below:

Separation of Names and Addresses: MobilityFirst cleanly separates human-readable names, globally unique identifiers, and network address locators. In contrast to the current Internet, the human-readable name can be managed and assigned to a unique GUID by multiple name certification services (NCSs) without a global root of trust. No coordination is required between NCS providers because the GUID space is very large with arbitrarily low probability of duplication. GUIDs assigned to network objects are mapped to a set of network addresses (NAs) or locators corresponding to the current points of attachment.

Security based on Public Key Based Globally Unique Identifier: The GUID assigned by an NCS is a public key identifier that provides a uniform cryptographic basis for authentication and security services in the network. The use of public keys also makes it possible to support anonymous or self-certifying end-points if so desired.

Name-based Network Service API: The service API in MobilityFirst is based on the names of source and destination network objects, rather than on the network addresses/interfaces. This allows us to build abstract services involving multi-homed devices, groups of objects, named content, etc. Because a single network object may consist of multiple devices or have multiple interfaces, the service abstraction is inherently multicast in nature and is thus well matched to the wireless environment.

Global Name Resolution Service: The proposed architecture uses hybrid name/address based routing in the interest of scalability. The total name space of attached network objects is perhaps ~10-100B, while the number of unique routable networks tends to be much lower ~0.1-1M, thus making it expedient to map GUIDs to NAs and then route using NAs where available. This approach requires the existence of a global service (which we call the GNRS) that dynamically

maps GUIDs to NAs. Clearly, scale and speed of the GNRS are critical design requirements for the proposed approach to be viable for mobility services.

Mobile End Points with Multi-Homing: Our future Internet design assumes the existence of billions of mobile end-points each of which traverses as many as 100's of wireless access networks in a day. In addition, mobile end-points will typically be multi-homed devices with access to multiple wireless networks such as WiFi and cellular. Name (GUID) based message delivery makes it possible to offer seamless mobility and multi-homing services without the problems associated with today's IP. Note that multi-homing generally involves reaching a mobile device via two or more distinct network domains, and thus has implications for both intra- and inter-domain routing.

Network Mobility, Ad-Hoc and Disconnected Modes: The MobilityFirst protocol stack is also being designed to support network mobility, i.e. migration of entire networks and not just end-points. In addition, the network should support ad hoc infrastructure-less communication between mobile devices in proximity (for example vehicle-to-vehicle) without the need for a connection to the Internet. Thus the name resolution and routing protocols should be able to deal with periods of disconnection in a robust manner.

Heterogeneous Wireless Access: Heterogeneity is a basic property of the wireless/mobile environment for which the MobilityFirst protocol is designed. Differences in radio access technology (e.g. WiFi vs cellular vs. Zigbee) along with variations in signal quality can lead to orders-of-magnitude differences in available bit-rate even during a single session, motivating techniques such as in-network storage and edge-aware routing.

Routers with Storage and Late-Binding: Routers in the MobilityFirst network are designed to deliver large files on a hop-by-hop basis with the option for temporary storage (to overcome short-term link quality fluctuations and disconnections) and late binding (to deal with changing points of attachment due to mobility).

Hybrid Name/Address Based Routing: The MobilityFirst protocol uses both GUIDs (i.e. names) and NAs to deliver data across the network. Source and destination GUIDs define the authoritative packet header while an optional set of NAs can be added to specify a "fast path" wherever possible. Routers normally forward packets based on NAs but refer back to the GUID as needed to deal with topology changes due to mobility or disconnection. End-to-end routing between network objects thus requires the combination of the GNRS and an NA-based global routing protocol.

Storage-Aware Intra-domain and Edge-Aware Inter-Domain Routing: MobilityFirst intra domain routing protocols are designed to support in-network storage when necessary to overcome link quality fluctuations and disconnection. In addition, the global inter-domain routing protocol needs to have some degree of edge awareness because of the need to deliver data efficiently to/from multiple edge networks with very different properties, e.g. slow cellular vs. fast wired.

Hop-by-Hop Transport: The MobilityFirst protocol uses hop-by-hop (or segment-by-segment) transfer of large files between routers, with the entire file received and stored at each node before sending on to the next hop. This approach makes it possible to implement storage and late binding functions at routers, while also providing important performance benefits (over conventional flows with end-to-end transport) in complex wireless/mobile environments.

