State-of-the-Art Multihoming Solutions for Android: a Quantitative Evaluation and Experience Report

Luca Stornaiuolo* and Paolo Bellavista†
DISI – University of Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
Email: *luca.stornaiuolo@studio.unibo.it, †paolo.bellavista@unibo.it

***Abstract*—The technical challenges associated with multihoming management in mobile systems and applications have attracted relevant research activities, as demonstrated by the wide related literature of the recent years. However, only very recently some multihoming solutions and techniques have started to be applied in industrially-relevant platforms and cases, often in a limited and very controlled way. This paper has the specific and focused objective of reporting a fresh state-of-the-art overview of the maturity of multihoming solutions for Android and to describe our practical experience of multihoming configuration and evaluation over off-the-shelf Android devices. In particular, we report the experience made while considering the relevant Locator/Identifier Separation Protocol (LISP) and especially LISPmob support solutions, by *i)* showing how to efficiently configure LISPmob on non-rooted Android devices; and *ii)* thoroughly analyzing its supported features towards the abstraction of seamless mobility. In addition, the paper includes a qualitative and quantitative comparison of different multihoming support approaches, as well as original experimental results about the performance of LISPmob over Android terminals.**

I. Introduction

Multihoming is widely recognized as a relevant feature for mobile systems/applications support, capable of significantly and suitably enhancing the abstraction layer of application development while potentially improving the efficiency of handoff management thanks to non-application-driven resource management. Therefore, the technical motivations why multihoming support solutions are relevant are many, including *i)* they help in minimizing the downtime due to connection failure by potentially realizing the abstraction of reliable Internet connectivity; *ii)* they can take advantage from policy-based routing to choose the most appropriate link to exploit for an application-level connection; *iii)* they can support and manage multiple broadband links to different ISPs, which is usually more cost-effective than a single high bandwidth link; and *iv)* they can enable dynamic, transparent, and efficient load distribution over all the available links [1].

A very few recent papers have started to address the relevant issue of surveying the status of maturity of multihoming support implementations. In particular, [2] addresses the general topic by presenting the primary protocols available that adopt a locator/identifier split approach. More specifically related to our paper contribution, [3] provides a very recent description of multihoming solutions for standard Linux distributions, while [4] reports about available solutions for multihoming over Android devices that can be rooted in the considered application domains and execution environments. To the best of our knowledge, our contribution is strongly original in describing *i)* a fresh state-of-the-art overview of the maturity of multihoming solutions for Android with the focus on more usual non-rooted devices; and *ii)* our practical experience of multihoming configuration and evaluation over non-rooted off-the-shelf Android terminals. In particular, we report our practical experience made while considering the relevant Locator/Identifier Separation Protocol (LISP) and especially LISPmob support solutions. Original technical contributions of this paper include an experience report and some indications of general applicability on how to efficiently configure LISPmob on non-rooted Android devices.

The remainder of the paper is structured as follows. Section II is dedicated to surveying the primary multihoming protocol solutions in the literature. Section III, specifically focuses on multihoming solutions for the Android platform and on their integration/availability over Android devices for mass-market users. In Section IV testbed validations, experimental results, and a discussion on the efficiency of the currently available implementation of LISPmob for Android follow, while Section V ends the paper with conclusive remarks and main directions for our future work.

II. State-of-the-Art Multihoming Solutions: an Overview

The relevance of multihoming is well recognized since several years [5], [6]. In principle multihoming support can involve either a single or multiple layers of the ISO/OSI protocol stack. Each layering approach has demonstrated to have specific advantages and disadvantages, which must be thoroughly considered, often in conjunction with deployment considerations and constraints. In the following we will concentrate on proposals that adopt the so-called locator/identifier split approach, requiring that the transport layer identity of endpoints is decoupled from the network layer locators of hosts in order to allow multiple forwarding paths for a single transport session. The locator/identifier mapping must be ensured by a dynamic process, so that a session can include different features, such as constant endpoint identifiers throughout the session lifetime, and modification of locators to maintain end-to-end reachability [7].

By delving into some finer details about the main solutions currently available in the related literature, Mobile IPv6

978-3-901882-77-7

(MIPv6) and its extension Network Mobility (NEMO) [8] have been standardized by IETF to allow a node or a network to remain reachable at the same IPv6 address and prefix while moving and changing their point of access to the Internet. Even if this solution has achieved some relevant results, it must be said that its implementation development, experimentation, and in-the-field diffusion got stuck at 2007 [9], [10], [11].

It is well recognized that protocols like Site Multihoming by IPv6 Intermediation (SHIM6) or Host Identity Protocol (HIP) may provide articulated multihoming features along with relative easy implementation, failure detection and recovery, or ubiquity and security support. However, none of them can be considered as the ideal solution [12]. In fact, while SHIM6 may break the functionality of other protocols [7], HIP implementation complexity is the reason of its recent de facto cooldown of the researcher community in the field.

