Enabling Advanced Network Services in the Future Internet Using Named Object Identifiers and Global Name wResolution

Shreyasee Mukherjee, Parishad Karimi, Dipankar Raychaudhuri
WINLAB, Rutgers University
North Brunswick, New Jersey, USA.
Email: {*shreya, parishad, ray*}*@winlab.rutgers.edu*

Francesco Bronzino
Inria
Paris, France
Email: *francesco.bronzino@inria.fr*

***Abstract*—This paper presents the concept of named object identifiers as the architectural foundation for realizing advanced services, mobility and security in the future Internet. The proposed named object approach uses unique identifiers for service definition and end-to-end message delivery, and can be added as a new layer on top of the IP architecture in a backward-compatible manner. The proposed ID-based service layer requires control plane support in the form of a global name resolution service (GNRS) for dynamic binding of names to network addresses. The requirements for a generalized and flexible name resolution service are discussed considering both functional and performance aspects. Several proposed realizations of the name resolution service are described, including DMap and Auspice used in the MobilityFirst future Internet architecture. In conclusion, examples are given for some new services supported by the proposed identifier-based architecture and specific identifier-based protocol designs, such as mobility, multi-homing, multicast and context-based services.**

Keywords–Future Internet Architecture (FIA); Named services; Name resolution.

I. Introduction

The current Internet architecture, which was designed with fixed hosts in mind, uses IP address to identify both the users, as well as their location. This overloading of the namespace, also called location-identity conflation [1], makes deploying basic services such as mobility or multi-network access, challenging. End-to-end protocols such as, TCP are tied to the IP address of an interface which changes as an end-point moves, causing transport and application sessions to break. Group based communication to Internet-of-Things or anycast based cloud service access are important use cases which currently require overlay networking solutions above the IP layer and could benefit from improved network layer services. We believe that it is timely to consider evolving the IP architecture to support location-independent identifier-based communications between "named objects" in order to realize significant service flexibility and security benefits [1]–[3]. Separation of names or identities (IDs) from network address/locator has been proposed in multiple architectures to facilitate location-independent communication [2]–[6]. Note that while some architectures use *names* to denote network-attached objects, others use identities (IDs), both of which can be loosely defined as a string defining a communicating end-point, and, for the purpose of this paper, we use them interchangeably. Assigning long-lasting unique IDs to different

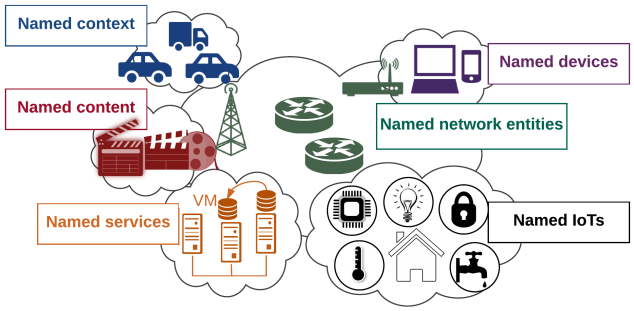

Figure 1. Named-object abstraction: names can be assigned to any network connected entity

network entities ranging from end-points to contents will allow for native support of services on top of any underlying routing mechanism, including IP. As shown in Figure 1, names are quite versatile, and can be used to identify any network-attached object, from traditional end-user devices to IoT groups (for example, sensors in a smart home), specific content in information-centric networks, named network entities (such as access points and routers within a domain) and context, (for example, vehicles on the New Jersey Turnpike between exits 9 and 10).

The naming layer will be placed between the network and transport layer, alleviating the need for those layers to inefficiently support different services like mobility [7] and multihoming [8]. The packet header will include both the ID and the routable address(es), allowing for address-based data traversal within the routers. This will provide a backward-compatible solution, which can be incrementally-deployable on top of the current IP-based Internet. The control plane that enables routing of ID-based packets consists of a globally reachable name resolution service, which will provide name to address mapping to end-points, first-hop routers or any core router depending on the service.