Optional in-network computing services: Storage of messages/files at routers makes it possible to introduce enhanced services via an optional computing layer at the routers. This computing layer can be invoked for certain GUIDs and SIDs, enabling functions such as content caching, location-

aware routing, or context-aware message delivery. This feature also offers a path for evolution of protocol functionality over time.

The architecture outlined above can be realized with the following basic protocol building blocks, summarized in Fig. 1.2.2. In addition to name certification services (NCS) shown at the top level, the MF protocol design involves two distinct layers, the meta-level network services layer responsible for realization of abstract name-based services and a core transport services layer responsible for routing and forwarding. Meta-level network services are implemented using three basic modules – the global name resolution service (GNRS) which is a distributed service across the whole network, the name-based service software module running on all end-points and routers, and optional compute layer plug-ins at participating routers. Core transport services are also implemented using three distinct modules – the hybrid GUID/NA based global routing protocol, storage-aware/DTN intra-domain routing and hop-by-hop transport.

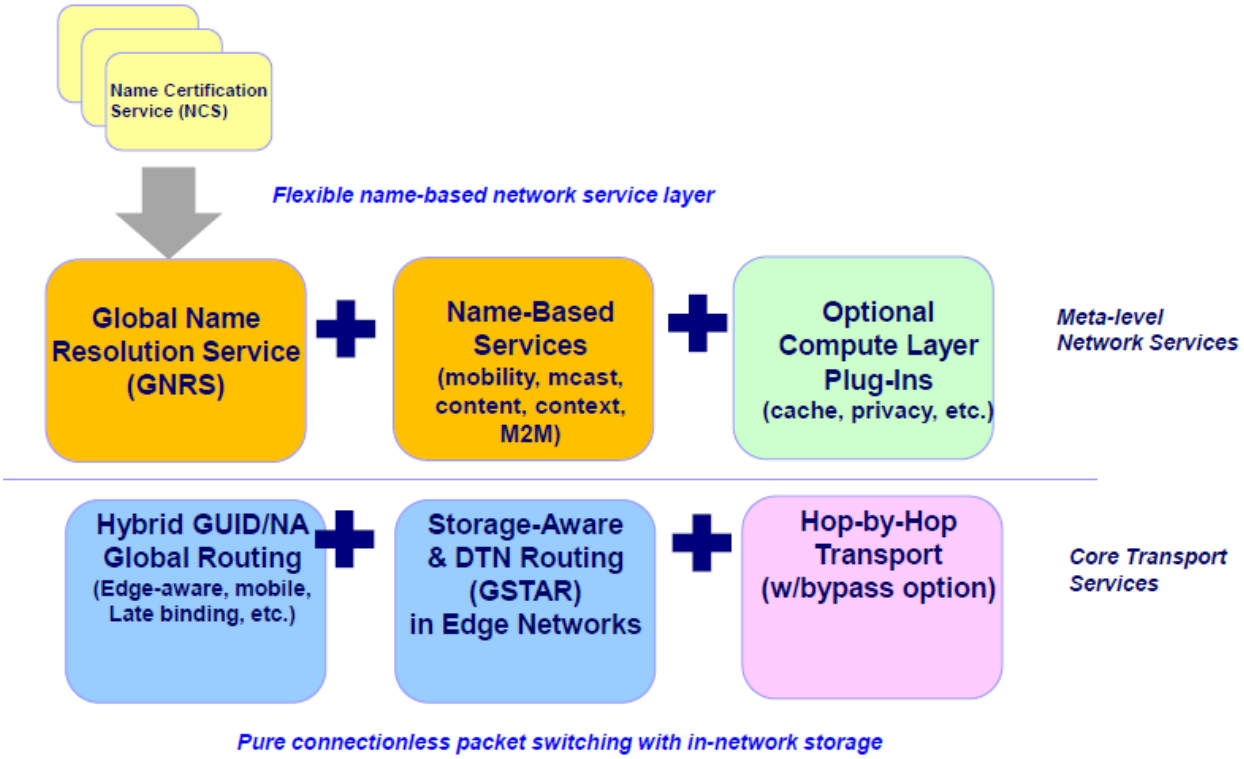

Fig. 1.2.2 Basic Protocol Building Blocks in MobilityFirst

1.3 MobilityFirst Protocol Overview:

