Experiences with Testbed Evaluation of 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*—This paper reports results and experiences from the validation process performed to experimentally evaluate the design of the Future Internet Architecture MobilityFirst. Next, we discuss possible evaluation strategies that take into account the desired scale and degree of realism necessary for validation. Specific examples of experimental evaluations for the MobilityFirst architecture running on the ORBIT and GENI testbed are given. These include routing and name resolution scalability experiments on ORBIT, service-level evaluations on GENI, and real world experimentation with end-users using both ORBIT and GENI capabilities. Selected results from these experiments are presented and discussed in context of MobilityFirst evaluation goals.**

I. Introduction

Research programs exploring clean-slate architectures are steadily growing over the last few years. Among others NSF future Internet architecture (FIA) [1] and Future Internet Design (FIND) [2] in the US and FP7 Future Networks [3] in Europe are example of the increased research activity on this subject. As most of these projects present new Internet architectures and protocols often starting from a clean slate approach, it is important to understand the different tools available to carry out the required validation work. Large-scale experimental networking, such as GENI [4] in the US and FIRE [5] in Europe, aim to provide the necessary infrastructures and environments for validation of new protocols. As the availability and the quality of these tools improve by the year, it is more and more important to understand the advantages and the limits of validation through alternative testbed deployments.

The MobilityFirst project [6], funded by the NSF FIA program, recognizes the predominance of mobile networking and aims to directly address the challenges of this paradigm shift. MobilityFirst is designed around the principle that mobile devices, and their associated applications, should be treated as first-class Internet citizens. There are many challenges associated with integrating wireless/mobile communications as a core requirement of the Internet architecture - these include mobility, varying levels of connectivity, intermittent disconnection, multiple network attachment points per device, and a desire for flexible, group-based routing paradigms. Current Internet protocols, such as TCP/IP, are limited in their support for these challenges as they were built using a connection oriented model. Instead, MobilityFirst takes advantage of Moore's Law improvements in processing and storage, shifting some intelligence into the network and decreasing the emphasis on end-to-end functions.

*Research supported by NSF Future Internet Architecture (FIA) grants CNS-1040735, CNS-1345295, and CNS-1040043

Following the MobilityFirst project's initial design phase, a growing focus has been directed towards exploring the feasibility of this design using different experimental environments and facilities. In this paper, we present results and experiences obtained via experiments with the FIA MobilityFirst prototype on two different testbeds: ORBIT [7] and GENI. Starting in Section II from an introduction to the MobilityFirst protocol stack and main architectural components, we present in Section III a collection of experiments that provides an idea of how these testbeds have been used together with the developed MobilityFirst protocol stack to meet different validation requirements; finally we conclude the paper in Section IV.

II. MobilityFirst Protocol Stack

The MobilityFirst [8] 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, multi-homing, 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 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 [9]. The segmented block transport protocol enables content

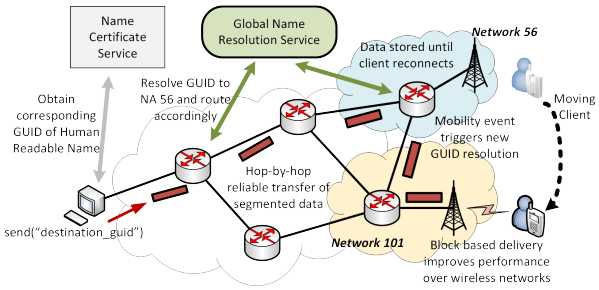

Fig. 1: MobilityFirst architecture design.

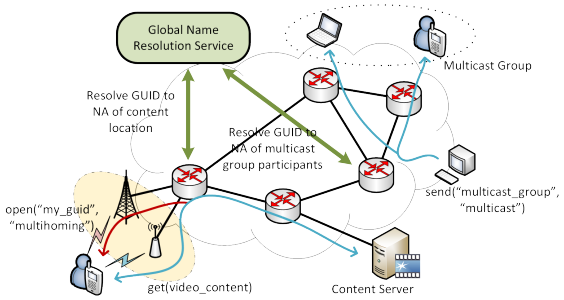

Fig. 2: Service abstractions provided via the client API.

to be transferred as one or more blocks each with a self-certifying content identifier, which enables on-path MF routers to opportunistically cache and serve popular content.