In this paper, we first discuss the necessity of such a global name resolution service (Section II). Then we identify the design requirements for realization of a generalized distributed name resolution service for ID-based networks (Section III). Next we describe two in-network and one overlay name resolution services developed for the MobilityFirst architecture [4] and highlight the set of requirements they fulfill (Section IV). Finally, we explain how such a generic name resolution service

TABLE I. COMPARISON OF EXISTING AND PROPOSED MAPPING RESOLUTION SYSTEMS

System	Mapping resolution	Implementation
Domain Name System (DNS)	URL → IP addr	Dedicated servers in application layer
Network address translation (NAT)	IP addr → IP addr	In-network at NAT-capable routers
MobilityFirst GNRS	GUID name → network addr	In-network routers
LISP ALT	IP addr name → IP addr location	Dedicated overlay servers
Serval	Service ID → sock/addr	In-network at Serval-capable routers

can enable basic mobility or session continuity as well as advanced services, such as multi-network access, large scale multicast and context aware services (Section V).

II. NAME RESOLUTION SERVICES

Clearly, the use of identifiers implies the need for an efficient resolution system that can provide fast and efficient identity to location translation for all such named objects. In the current Internet, two similar resolution systems exist. A distributed globally available Domain Name System (DNS) translates identities (URLs) to obtain network locations (IP addresses) [9] and NAT capable routers locally translate private IP addresses to public IP addresses. However, a key drawback of existing systems is that they were designed based on the notion that most entries will either be static or change at a relatively slow time-scale. Even though DNS has historically evolved significantly from the time it was based on text files to sophisticated hierarchically distributed resolvers, it still lacks the support for the requirements of next generation networks, i.e. a distributed mapping infrastructure that can scale to orders of magnitude higher update rates with orders of magnitude of lower user-perceived latency. Alternatively new designs for distributed resolution services have been proposed for a variety of device, content and service oriented communication, most notably the global name resolution service (GNRS) in MobilityFirst [10]–[12], the distributed overlay ALT servers in LISP [13] as well as distributed translation of service identifiers to interfaces in Serval [14].

Table I summarizes the basic design choices and namespace translation of the aforementioned resolution systems. As shown, each of these resolution systems have their own implementation logic or APIs, and reside in different layers of the internet architecture. While they each focus on slightly different objectives, we believe that it is useful to look into the fundamental requirements for identity-based networks and propose a generic name resolution service with a unified control plane that allows interoperability in the data plane of existing and proposed ID-based architectures.

III. FUNCTIONAL REQUIREMENTS

In this section we describe the key requirements from a resolution system to enable ID oriented future services.

A. Low Update and Response Latency

User perceived latency plays a crucial role in the quality of experience of any digital commercial service subscribers. As reported by VMware, typical network latency of about 100 milliseconds is considered acceptable for the usage of

TABLE II. BASIC RESOLUTION SYSTEM STRUCTURE AND API SEMANTICS

insert(ID, <value set>, opts)
update(ID, <value set>, opts)
query(ID)
delete(ID)

(a) API semantics

Key	Value	Metadata
ID	<values>	Opts
...	...	...

(b) Database structure

their office productivity services [15]. In 2006, Amazon found that every 100 millisecond of added latency reduces sales by 1%. Considering that Amazon's total revenue in 2006 was 19B+, this would have amounted to a loss of 190M per 100 milliseconds of added latency [16]. Future 5G applications such as vehicle-to-vehicle safety messaging or real-time mobile control may require less than one millisecond network latency to be deployable in practice [17], which mandates future name resolution systems to be able to return up-to-date responses within milliseconds.