Based on the considerations outlined in Sections 1.1 and 1.2, we have developed an initial specification (v1.0) for the high-level protocol architecture of the network. Although the design is still evolving and detail is being added at each level, there is a general understanding of the packet structure, major header elements, primary protocol mechanisms (such as name resolution and routing), service types and major security mechanisms. The reference MobilityFirst protocol stack as currently defined is shown in Fig 1.3.1 below. As mentioned earlier, the protocol stack is centered around the GUID service layer which provides abstractions for name-based services. The GUID service layer in the data plane is supported by the GNRS in the control plane, while the routing functions in the data plane are supported by applicable intra- and inter-domain control protocols for routing. The figure also shows how the optional compute layer can be

added above the GUID service layer. Also shown are multiple end-to-end transport protocol options which provide socket API's for services such as reliable message delivery, delayed delivery, reliable content retrieval and so on. Applications are supported in the control plane by the name certification services which provide GUIDs corresponding to specified human-readable names.

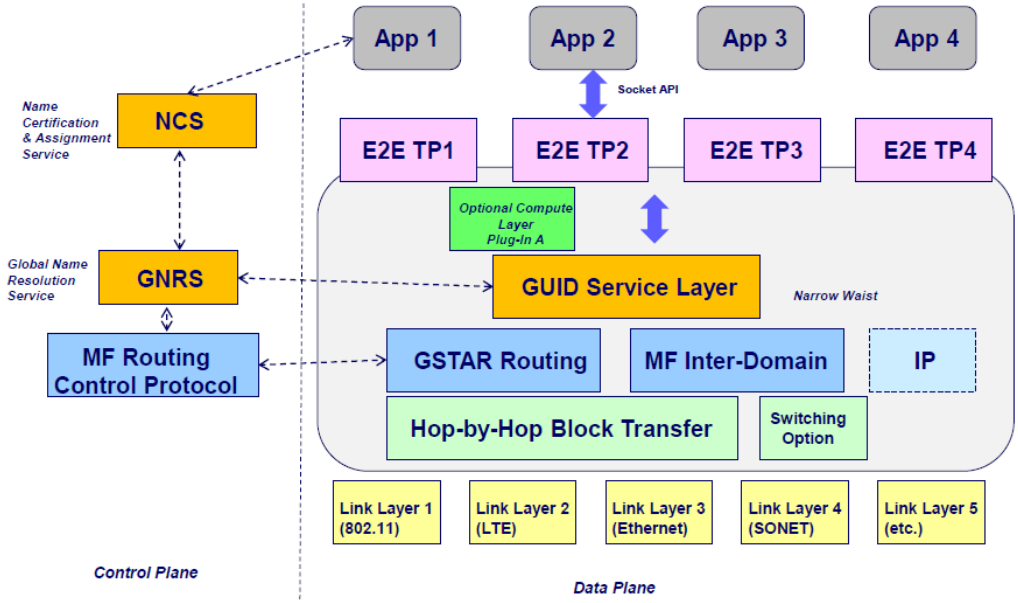

Fig. 1.3.1 MobilityFirst Protocol Stack

In order to understand how the protocol works in further detail, first consider how a name is converted into a GUID by the name certification service (NCS). As shown in Fig. 1.3.2 below, a number of specialized NCS providers may cater to name assignment and trust management in different domains such as devices/hosts, M2M, content or context. There may also be a network naming and trust service from which constituent networks obtain their GUIDs and build trust relationships with other peer networks.

