Demonstrating Context-Aware Services in the MobilityFirst Future Internet Architecture

Francesco Bronzino, Dipankar Raychaudhuri and Ivan Seskar
WINLAB, Rutgers University, North Brunswick, NJ 08902, USA
Email: {bronzino, ray, seskar}@winlab.rutgers.edu

***Abstract*—As the amount of mobile devices populating the Internet keeps growing at tremendous pace, context-aware services have gained a lot of traction thanks to the wide set of potential use cases they can be applied to. Environmental sensing applications, emergency services, and location-aware messaging are just a few examples of applications that are expected to increase in popularity in the next few years.**

The MobilityFirst future Internet architecture, a clean-slate Internet architecture design, provides the necessary abstractions for creating and managing context-aware services. Starting from these abstractions we design a context services framework, which is based on a set of three fundamental mechanisms: an easy way to specify context based on human understandable techniques, i.e. use of names; an architecture supported management mechanism that allows both to conveniently deploy the service and efficiently provide management capabilities; and a native delivery system that reduces the tax on the network components and on the overhead cost of deploying such applications.

In this paper, we present an emergency alert system for vehicles assisting first responders that exploits users location awareness to support quick and reliable alert messages for interested vehicles. By deploying a demo of the system on a nationwide testbed, we aim to provide better understanding of the dynamics involved in our designed framework.

I. Introduction

Context-aware systems offer entirely new opportunities for application developers and for end users by gathering context data and adapting systems behaviour accordingly [1]. Especially in combination with mobile devices these mechanisms are of high value and are used to increase usability tremendously. Context data, usually identified as external factors from the network environment, can extend across a wide variety of different fields including for example: environmental conditions, time, location, available energy, network attachments points, channel conditions, and communicating sources and destinations. Moreover, human related social states can be part of the analyzed environment, for example, if a user is in meeting, busy, or free.

The MobilityFirst (MF) future Internet architecture project represents a clean-slate Internet architecture which provides the necessary abstractions for creating and managing context-aware services. In particular, the architecture enables dynamic identification of endpoints based on context attributes through the use of named-object identifiers and global name resolution [2]. The current Internet primarily supports a primitive to send data to a specific IP address, which limits applications to cast all communication intent in those terms. This primitive is inflexible when the network location of the destination (or even the principals constituting the destination) is not known a priori. For example, several mobile or Internet-of-Things applications can benefit from context-aware primitives such as “send this message to all taxis in the Times Square area” or “request power consumption readings from devices in my living room”, which are cumbersome to implement in IP. In contrast, MF enables context-aware communication primitives based on attributes more general than just the network location and dynamically associates a context identifier to its constituent principals.

Our strategy is to develop an architecture where we can name environmental contexts that change where and how messages are routed and delivered. Application services could expect from such architecture improvements that span areas including security, communication efficiency, and energy management. In order to do so, we identify three strategic mechanisms that are required to accomplish the set goals: an easy way to specify context based on human understandable techniques, i.e. use of names; an architecture supported management mechanism that allows both to conveniently deploy the service and efficiently provides management capabilities; and a native delivery system that reduces the tax on the network components and on the overhead cost of deploying such applications.

II. MobilityFirst Protocol Stack

The context services framework design presented in this paper is based on the MobilityFirst future Internet architecture. In order to at best understand this design, we first need to introduce the architecture network protocol stack and its core technology components, previously presented at different venues [2], [3]. The MobilityFirst architecture’s main design centers around a new name-based service layer which serves as the “narrow-waist” of the protocol stack. 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 to a person, a group of devices/people, contents or even contexts. 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. Network services are defined by the source and destination GUID and a service identifier to specify the delivery mode such as multicast, anycast, multihoming, content retrieval or context-based message delivery. A hybrid name/address based routing scheme is used for scalability, employing a Global Name Resolution Service (GNRS) to dynamically bind the GUID to a current set of network addresses (NAs). The GNRS in MobilityFirst is a logically centralized service responsible for naming, security, and augmenting network layer functionality. The clean separation

*Research supported under NSF Future Internet Architecture - Next Phase (FIA-NP) grant CNS-1345295