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. Feasibility analysis of these services is thus a key aspect in the evaluation of the architecture.

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 [10] offers network primitives for basic messaging (*send, recv*) and content operations (*get and post*) while supporting several delivery modes innately supported by the MF network such as multihoming, multicast, anycast and DTN delivery. Combined with the GUID indirection and grouping (GUID mapped to one or more other GUIDs) concepts supported by the naming services, the new communication API can produce novel addressing and delivery capabilities only indirectly possible (and with certain in-efficiency) in today's IP architecture. Supporting the variety of services that are introduced in this API are then fundamental towards the validation of the architecture design.

III. Validating the Architecture Design

Validating the design of a novel architecture such as MobilityFirst requires a comprehensive effort that spaces across different experimental techniques. This is due to the different requirements and goals that are part of the process. In order to understand the value of alternative experimental testbeds it is indeed important to identify the particular functional aspects (scale, performance, protocol validity, etc.) that need to be evaluated. In many cases, performance verification at scale is still best suited for simulated environments, such as NS3. This approach applies well to classical network problems such as aggregate routing evaluations looking at different metrics such as protocol overhead and achievable throughput. A tradeoff with the scale of the experiments can be desirable in order to obtain higher levels of realism. Performance critical systems and elements of the architecture might require this approach in order to validate their feasibility. Finally, even higher levels of realism can be obtained by means of deploying the architecture with real end-users that can interact with the network and the deployed services through specific application.

Figure 3 summarizes realism and scale achieved by different evaluation methods/testbeds, and the typical sequence of simulation to testbed evaluation to large-scale user trials. In the following sections we will use specific examples of experiments and scenarios for the main steps: large-scale simulation and experimental evaluation through prototyping and use of Emulator testbeds and real-world deployments. Starting from the initial validation of the architecture routing protocols performed through simulation models, we describe the main components that constitute our protocol stack implementation that has been the basis of experiments performed on two different testbeds: ORBIT and GENI.

A. Protocol Evaluation Through Simulation

MobilityFirst introduces, as part of its design, a novel inter domain routing protocol called Edge-Aware Inter-Domain Routing protocol (EIR) [11]. In an effort to better adapt to changes at the edge network and better support for multi-path, multi-homing and multi-network operation, the EIR protocol defines abstract network entities such as aggregated routers, called aNodes, and virtual links between them called vLinks. Aggregated information about aNodes and vLinks in a network are disseminated to every other AS through network state packets (nSPs) in order to provide a complete view of the network graph.

To evaluate the protocol at a global scale, we designed and implemented a custom-built discrete event-driven simulation to reflect the actual Internet topology according provided by DIMES [12]. Through the simulation, we evaluate routing event dissemination overhead and dissemination latency both for EIR and for GSTAR [13], our intra domain routing protocol. The simulator takes AS-level topology of the current Internet as the network model by extracting the following real-measurements from the DIMES database: (i) Connectivity graphs containing 26235 ASs and 100K links between them, (ii) Link latency between each pair of AS. As a sample result, we show in Figure 4 the obtained routing overhead generated

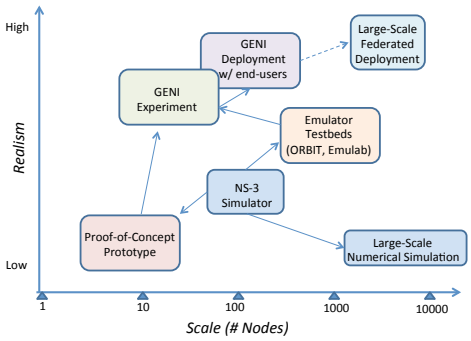

Fig. 3: Realism vs scale provided by different network evaluation methods.

by the dissemination of link state packets under different transmission schemes.