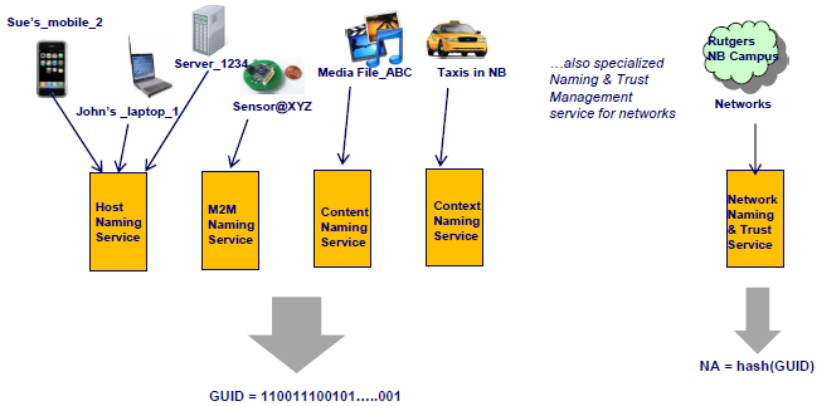

Fig. 1.3.2 Multiple NCS providers in MobilityFirst

Next, consider how a message is sent from between two end points. As shown in Fig. 1.3.3, a host wishing to send a message to all of "John Smith22's devices" will obtain the corresponding GUID from either the NCS or the end-user and then invoke the service API using a command such as Send (GUID, options, data), where options can include service features such as anycast, multicast, timed delivery and so on. The host driver then prepares a MobilityFirst packet with GUID and SID in the header as shown. The GUID is then resolved through a GNRS lookup (either at the sending host or at the edge router) to a set of network

addresses (NAs) corresponding to the current points of attachment of this abstract object, in this case NA99 and NA32. The packet header actually sent out by the host (or edge router) then consists of a destination GUID, SID and list of NAs. Routers in the network will use the NAs (which can be thought of as a "fast path") to make forwarding decisions, with multicast and copy functions added in where appropriate to reach both NA99 and 32. If a delivery failure occurs due to disconnection or mobility, the packet is stored inside the network and the GNRS is periodically queried for a rebinding of the GUID with NAs. Depending on policy, the packet is either delivered to the new destination within a certain amount of time, or discarded due to time-out.

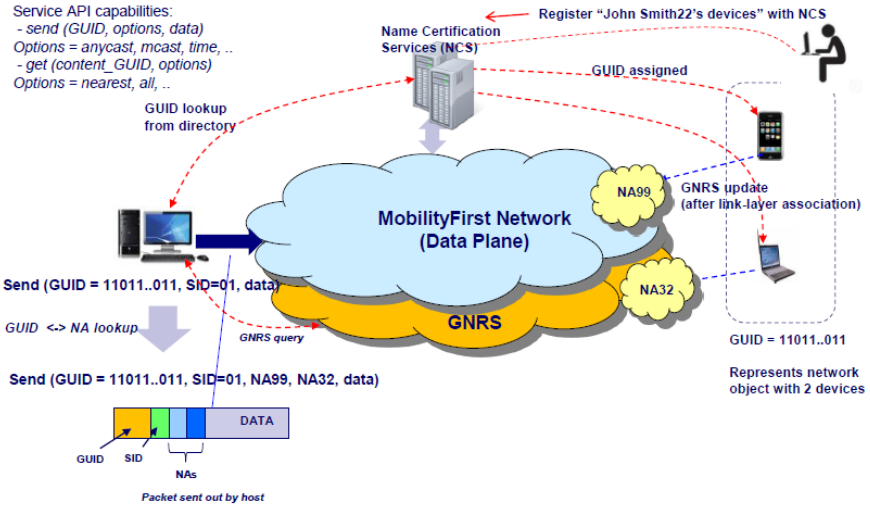

Fig. 1.3.3 Example showing message delivery between end-points

Consider next the actions at a MobilityFirst router, as shown in Fig. 1.3.4. Each router in the network has access to two kinds of routing tables – one which maps the GUID to NAs (implemented as a virtual DHT table as discussed later), and other which maps the destination NA to a next hop or port number for forwarding. From the figure, it is observed that packets entering the network always have the destination (and source) GUID attached to the protocol data unit (PDU). There is also a service identifier (SID) in the packet header which indicates the type of service required by the PDU including options such as unicast, multicast, anycast, context delivery, content query, etc. As explained earlier, there may also be an optional list of NAs appended to the GUID and SID in the packet header.