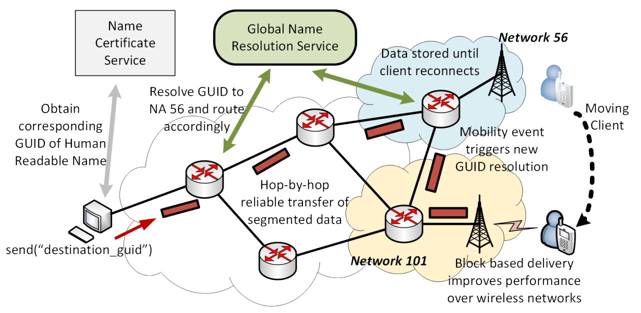

Name Certificate Service
Global Name Resolution Service
Data stored until client reconnects
Network 56
Obtain corresponding GUID of Human Readable Name
Resolve GUID to NA 56 and route accordingly
Mobility event triggers new GUID resolution
Moving Client
Hop-by-hop reliable transfer of segmented data
send("destination_guid")
Network 101
Block based delivery improves performance over wireless networks

Fig. 1: MobilityFirst architecture design and late-binding example.

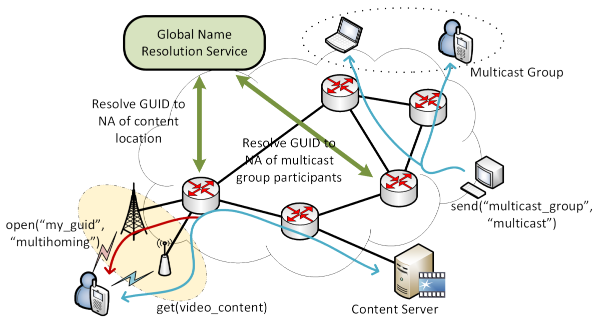

Global Name Resolution Service
Multicast Group
Resolve GUID to NA of content location
Resolve GUID to NA of multicast group participants
send("multicast_group", "multicast")
open("my_guid", "multihoming")
get(video_content)
Content Server

Fig. 2: Service abstractions provided via the client API.

of identity and location of endpoints is the key to achieve seamless mobility and in achieving trustworthiness ensuring that identities and their locations can be easily verified.

Data transport in MF is achieved by transferring blocks in a segmented manner using storage-aware routers unlike the current Internets end-to-end approach using TCP/IP. A block transport protocol transports blocks, or large chunks of contiguous data, in a hop-by-hop reliable manner as opposed to traditional transport protocols like TCP that transport small packets in an end-to-end rate-controlled manner. Segmented transport generalizes hop-by-hop transport to segments or a sequence of contiguous links terminated by storage-aware routers or endpoints. Congestion control across segments follows a segment-level back pressure approach similar to Hop [4].

For evolvability, MobilityFirst incorporates a virtualized compute layer that enables novel programmable network services to be deployed rapidly into the routing fabric. However, there are two key challenges to be addressed to make this approach practical. First, the compute layer must not introduce significant overhead on the default forwarding path for legacy traffic. Second, the compute layer must ensure resource containment for each service and isolation across different services for security and accountability. The compute layer in MF relies on such an API similar in spirit to software defined networks, but goes beyond the basic use case of virtualized network control and management to offer more general "packet cloud" services in the data path.

Finally, at the core of the architecture is a name-based networking abstraction that contrasts with the name-address conflated communication interface associated with Berkeley sockets and the TCP/IP stack. All network-attached objects in the MobilityFirst architecture enjoy direct addressability through long lasting unique network names or identifiers (we use GUIDs). This new GUID-centric network service API, first presented in [5], offers network primitives for basic messaging (*send, recv*) and content operations (*get and post*) while supporting several delivery modes natively supported by the MF network such as multihoming, multicast, anycast and DTN delivery.

III. Context Service

While the MobilityFirst architecture provides the necessary abstractions for creating and managing context-aware services, new mechanisms are required in order to fully exploit them. Starting from the three strategic mechanisms defined, i.e. an easy way to specify context, an architecture supported management mechanism, and a native multi-point delivery system, three fundamental technology components are exploited to design the MobilityFirst based context services framework.