Note that the network latency includes both the time to update a mapping and the time to return a correct response to a querying entity for the mapping. Therefore, the resolution system should be physically distributed with the distribution optimized to find the sweet spot of minimizing lookup latency and update latency. That is, an ideal resolution system should have potentially all mappings in close network-proximity to both the entities making inserts and queries. While this could be practically hard to achieve in some cases, specially if the entities are topologically far away, it brings forth interesting challenges on how to optimize distribution based on the identity of the service itself; for example, vehicle to vehicle safety communication ($\sim$1 millisecond) vs. locally popular content caching (10s of milliseconds) vs. globally available personal cloud storage($\sim$100 milliseconds).

B. Storage and Load Scalability

There are approximately 4.9 billion global mobile data users and according to a recent study, over 20% of these users currently change network addresses over 10 times a day [18]. Cisco has predicted that by 2021, the number of mobile users will go up to 5.5 billion, whereas the total number of mobile connected devices could be as high as 12 billion [19]. Even if the mobile data users follow certain predictable patterns of mobility, this growth in the number of mobile objects will generate in the order of 10s of billions of daily updates. This in turn would require further resources and create additional workload for the common name resolution infrastructure.

To address these scalability concerns, DNS currently relies heavily on caching of mapping entries through its hierarchy (local name servers, authoritative name servers, top level domains) to help reduce both system load and client-perceived latency. However, handling mobility at this scale requires up-to-date responses, which makes caching ineffective (near-zero TTLs). As a result, the load and client-perceived latency increase with the mobility rate. Therefore the proposed resolution system should be able to scale to orders of magnitude higher storage and load scalability than existing systems.

C. Extensibility and Flexibility

In order to simplify the deployment of a range of ID based services, the resolution system should be flexible enough to

TABLE III. SUMMARY OF REQUIREMENTS FROM A NAME RESOLUTION SYSTEM

Requirements	Goals
Latency	Low(<1ms) to medium(100ms) based on service req.
Scalabilty	Very high workload(>100B updates per day)
	Moderately high storage based on distribution
Implementation	In-network and distributed
Semantics	Flexible and scalable information schema <key, value> pair + supplementary information (optional)
	Standardized APIs
Security	Attack resilient, access control, flexible policies
	Optional confidential info, private instantiations

store multiple kinds of mapping (key→values). For example, it could capture relationships like grouping between names by providing name-to-name mapping and recursive resolution of names. This would not only enable name based multicast communication but also allow a richer information schema to be mapped onto names and then stored in the same resolution system, as explained further in Section V.

The syntax and semantics should also be flexible enough to support a range of existing and future name based architectures [4]–[6], which could all utilize the resolution system as a common control plane, accessible through well-defined standardized APIs. Therefore, the structure of the database itself should not be bound to the structure of the names. For example, HIP [5] and MobilityFirst [4] use flat names, whereas LISP [6] which utilizes IP addresses, has hierarchical names. The database should also allow extensible fields or some form of optional information to be stored per mapping as meta-data, which could be essential for certain kinds of service deployments.

Table IIIa highlights the basic API that includes semantics for inserting a new entry, updating an existing entry, as well as querying and explicitly deleting of an entry. Although time-to-live (TTL) based delete could be performed, similar to DNS, we believe that TTL based designs make it difficult to handle fast mobility as well as temporary disconnections prevalent in wireless access scenarios. Table IIIb further shows the database structure with fields for inserting the ID as the key, a set of values and optional meta-data.

D. Security and Reliability

The resolution service serves as a database, mapping IDs to the location of network-attached objects (which may be correlated to physical locations). Its central role in providing such name resolution entails security and privacy as important design considerations. Local or private instantiations and confidential mappings should also be provisioned for. However, there should not be a single root of trust and strict hierarchical distributions, since, database placement should be optimized based on the service requirements, which in most cases is not closely tied to autonomous systems and network hierarchy. It is also important to allow access control and flexible policy support to prevent malicious usage of the infrastructure [20].

Table III summarizes the broad set of functional requirements for a generic name resolution system for ID-oriented communication in the future internet.