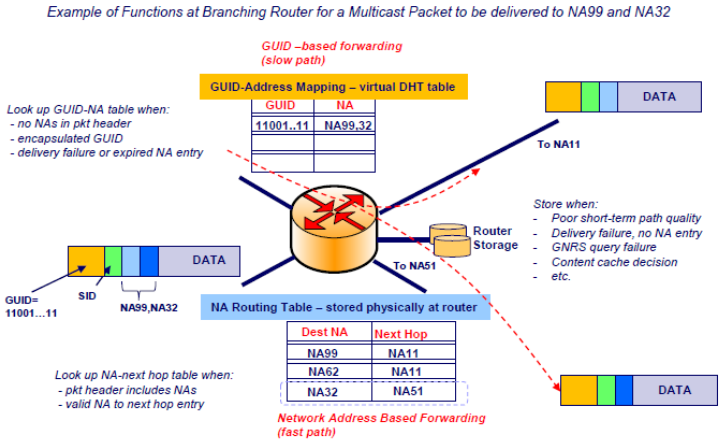

Fig. 1.3.4 Hybrid GUID/NA packet headers in MobilityFirst

When a list of resolved NAs corresponding to the GUID is available, the router only needs to look up the NA routing table as in a conventional IP router – this is referred to as "fast path" forwarding. Any router along the path has the option of resolving the GUID again by querying the GNRS – this is the so-called "slow path" which allows for rebinding to a new set of NA's that may have resulted from mobility or temporary disconnection. The GUID routing option makes it possible to implement "late binding" algorithms where the decision on which NAs for routing can be identified or modified while the PDU is in transit. Of course, there is a higher cost for GUID lookups at routers due to latency and GNRS protocol overhead, but this is incurred only once for large protocol data units which may be ~10M-1GB in size. Note that both GUIDs and NAs in the architecture are flat, i.e. no hierarchical structure is assumed.

Another key feature of the architecture is the existence of in-network storage at routers. Each router along the path has the option of temporarily storing PDUs at routers instead of forwarding towards the destination when poor link quality or disconnection is detected by the path quality metric. PDUs which are placed in temporary storage are scheduled for periodic checks of path quality and forwarded when appropriate. If the destination is completely disconnected, the router will also periodically initiate GNRS queries to determine the new point of attachment if any. Also, a reliable hop-by-hop transport protocol is used to deliver packets between routers in contrast to the end-to-end approach used in TCP/IP.