Global Name Resolution Service: Recall that in MobilityFirst the GUID represents an abstract endpoint that is independent of the network topology. We leverage this independence to build contextual networking; our approach is to overload the GUIDs at many levels of the network. A GUID could thus represent many abstractions; for example, a cellphone, a person, or a group that is defined by context. In this sense, contextual communications are similar to *-cast types networking. For example, we could overload a GUID that names sending a message to a meeting. The system would have to first translate the GUID into a list of each person in the meeting. Further translation would be needed to identify the device each person is currently using. The Global Name Resolution Service is then the first key enabling technology that allows us to define context by collecting common set of network entities under a single name defining the context.

In-network computing capabilities: To enable future extensions to the network protocol without expensive hardware replacements and disruption, MobilityFirst builds in an optional and dynamically pluggable *compute plane*. Examples of such need are additions of new service types, new principal types, new addressing structures, or extensions to the end-to-end security protocol. We envision service providers and network operators to be able to perform relatively simple upgrades in the form of software updates and addition/replacement of pluggable hardware modules to extend the data plane functionality. Furthermore, we also postulate that such extensibility can enable third party application service providers (via the ISPs) to deploy either service end-points or service adaptors that are both closely integrated with the delivery path and best located to improve client experience. In-network computing capabilities are perfectly suited to deploy distributed and scalable management mechanisms to collect contextual information and manage membership. This last step is fundamental to the success of the architecture; for this purpose, in-network deployed services interact with the GNRS to translate collected contextual information into context names. In order to best support the distributed nature of the in-network components, service addressing is of fundamental importance. Generally, the compute layer will be hosted at key locations, whereas network delivery mechanisms will allow to select the best located replica of the service.

Multicast delivery: Multicast protocols have been long studied in the literature and different protocols and architectures have been proposed to support this type of delivery mechanisms. While these mechanisms are perfectly suited for static, tree based communications a different approach might be required for contextual applications. In particular, given the highly dynamic nature that defines context services, due to their multiple evolving factors, more flexible multicast delivery mechanisms are required. As importantly, in wireless environments, there is a desire to take advantage of inherent multicast/broadcast medium to enable efficient point to multipoint delivery services. The MobilityFirst architecture solves this problem by implement a lightweight multicast delivery system that, through name grouping in the GNRS, limits per group state within network elements taking per hop decisions on multicast splitting based on *Longest-common (LC)* lookahead techniques.

IV. MobilityFirst Based Prototype

In order to move towards testbed based experimentation we developed a prototype that included the main components that are part of the designed architecture. Due to space constraints, in this paper we will only introduce the main features of such prototype as first presented in [3] and the new components employed in the demo. As the MobilityFirst project addresses the feasibility of building systems and networks in a clean-slate design, it requires the development of such components from scratch. The result of this efforts consists in three main tools: a GNRS implementation based on DMap's design [6], a Click [7] based software router, and a multiplatform protocol stack and network API for clients. Applications and network services can be implemented as extensions of these basic elements.

Global Name Resolution Service. A GNRS implementation has been written in Java to provide a hardware and operating system agnostic implementation. The server is organized into several individual modules: network access, GUID mapping, persistent storage, and application logic.The application logic serves as a central point of coordination within the framework of the GNRS server daemon. The network access component ensures that the GNRS server is able to operate over any networking layer/technology without changes to the core code. This replaceable component currently supports IPv4 and MF routing. The GUID mapping module, relying partly on a networking implementation, enables the server to determine the remote GNRS hosts responsible for maintaining the current bindings of GUID values. Persistent storage is handled independently from the rest of the server and exposes only a very simple interface, mapping to the application messages available in the protocol. A BerkeleyDB provides both in-memory and on-disk storage for GUID bindings.