IV. GLOBAL NAME RESOLUTION INFRASTRUCTURE FOR MOBILITYFIRST

MobilityFirst relies heavily on the name resolution service for advanced network-layer functionalities. This reliance necessitates high performance from the resolution service, which depends on resolving identifiers to dynamic attributes in a fast, consistent, and cost-effective manner at Internet scale. Keeping the above requirements in mind, the project has looked into alternative designs of the name resolution system [10]–[12], including both in-network and overlay designs. The MobilityFirst namespace is flat, with globally unique identifiers (GUIDs) that can be assigned to any network-attached object, from individual devices to groups, network routers and services, as shown earlier in Figure 1. These GUIDs are 160 bits and derived from public keys, hence they are self-certifiable and cryptographically secure. Routing is based on network addresses with the name resolution system storing up-to-date mappings between the GUID and its corresponding network addresses. The data packets are also self-sustained and carry both the GUID and the routing address in the header. This ensures in-transit packets to be rebinded by any router along the path, to a new network address through a mapping re-query, as and when required (for example, during mobility). All of the three designs, that is, DMap [10], Auspice [11] and GMap [12], support the basic APIs for insert, update and querying an entry based on GUIDs and have similar database structures with globally distributed implementation and no centralized root of trust.

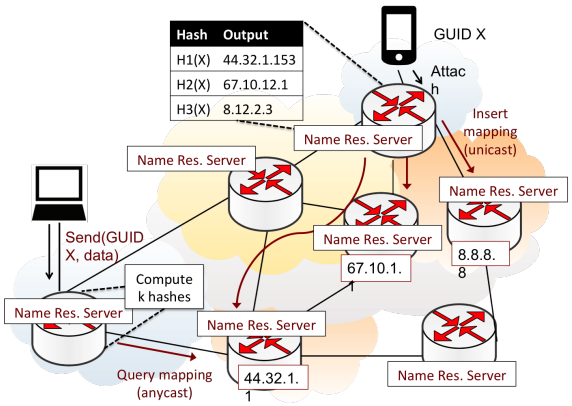

Figure 2. DMap based insertion and lookup of GUID X to locator mapping

DMap: The direct mapping (DMap) design was the first proposed implementation, which is an in-network approach, wherein every autonomous system (AS) in the global network participates in a hashmap based name resolution service in order to share the workload of hosting GUID to network address mapping. Figure 2 provides an overview of how DMap distributes each GUID mapping across K replica servers in the internet. Assuming the underlying routing to be stable and all networks to be reachable, DMap hashes every GUID to K network addresses (which are IP addresses in this example) and then stores the mapping at those K addresses. Every time the mapping changes, K update messages are sent to each of the servers at these locations. Correspondingly, every query for the current mapping of the GUID is anycasted to the nearest of the K locations, as shown.

DMap is the simplest of the three designs and it manages workload balance across all the ASes efficiently. Since uniform hash functions decide *where* a mapping is stored, basic DMap implementation is not suitable for geographically optimized

mapping placement based on service requirements. However the focus of this work was on providing a globally available mapping system with high availability, and moderate latencies, making it ideal to handle basic mobility and services with medium latency requirements. Detailed internet scale simulation of DMap shows that with 5 replicas per GUID, the 95^{th} percentile latency is around 86 milliseconds [10], which is reasonable for most user-mobility centric applications.

Auspice: The main design goal of Auspice, which uses an overlay approach above the network layer, is to provide an automated infrastructure for the placement of geo-distributed name resolvers in order to reduce update and query latencies to tens of milliseconds [11]. The two main components of Auspice are the *replica controllers*, which determine the number and geo-location of name resolvers, and the *name resolvers (active replicas)*, which are responsible for maintaining the identifiers attributes and replying to user-request read or write operations. Each name is associated to a fixed K number of replica-controllers and a variable number of active replicas of the corresponding resolver.