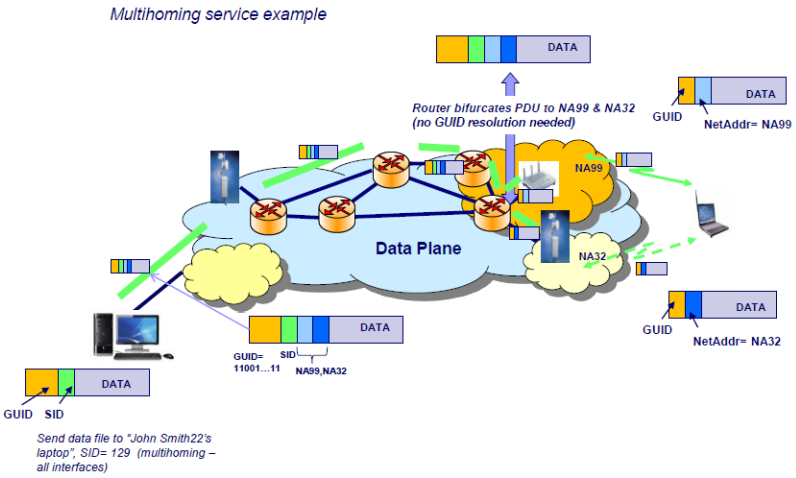

Fig. 1.3.5 Supporting Dual-Homing in MobilityFirst Routing

Another key feature of the proposed MobilityFirst protocol stack is the service flexibility, with particular emphasis on multicasting and anycasting modes as integral capabilities of the routing protocol. These service features have been provided in response to the needs of mobility applications which often care more about the context (e.g. device location or function) than its network address. The GUID mechanism outlined above allows for context and content addressability with multicasting or anycasting to the set of network addresses associated with a GUID (such as taxis in New Brunswick or Alice's_laptop). A particularly interesting use case that is difficult to handle with conventional IP is that of "dual-homing" where a user's laptop may have two wireless interfaces (such as WiFi and 3G) on separate access networks, and the service objective is to deliver to at least one of these interfaces based on a suitable cost metric. An example of how the protocol works for such a dual-homing scenario is given in Fig. 1.3.5 below. In this example, the GUID for "Alice's_laptop" is resolved to two distinct network addresses corresponding to the 3G and WiFi networks that it is currently connected to. The PDU carries both these network addresses and the network routing protocol implements a "longest common path" type of algorithm before bifurcating the same message to both destinations.

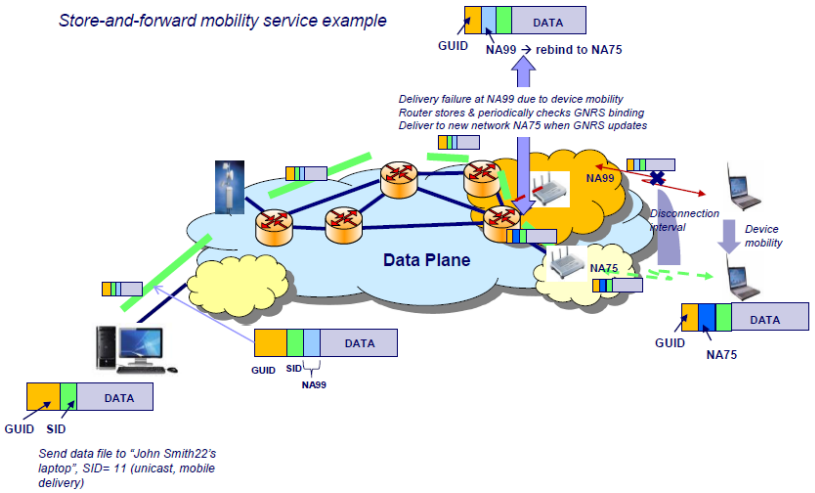

Fig. 1.3.6 Example of Mobile Service with Disconnection

Another service example is given in Fig. 1.3.6 above, which shows the case with mobility and complete disconnection of the end point. In this example, the initial binding of the GUID is to NA99, but a delivery failure occurs at the edge network shown due to destination device mobility. The protocol data unit is then stored at the edge router and the GNRS is periodically queried for the new NA, which later turns out to be NA75. The packet is then updated to include the new NA and then forwarded towards the revised destination.

As final example, Fig. 1.3.7 shows an enhanced content caching service using the optional compute layer. In this case, a mobile device wishing to retrieve content simply makes a query such as Get (Content_GUID, SID=cache service). The GNRS resolves this to a set of NAs (NA 99, 31, 22 and 43) where the content is currently cached. The access router then looks up the closest cache location from the routing table and forwards the query to the applicable network (NA99). The caching router which offers the enhanced service processes this query at the computing layer and then sends back the requested content to the mobile device as shown. Other services such as location-based message forwarding can also be implemented in a similar manner.

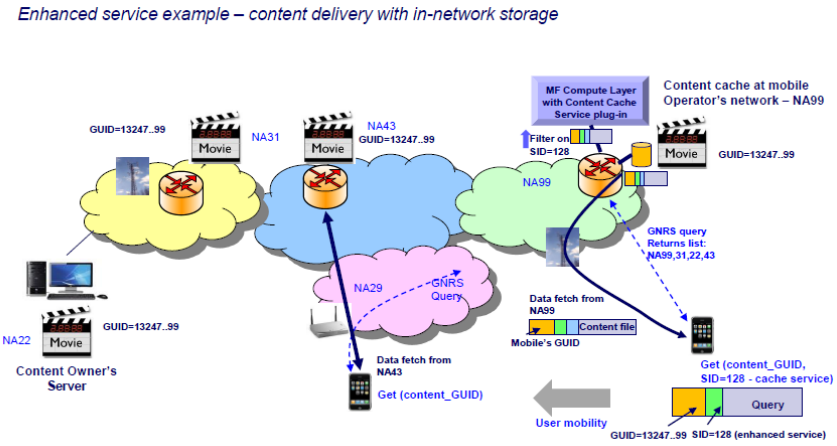

Fig. 1.3.7 Example of an enhanced service (content caching) using the compute layer

2. MobilityFirst Protocol Spec Summary

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.