B. Protocol Stack Implementation

In order to move towards testbed based experimentation we needed to develop a prototype that included the main components that are part of the designed architecture . 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 [14], a Click [15] 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. Moreover, we developed the necessary support to automate experimentation using the OMF [16] framework and provide statistic collections through OML [17].

Global Name Resolution Service. A GNRS implementation has been written in Java to provide a hardware and operating system agnostic implementation. Wherever possible, standard libraries are utilized to provide the required functionality, and only the application logic needed to be written by hand. 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. It integrates a large storage, an in-memory *hold buffer*, to temporarily hold data blocks when destination endpoints during 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. Access routers also implement a rate monitoring service that tracks the available bandwidth for each attached client. For WiMAX networks, the rate is obtained by querying the WiMAX base station when possible which exports the most recent downlink bitrate allocated to each client by the scheduler based on a client's location, client offered traffic, and overall load on the BSS. A similar rate monitoring capability is implemented for WiFi Access Points using standard 802.11 netlink configuration utilities. Thanks to the modular structure of Click, we are able to extend the software implementation with additional logic modules to support programmable network services. An example of this will be presented in Section III-D. The router software also collects statistics at different layers of the protocol stack that can be reported through the use of an OML-based monitor that runs as a separate process can interact with the router through Click's control interface.

Host 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 user to manage how data is multiplexed across one or more active interfaces. The previously introduced socket API [10] 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 OML 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). Future developments will also focus in providing implementation for emerging technologies (e.g. SDN and hardware enhancements).

C. Scalability Results Using ORBIT

The Orbit testbed is a two-tier wireless network emulator/field trial designed to achieve reproducible experimentation. Its main facility is the radio grid testbed which uses a 20x20 two-dimensional grid of programmable radio nodes which can be interconnected into specified topologies both using reproducible wireless channel models allowing fine grained control over connectivity resources, both using a fully connected 1Gbit ethernet based layer 2 network. Thanks to its large set of resources it provides a perfect environment to support realistic evaluation of protocols and applications up to medium scale.

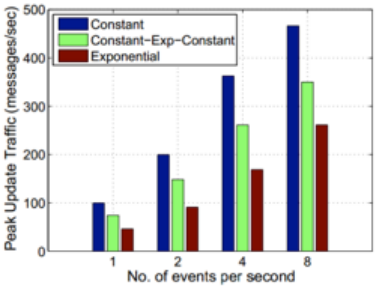

Fig. 4: Routing event update overhead [11]

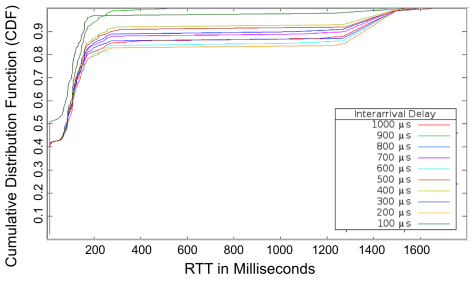

Fig. 5: Client RTT for GNRS lookups for 200 servers and 100K GUIDs

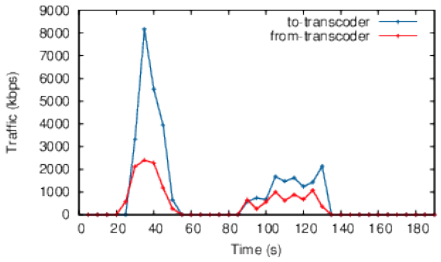

Fig. 6: Traffic between Rutgers MFR and transcoder

The GNRS implementation has been evaluated at medium-scale on the ORBIT grid using the wired infrastructure. The experiment was intended to evaluate basic metrics for the distributed system such as: computation cost, server throughput, query latency and load distribution. Previous simulation based results [14] had shown that the design could achieve latencies under 100ms for the 90th percentile of client requests. Exploiting the 400 nodes of the Orbit main grid, we used the Jellyfish [18] model to map the real Internet onto the testbed: each grid node represented an AS, and the number of nodes for different layers and links between different layers was proportional to Internet AS-level topology. We analyzed the real-Internet datasets from DIMES and CAIDA [19], and generated the network topology based on that. Figure 5 shows some validation experimental results obtained from such ORBIT experiments, in particular the latency experienced upon issuing client requests. From the graph, we notice that 90% of the experienced latencies were in the range of 200 ms, somewhat higher than the 100ms predicted by simulation. We believe some of this difference can be attributed to the specifics of the Jellyfish model as mapped on to the ORBIT grid, and we are now conducting experiments with other topologies as well.