Routers. The software router is implemented as a set of routing and forwarding elements within the Click modular router. The router implements dynamic-binding using GNRS, hop-by-hop transport, and storage-aware routing as presented in Section II. It integrates a large storage, an in-memory *hold buffer*, to temporarily hold data blocks when destination endpoints experience short-lived disconnections or poor access connections. For dynamic in-network binding of GUID to NA, the router is closely integrated with the in-network GNRS by attaching to a local instance of the distributed service. A particular instance of this system, implements what we call a MobilityFirst access router, a router providing access connectivity to clients, supporting different access technologies (e.g. WiFi, WiMax, Ethernet). Thanks to the modular structure of Click, we are able to extend the software implementation with additional logic modules to support programmable network services. The router software also collects statistics at different layers of the protocol stack that can be reported through Click's control interface.

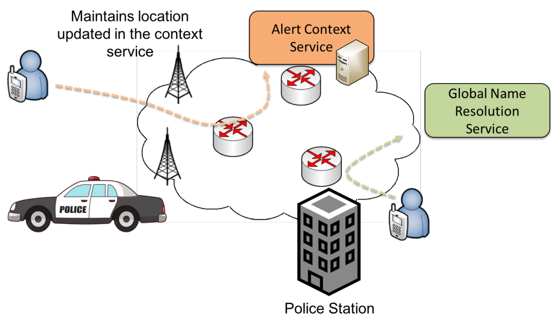
Host network stack and API. The host stack has been implemented on Linux and Android platforms as a user-level process built as an event-based data pipeline. The stack is composed of a flexible end-to-end transport to provide message level reliability, the name-based network protocol including the GUID service layer, a reliable link data transport layer, and a policy-driven interface manager to handle multiple concurrent interfaces. The device-level policies allow users to manage how data is multiplexed across one or more active interfaces. The previously introduced socket API [5] is available both as C/C++ and JAVA libraries and implements the name-based service API which include the primitives *send, recv, and get* and a set of meta-operations available for instance to bind or *attach* a GUID to one or more NAs, configure transport parameters in the stack, or to request custom delivery service types such as multicast, anycast, multihoming, or in-network compute. Similarly to the router implementation, the protocol stack collects and optionally reports traffic and resource statistics to a backend data repository.

All these components have been designed with flexibility in mind trying to reduce dependency from specific systems to a minimum. The set of basic requirements necessary to run any of these elements is minimal as any x86/x64 machine (physical or virtualized) running a recent Linux distribution can host them (the development has been based on Ubuntu 12.04 LTS).

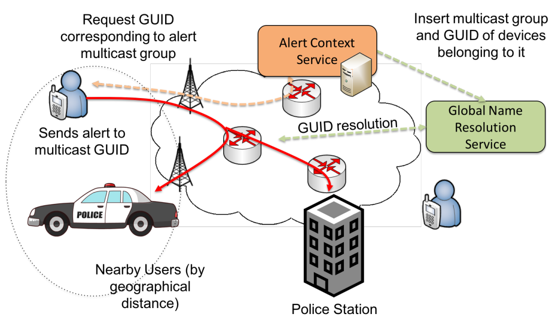
V. Emergency Service Demo

We use the developed MobilityFirst prototype presented in Section IV to implement a context services framework designed around the one described in this paper. The GNRS and the native delivery mechanisms, together with the newly implemented in-network context service, enable the MobilityFirst architecture to efficiently deploy context based services improving current architectures in a variety of ways. In order to demonstrate the mechanisms involved in implementing context based services and to showcase some of its benefits, we developed and deployed a contextual application that implements an alert system for vehicles assisting first responders. This service is aimed at providing ways to quickly and reliably transmit emergency messages to group of receivers identified by the conveniency of their geographical position or by the authoritative importance in the emergency matter. To better understand this consider an example where a car accident occurs on the roads of a metropolitan area; in this case, three potential candidates emerge for providing assistance: first, defined by a very small geographical area, other close passbyers could be informed in order to provide first assistance; second, based on a larger area, emergency vehicles such as ambulances could be alerted; finally, given the importance of informing the central authorities, a police station could be included due to its relevance on emergency matters.