Auspice performs per GUID optimized replica placement with the replica controllers aggregating update and query frequency to compute popularity and hence number of replicas of the mapping required and where to place them. Although the mapping infrastructure is distributed, Auspice is an overlay implementation and does not require in-network routers to participate in sharing the workload. The database design is also more generic with key as the GUID and the mapping being expressed as a <type, length, value> field. Therefore, Auspice can store arbitrary strings as a value mapped onto a GUID. Auspice also takes into account the resource and latency trade off in its optimization for replica management. So if more resources are available, it can decide to disseminate more replicas per GUID and hence reduce overall lookup latency. Detailed comparative evaluation shows that Auspice with 5 replicas is comparable to commercially deployed UltraDNS (16 replicas) and with 15 replicas has 60% lower latency than UltraDNS. Auspice with 5 replicas is also 1.0 to 24.7 secs lower than three top-tier managed DNS service providers for propagating updates globally.

GMap: Finally GMap [12] is an updated version of DMap, in which the GUID→address mapping is distributed hierarchically considering geo-location and local popularity. For each GUID, similar consistent hash functions are used to assign resolution servers. However for each mapping, the servers are categorized into local, regional and global sets, based on geo-locality. Each mapping now gets replicated into K1 local servers, K2 regional servers and K3 global servers. Therefore, unlike Auspice, GMap does not require per-GUID replica optimization, but still achieves better latency than DMap, at the cost of higher storage workload, due to increased number of replicas per GUID. In addition, GMap allows temporary in-network caching of the mapping along the route between a resolution server and a querying entity, to ensure future mapping requests for the same GUID to be resolved faster. Internet-scale simulations show GMap to achieve similar latency goals of tens of milliseconds as Auspice but with lower complexity and computation overhead. Table IV summarizes the key features of each of the designs.

TABLE IV. SUMMARY OF MOBILITYFIRST NAME RESOLUTION SERVICE IMPLEMENTATIONS

	Auspice	**GMap**	**DMap**
Implementation	Overlay	In-network	In-network
Algorithm type	Demand-aware replicated state machine	Distributed hash table	Distributed hash table
Record content	GUID to arbitrary number of values	GUID to arbitrary values (recursively other GUIDs or Network Addresses)	GUID to up to 5 NAs, each with an expiration time and prioritization weight
Name server placement	Geo-located based on requests	Geo-located based on physical location of the GUID	Not Geo-located, except 1 local mapping
Number of replicas per GUID	Based on recent demand and update frequency	Fixed number; each GUID has K1 local, K2 regional, K3 global replicas	Fixed number: each GUID has K global, 1 local replicas
Caching	No caching; load balancing by adjusting number of name servers	Caches response along the path from querying entity and name server	Future work

V. NAME BASED SERVICES

In this section, we explain how a range of services, namely mobility, multihoming, multicast and context-aware services can be supported efficiently, using the concept of "named object" identity within the network.

A. Host and Network Mobility

Due to the rapid proliferation of mobile users, ranging from cellphones to drones, mobility should be treated as a first-class service. One of the most significant use cases for future networks is supporting mobile data services on a fast scale, like authentication and dynamic mobility, involving both micro-level handoff and macro-level roaming. The current approaches for mobility support such as mobile IP [7] suffer from routing inefficiency (in terms of latency, overhead and congestion at service gateways), due to triangular routing through an anchor point. Mobility can be handled better within a name-based architecture which is facilitated by a name resolution service meeting the functional requirements discussed in Section III.

- Baseline: This is the simplest case where on delivery failure, the packet is re-sent from the original sender's location.
- Re-bind (also called "late binding"): When a delivery fails, the name resolution service is queried for an updated location and the packet is forwarded from the current network address, instead of the original sender's location.
- Last Known: This is an extension to the 're-bind' case. The main difference is observed when the user is disconnected and the current location is not available in the name resolution service. In such a case while the 're-bind' scheme holds the packet, waiting for a location update, the 'last known' scheme forwards the packet to the last known location in the GNRS. We expect the user to be closer to his previously known location when compared to the location of the sender.
- Ideal: This scheme represents best possible scenario. Using prediction schemes with the information available in the name resolution service it is possible to get closer to the performance of the ideal case.