D. Service Deployment in GENI Based Realistic Scenarios

The GENI nationwide testbed offers an infrastructure with Internet2 and NLR backbones connecting several university campuses. The wide area hosts, interconnected by a 1/10 Gbit core network allows for a realistic deployment and evaluation of MobilityFirst architecture and protocols. Participation from access networks and mobile clients at collaborating campuses when combined with deployments at the GENI core can establish reasonably large size networks of the order 10s to a few hundred nodes with realistic wide-area network conditions. In an effort to provide long term experimentation for the architecture, MobilityFirst has been assigned a permanent slice, that provides always running resources distributed over 7 GENI sites. Figure **??** shows the distribution of these locations. 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). Routers at three sites (viz. Wisconsin, Rutgers, NYU) are configured with a second interface connecting to the local wireless network (WiMAX). Mobile wireless or emulated clients connect to MF network through this interface. Routers are each configured with 500 MB of hold buffer space, and have access to a GNRS service instance co-located on each of the seven sites locations.

We used the above setup to evaluate the feasibility of designing and implementing in-network services and deploying them into a realistic testbed. As an example of these experiments, we will describe a rate-adaptation service aimed at enhancing DASH video streaming applications mobile devices that experience variable connection quality. This implemented transcoding service is deployed into the edge network and combines both caching and transcoding functionalities. As introduced in Section II, MobilityFirst offers native support for a variety of delivery services through the use of service type and service options. We extended the basic set of name-based communication primitives to support addressing compute-layer service type and options. For this particular case an interface was provided in order to specify the quality of the video segments as an option in an network extension header. The routers' code was extended to take forwarding decisions based on these parameters.

We deployed a MF-enabled DASH server ran at the Wisconsin site, while a VLC client was at Rutgers connected over WiMAX. The VLC client was modified to perform MobilityFirst content requests instead of normal HTTP requests. Routers use the previously described how the client link is monitored at the access router via the information exported by the WiMAX base-station. Any variation in available bandwidth due to mobility or load is available almost instantaneously at the router. As the used client was a fixed node and hence sees little variation in its access link properties, we emulated the client mobility by intentionally modifying the bandwidth reported by the measurement service to drop below the rate required to support smooth video streaming at the original encoded bitrate. We used this use case to evaluate feasibility of integration of services into our network logic. For example Figure 6 reports the steered traffic and the response traffic with transcoded segments at the router-transcoder link. It can be observed how at two different moments, the transcoded effectively reduced the traffic load on the wireless link.

E. Real World Experimentation with End Users

A contextual messaging application, Drop It, was developed using the name-based networking abstractions provided by MobilityFirst, which allows users to drop messages at particular locations, and to pick up messages left by others at the same location. MobilityFirst allows locations (contexts, in general) to be assigned unique names (a GUID globally unique