The system architecture deployed, as depicted in Figure 3, includes, together with the defining technologies of the MobilityFirst architecture, two core components: the in-network compute management framework and an Android


Maintains location updated in the context service
Alert Context Service
Global Name Resolution Service
POLICE
Police Station

(a) Creation and management of context groups based on geographical location


Request GUID corresponding to alert multicast group
Alert Context Service
Insert multicast group and GUID of devices belonging to it
Sends alert to multicast GUID
GUID resolution
Global Name Resolution Service
POLICE
Nearby Users (by geographical distance)
Police Station

(b) Dynamic creation of routing information for multicast based delivery

Fig. 3: Alert system architecture based on the MobilityFirst context services framework

based smartphone application.

In-network compute management. A *Ruby* based context service is implemented to enable in-network computing capabilities that provide the distributed service described in the previous section. This service is then deployed in proximity of favorably located routers by directly connecting it to the Routers API that provides access to the in-network compute capabilities. Context based operations are exposed using web based APIs (i.e. using REST) and provide the core required management functionalities to create and manage context groups and report context information (e.g. users current location). Moreover, to implement the required management operations, the GNRS API is exploited to interact with the distributed service. A web interface is also developed for human based interactions to better understand the presented demo.

Contextual application. We exploit the available network host stack and API to deploy the context application components. In particular, an Android smartphone application has been developed providing two groups of functionalities: first, the core alert operations that allow for quickly broadcasting the alert message to the current context members and to generate notifications for received emergency messages. Second, management operations are allowed, providing ways to interact with the service management framework and create, join and leave given contexts, by exploiting locally collected GPS coordinates.

The demo will be centered around three key operations: 1) creation and management of context groups based on geographical location of active users (Figure 3(a)); 2) dynamic creation of routing information for multicast based delivery (Figure 3(b)); 3) evaluation of efficiency in providing the service. The demo will use the GENI nationwide testbed infrastructure [8] to deploy 3 different locations from where wireless nodes (i.e. Android phones) access via WiFi and WiMax the MobilityFirst architecture. Our current deployment spans 7 GENI sites across the US. 14 Xen VMs (2 VMs per site) each with 1 GB memory and one 2.09 GHz processor core provide us with the possibility to run one router per location and use the other node for application or services. All routers have a core-facing interface connected to a layer-2 network that connects all seven sites. This was setup using a multi-point VLAN feature provided by Internet2's Advanced Layer-2 Service (AL2S).

In order to better understand all the dynamics involved in the demo, a visualization system has been developed using web-based technologies (e.g. javascript, Google Maps, etc.). This visualization will provide information about different components state. In particular it will show the following information: location of mobile devices and their current status (idle, sending alert, alerted by other nodes), GNRS entries that enable context management and multicast routing, traffic crossing all the nodes in the system and finally important general events spanning all the networking layers of the architecture.

VI. Conclusions

In this paper we presented the details of a context service framework that exploiting core components of the future Internet architecture called MobilityFirst, allows the deployment of flexible and efficient context based services and applications. To demonstrate its properties, a demo using a working prototype has been presented showcasing the core functionalities of the system, using a set of programmable and mobile nodes distributed across different sites on a US testbed. In the future we plan to extend the showcased demo components to support a variety of context services, allowing other research and industry players to exploit the available framework to further push the state of the art.

References

[1] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[2] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[3] F. Bronzino, D. Raychaudhuri, and I. Seskar, "Experiences with testbed evaluation of the mobilityfirst future internet architecture," in *Networks and Communications (EuCNC), 2015 European Conference on*. IEEE, 2015, pp. 507–511.

[4] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani, "Block-switched networks: a new paradigm for wireless transport," in *Proc. of NSDI*, 2009.

[5] F. Bronzino, K. Nagaraja, I. Seskar, and D. Raychaudhuri, "Network service abstractions for a mobility-centric future internet architecture," in *Mobiarch 2013*. ACM, 2013, pp. 5–10.

[6] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *ICDCS 2012*. IEEE, 2012, pp. 698–707.

[7] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router," in *ACM Transactions on Computer Systems*. Citeseer, 2000.

[8] "Global environment for networking innovations (GENI), NSF program solicitation," http://www.geni.net, 2006.