2.1 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.

2.1.1 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.

2.1.2 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:

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

2.1.3 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.

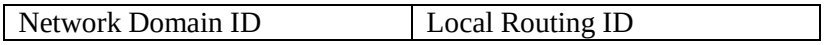

Figure 2.1.3: General structure of a topological address

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 public key => 160 bits
Unique Local IPv6 Unicast Address	RFC 4193 => 128 bits

Table 2.1.3: Example of local routing identifiers that could be used for routing within a domain

2.1.4 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.

2.2 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 'over-buffering' within the network core where link/path qualities are normally stable.

2.3 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.

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.4 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.

2.5 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

2.6 Service-Oriented Routing Header

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

0		31
Service ID	*Header Length*	*Next Header*

Protocol ID	*Hop Count*	*Payload Offset*
Payload Length		
Destination GUID (short) - 5w		
Destination Network Address – 5w		
Source GUID (short) – 5w		
Source Network Address – 5w		
Service Header Extension(s) ...		

Where the fields are:

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 a cryptographic hash of destination GUID, e.g., 160 bits
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 a cryptographic hash of the source GUID, e.g., 160 bits
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

2.6.1 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 a service header extension is shown below.

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

Figure 2.6.1: 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).

SID Encoding: 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 follows the pattern 0x8xxx.

3. Protocol Layering in MobilityFirst Stack

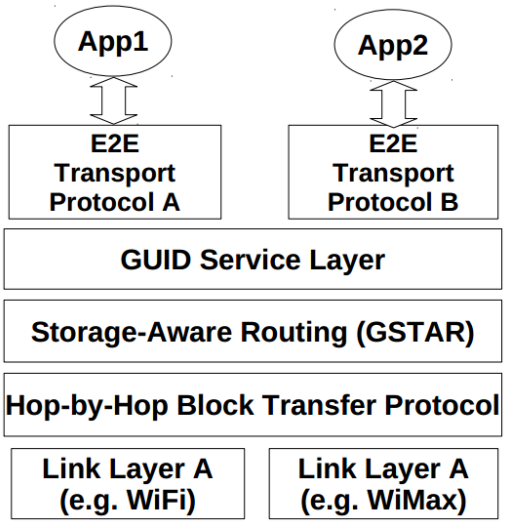

Figure 3: Layering in MobilityFirst Protocol Stack (host protocol stack shown)

Figure 3 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 end-applications. 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.

3.1 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.

3.2 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 stacks 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.

3.3 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.

3.4 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, svc-opts)*	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.

REFERENCES

[1] D. Raychaudhuri, MobilityFirst: A Robust and Trustworthy Architecture for the Future Mobile Internet, IEEE PIMRC 2011 Keynote Talk, http://mobilityfirst.winlab.rutgers.edu/documents/PIMRC_keynote_MF_911.pdf

[2] I. Seskar, K. Nagaraja, S. Nelson and D. Raychaudhuri, "MobilityFirst Future Internet Architecture Project", ACM AINTEC 2011, http://mobilityfirst.winlab.rutgers.edu/documents/ACM_AINTEC2011_Seskar_paper.pdf

[3] MobilityFirst project website: http://mobilityfirst.winlab.rutgers.edu

[4] Hop: A Fast Wireless Transport Protocol http://hop.cs.umass.edu/

[5] GSTAR: Generalized Storage-Aware Routing for MobilityFirst in the Future Mobile Internet. S. Nelson, G. Bhanage, D. Raychaudhuri. In Proceedings of ACM MobiArch, 2011 http://mobilityfirst.winlab.rutgers.edu/documents/GSTAR.pdf

[6] DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet. T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, D. Raychaudhuri. In Proceedings of IEEE ICDCS, June 2012 http://mobilityfirst.winlab.rutgers.edu/documents/GNRS_ICDCS2012_Vu.pdf