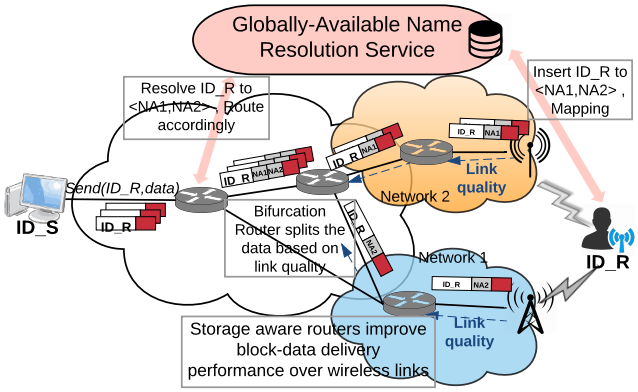

Figure 3. Overview of multihoming supported by a globally-available name resolution service

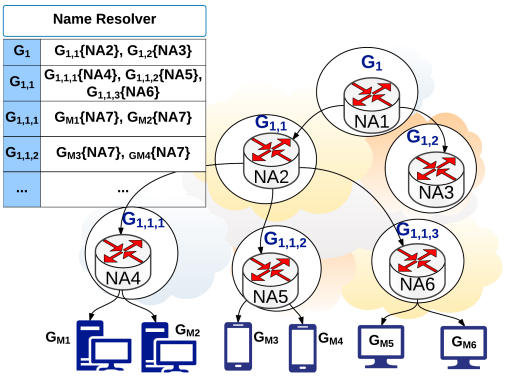

Figure 4. Name based multicast with recursive name lookups using the name resolution service

B. Multi-homed traffic engineering

Multihoming can natively be supported by a name-based architecture. A multi-homed device is simultaneously attached to more than one service network. The separation of names and addresses allows for a device or group name to be bound to a dynamic set of multiple network addresses, denoting the points of attachment of the device to the network. In-network multipath routing is enabled using a global name resolution service as follows: the first hop router that receives a packet destined to another endpoint's name queries the name resolution service for the locations of that name. After receiving the reply from the service, the first hop router appends all the network addresses associated with the receiver's ID to the packet. As data packets traverse through the core network, the routers forward the packets until a branching point is reached. This branching point is the router that faces different next hops for the various network addresses and can dynamically change in case of mobility. This bifurcation router can be programmed to schedule the data on each path according to link quality metrics or policies inserted by the multi-homed endpoints. The link quality metric can utilize cross-layer information from link layer protocols or a feedback mechanism from edge wireless networks. This service can be enabled on top of IP as well, with some limitations on performance, considering the lack of path quality information in current mainstream network and link layer protocols. An overview of how a distributed name resolution service which serves the functional goals discussed earlier can facilitate multihoming is shown in Figure 3.

These approaches have been shown to boost the performance of multihoming compared with current end-to-end approaches such as MPTCP [21], [22]. The extensible fields as metadata for each identifier in the name resolution service can further allow for storing fine-grained expressive policy information about the multi-path connection, e.g., prefer WiFi to LTE; or use WiFi for delay-tolerant downloads and LTE for delay-sensitive traffic, etc.