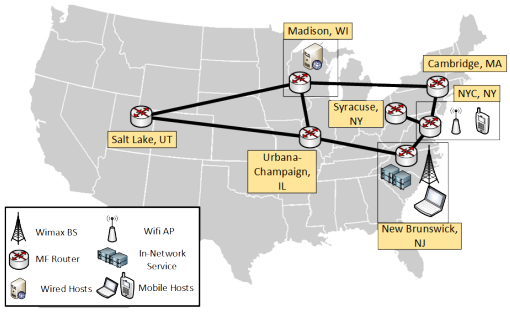

Fig. 7: Prototype components deployed on the GENI testbed

ID) which help identify them for network operations such as send, recv or get (for named content retrieval). Locations in physical space can be defined (or fenced) by a set of GPS coordinates, for example, and a persistent GUID can be assigned to them by a well-known service. Next, by maintaining meaningful address mappings for a location GUID in the GNRS, endpoints can send and receive messages to/from this context. For instance, a mapping of location GUID to the set of all phones that dropped messages at that particular location can enable a pure peer-to-peer realization of the contextual messaging service, where the pick-up can be implemented as an efficient multicast request to each of the phones by using MobilityFirsts get API. We run a demonstration of this application using resources across five of the seven sites within the long-running MobilityFirst network deployment. The two edge sites at NYU Poly and Rutgers WINLAB, hosted both WiFi and GENI WiMAX access networks that were connected back to the GENI core. Ten Android phones (some with dual WiFi/WiMAX interfaces), each running the MobilityFirst protocol stack and the Drop It application were carried around by volunteers (except two which were static at Rutgers and remotely accessible) who performed message drop and pick-up operations at several preset locations on the demo floor. Location GUID mappings were provided through the use of QR-code tags to overcome difficulties of using GPS indoors.

While no performance statistics were collected for this particular experiment, it was indeed useful to understand how user behavior affected our prototype implementation. One major was therefor encountered and solved involving a particular energy saving feature implemented in the WiFi module of the utilized phones (Samsung Galaxy S2) that triggered while the phone screen was turned off, negatively affecting the lower communication protocols of our prototype.

IV. Conclusions

In this paper we presented our experiences with the experimental validation effort carried out under the MobilityFirst Future Internet Architecture project. Following an initial simulation phase, we built a prototype that was deployed in two different testbeds in order to perform experiments aimed at providing a deeper understanding of the architecture design. This experiments ranged from critical services validation at medium scale to service deployments in real-world testbeds. In the future, we plan to further explore the use of the protocol stack in realistic network environments. This will include three different environments for trial deployments including a content production and delivery network in collaboration with a local broadcaster and a public service emergency notification system in partnership with a broadband wireless ISP. Moreover, through the open release of the code base of our protocol stack implementation and management tools, we plan to enable further community-based evaluation and experimentation in different federated networks.

References

[1] "NSF FIA project," http://www.nets-fia.net.

[2] "Networking technology and systems: Future internet design (FIND), NSF program solicitation," 2007.

[3] "FP7 information and communication technologies: Pervasive and trusted network and service infrastructures, european commission," 2007.

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

[5] M. Lemke, "Position statement: FIRE, NSF/OECD workshop on social and economic factors shaping the future of the internet," January 2007.

[6] "MobilityFirst Future Internet Architecture Project," http://mobilityfirst.winlab.rutgers.edu/.

[7] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking Conference 2005*, vol. 3. IEEE, 2005, pp. 1664–1669.

[8] 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.

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

[10] 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.

[11] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri, "Eir: Edge-aware interdomain routing protocol for the future mobile internet," *WINLAB, Rutgers University, Tech. Rep. WINLAB-TR-414*, 2013.

[12] Y. Shavitt and E. Shir, "Dimes: Let the internet measure itself," *ACM SIGCOMM CCR*, vol. 35, no. 5, pp. 71–74, 2005.

[13] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "Gstar: generalized storage-aware routing for mobilityfirst in the future mobile internet," in *Proceedings of the sixth international workshop on MobiArch*. ACM, 2011, pp. 19–24.

[14] 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.

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

[16] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "Omf: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 54–59, 2010.

[17] M. Singh, M. Ott, I. Seskar, and P. Kamat, "Orbit measurements framework and library (oml): motivations, implementation and features," in *Tridentcom 2005*. IEEE, 2005, pp. 146–152.

[18] Y. He, G. Siganos, and M. Faloutsos, "Internet topology," in *Encyclopedia of Complexity and Systems Science*. Springer, 2009, pp. 4930–4947.

[19] "CAIDA: The cooperative association for internet data analysis," http://www.caida.org/.