C. Large-scale multicast

Internet applications like video streaming, online gaming and social networks, e.g. Twitter, often require dissemination of the same piece of information to multiple consumers at the same time. While multicast routing protocols have long been available, most of these applications rely on unicast based solutions without support from the network. Using appropriate multicast routing solutions would help, however, existing network-layer multicast solutions (e.g., PIM-SM [23], MOSPF [24]) have not been widely adopted, mainly because issues with scalability and coordination across multiple domains. In view of the shortcomings of existing schemes, a network-layer multicast solution, that utilizes the named-object abstraction was designed as part of the MobilityFirst project. In this design, names are used to identify a multicast group, as well as the multicast tree itself and can be stored and managed in a distributed fashion through the name resolution infrastructure. As shown in Figure 4, a multicast service-manager computes the multicast tree and assigns GUIDs to each of the branching routers of the tree. This tree is then stored in the resolution service in a recursive manner, wherein each branching router maps onto the next set of downstream branching points along with their network addresses, with the leaves of the tree being the names of the actual devices subscribed to the multicast group. Data packets are sent encapsulated from one branching point to the next, with the outer header containing the GUID and network addresses of the branching router, whereas the inner header containing GUID of the multicast group. Our detailed simulations in [25] show that name-based multicast scales elegantly as the group size and network size increase compared to inter-domain IP multicast [26].

D. Next-generation context-aware services

Finally, using the same name abstraction and the name resolution service, we would like to highlight how a rich set of context-aware services can be supported. Figure 5 shows one such context, where a survivor wants to send a message to "firemen dealing with incident X". As shown in the figure, the information layer is very rich and can include a complicated graph of relationships, including incident hierarchy (all incidents→incident X→X Fire), geographical hierarchy (US→<NJ, CA>), responder relationships (first responders→<police, firemen>) and so on. However, these can be mapped onto a flat naming plane using GUIDs through an object resolution service, as shown. Therefore, the information schema can be flat with no relationships (for example, individual devices), strictly hierarchical (for example, content names in a content centric network [27]) or a mix of all of the above (such as Wikipedia categories [28]), but can still be efficiently mapped into a flat namespace, by cleanly separating the information-space from the namespace. Next the name

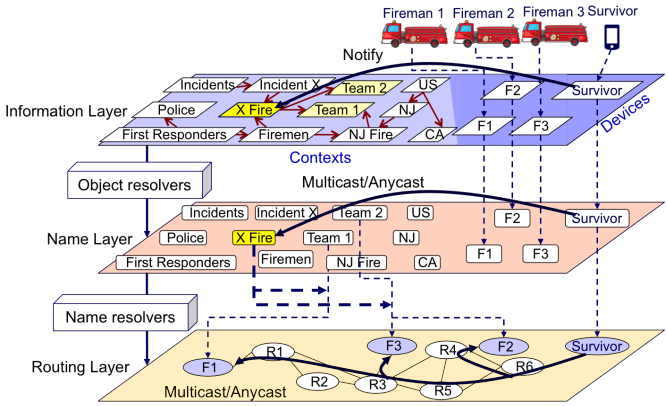

Figure 5. Context-aware services: Mapping a message, *Send* ("Fireman dealing with incident X", "Help message") from a survivor using names

resolution service can be updated to map these GUIDs to network addresses or recursively to other GUIDs.

The end-points do not need to be aware of the separation or the relationships, which can be handled by specific service managers related to each service. For example, in Figure 5, a disaster-management service manager, can determine the information schema, assign GUIDs and update the object resolvers and the name resolvers, such that when a survivor sends a contextual message (send to all all firemen handling incident X), the application on the end-host maps this context to an appropriate GUID and the network in-turn maps this GUID to an appropriate set of network addresses and anycasts or multicasts the message based on the service requirements. Ongoing work in MobilityFirst is focused on efficient design of the object resolvers and the name assignment services for enabling efficient contextual delivery use-cases. [29]

VI. Conclusion

This paper identifies the key set of requirements for a generic resolution service as a unified control plane for identifier-based architectures. Three alternative implementations of a global name resolution infrastructure were described and compared in terms of their design choices and trade-offs. Finally, the paper explained how advanced services such as mobility, multihoming and multicast and context-aware services can be supported using named-object service abstractions along with an efficient name resolution service.

Acknowledgment

The authors would like to thank Dr. Jiachen Chen, WINLAB, Rutgers University for his help with the figures and crucial feedback. This research was supported by the NSF Future Internet Architecture (FIA) grant CNS-134529.

References

[1] J. Saltzer, "On the Naming and Binding of Network Destinations." RFC 1498, 1993.

[2] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: Reorganizing the addressing architecture," in ACM SIGCOMM CCR, 2003.

[3] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the internet," in ACM SIGCOMM CCR, 2004.

[4] A. Venkataramani et al., "Mobilityfirst: A mobility-centric and trustworthy internet architecture," SIGCOMM CCR, 2014.

[5] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol." RFC 5201, 2008.

[6] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, "The Locator/ID Separation Protocol (LISP)." RFC 6830, 2013.

[7] C. E. Perkins, "Mobile ip," IEEE Communications Magazine, 1997.

[8] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses." RFC 6824, 2015.

[9] P. Mockapetris, "Domain Names - Concepts and Facilities." RFC 1034, 1987.

[10] T. Vu et al., "Dmap: a shared hosting scheme for dynamic identifier to locator mappings in the global internet," in IEEE ICDCS, 2012.

[11] A. Sharma et al., "A global name service for a highly mobile internetwork," in ACM SIGCOMM Computer Communication Review, 2014.

[12] Y. Hu, R. D. Yates, and D. Raychaudhuri, "A Hierarchically Aggregated In-Network Global Name Resolution Service for the Mobile Internet," in WINLAB TR 442.

[13] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis, "Lisp alternative topology (lisp+ alt)." RFC 6836, 2013.

[14] E. Nordström et al., "Serval: An end-host stack for service-centric networking," in Proc. of USENIX NSDI, 2012.

[15] "VMware View 5 with PCoIP, Network Optimization Guide White Paper," www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/view/vmware-view-5-pcoip-network-optimization-guide-white-paper.pdf, 2011 [accessed: 2017-02].

[16] G. Linden, "Make Data Useful," www.gduchamp.com/media/StanfordDataMining.2006-11-28.pdf, 2006 [accessed: 2017-02].

[17] "5G:A Technology Vision," www.huawei.com/5gwhitepaper, 2013 [accessed: 2017-02].

[18] Z. Gao, A. Venkataramani, J. F. Kurose, and S. Heimlicher, "Towards a Quantitative Comparison of Location-Independent Network Architectures ," in Proc. of ACM Sigcomm, 2014.

[19] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021," 2017.

[20] X. Liu, W. Trappe, and J. Lindqvist, "A policy-driven approach to access control in future internet name resolution services," in Proc. of ACM MobiArch, 2014.

[21] P. Karimi, I. Seskar, and D. Raychaudhuri, "Achieving high-performance cellular data services with multi-network access," in IEEE Globecom, 2016.

[22] S. Mukherjee, A. Baid, I. Seskar, and D. Raychaudhuri, "Network-assisted multihoming for emerging heterogeneous wireless access scenarios," in Proc. of IEEE PIMRC, 2014.

[23] D. Farinacci et al., "Protocol independent multicast-sparse mode (PIM-SM): Protocol specification," in RFC2362, 1998.

[24] J. Moy, "Multicast extensions to OSPF," in IETF RFC 1584, 1994.

[25] S. Mukherjee, F. Bronzino, S. Srinivasan, J. Chen, and D. Raychaudhuri, "Achieving Scalable Push Multicast Services Using Global Name Resolution," in Proc. of IEEE Globecom, 2016.

[26] D. Meyer and B. Fenner, "Multicast source discovery protocol (MSDP)," in RFC 3618, 2003.

[27] V. Jacobson et al., "Networking named content," in Proceedings of emerging networking experiments and technologies. ACM, 2009.

[28] "Wikipedia Categories," http://en.wikipedia.org/wiki/Help:Categories, [accessed: 2017-02].

[29] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "CNS: Content-oriented notification service for managing disasters," in Proc. of ACM ICN, 2016.