Another relevant solution in the literature is the Identifier/Locator Network Protocol (ILNP). Even if it has shown some significant pros, ILNP has demonstrated non negligible disadvantages in terms of *i)* required changes to standard DNS and the associated deployed equipment; and, even more relevant, *ii)* the "philosophy" of disruptive approach behind the protocol itself, calling for significant re-deployment and non-back-compatible evolution of the existing installation base.

The practical experience of experimentation of the Stream Control Transport Protocol (SCTP) shows that it would represent a feasible multihoming solution by itself [13], [14], but with some non-negligible restrictions. Its pros are many and valuable, and its independence to the network layer is crucially important. But it has to be supported and enabled with ad hoc modules running at hosts at the OS kernel layer.

Nowadays LISP appears as the best and most promising tradeoff solution available. Sponsored by Cisco Systems, LISP is born to solve a number of problems risen up after more than a decade of relatively unchanged Internet infrastructure and very slow wearying transition to IPv6, which in fact is still ongoing. LISP has demonstrated to be the most efficient solution in terms of scalability [15]—a reason that alone could justify its future usage. Moreover, LISP does not require any substantial change to the existing legacy infrastructure, except for the entities which specifically support it, even if for both outgoing and incoming packets a processing latency is added at the edge of the network [5]. Hence, considering that the most significant LISP performance limitations relate to network devices' hardware technology, we claim that LISP can play the role of the most widespread multihoming solution in next generation networks, possibly (but not necessarily) in association with SCTP.

Even if the multihoming literature is relatively rich of solutions and proposals [16], [17], [18], [19], also due to the large set of different possibilities and the complexity of efficient implementation of the above features, multihoming is supported neither by any widespread user application yet, nor by standard network devices. Table I concisely reports some main features of the considered protocols, while Table II their pros and cons.

Protocol	MH	R	U	L	F	S	P	Sec
NEMO	End-host	✗	✓	✗	✗	✗	✗	✓
SHIM6	End-host	✓	✗	✗	✗	✓	✗	✗
HIP	End-host	✓	✓	✗	✗	✓	✗	✓
ILNP	End-site	✓	✓	✓	✓	✗	✗	✗
LISP	End-site	✓	✓	✓	✗	✓	✓	✗
SCTP	End-host	✓	✓	✓	✓	✗	✓	✓

TABLE I. Primary Features of most Widespread Multihoming Protocols.

R: Resilience, **U**: Ubiquity, **L**: Load balancing/sharing, **F**: Flow distribution, **S**: Scalability, **P**: Policy, **Sec**: Security, **T**: Transport layer, **N**: Network layer

Protocol	Pros	Cons
NEMO	Mobility, Handover latency	Requires changes to hosts
SHIM6	Deployment	Mobility, Security
HIP	Compatibility, Security	Deployment, Policy
ILNP	Semantics, H-MH support	Requires changes to DNS
LISP	Scalability, Flexibility	Encapsulation overhead
SCTP	Security, Flow control	Requires changes to hosts

TABLE II. Multihoming Protocols Pros and Cons

T: Transport layer, **N**: Network layer

III. Multihoming in Android

A. State-of-the-Art Multihoming Support in Android

The core of Android network management is the `ConnectivityManager`. Added in API level 1, the class is responsible for all the operations between the Android platform and the currently available networking opportunities. Until API level 18, the Android system was officially able to manage only two networking opportunities, by selecting to activate (to the purpose of useful data communication at the application layer) only one of them at any given time. The Android platform automatically assigns the role of default network to the one with the best received signal strength and, based on that, the `ConnectivityManager` manages its *fail over* procedure.

Also with the growing number of networking opportunities that Android can support now, the approach with a single "default network preference" starts to be recognized in the community as a bit "old-fashioned". In most recent versions, the trend is in favor of alternative possibilities returning an array of prioritized connectivity opportunities [20]. This actually has a good impact on ubiquity and on efficiency of system interfaces' management. But anyway it does not include any strong and complete multihoming support.

Only a relatively recent solution can allow Android devices to enable all the on-board network interfaces, even if this is not permitted by default Android policies [21], [4]. With these rules, it is impossible to get the two interfaces running at the same time and there is no standard, available, and widespread management interface to allow users to change this default option. Let us note that advanced IP routing, multiple routing tables, and network filtering are features available in the Android kernel and that should be enabled too to contribute to seamless mobility and multihoming support. Another option is to implement an ad hoc multihoming solution: this may be achieved in different ways but generally a multihoming protocol has to be supported somehow.

A relevant distinguishing property, in current Android systems, is whether we are in a deployment environment where we can assume the possibility of exploiting rooted devices or not. If the choice is to root Android, it is possible to enable SCTP in Android's kernel and hence provide a good multihoming support for the host. As an alternative, so far HIP is available on rooted Android platforms thanks to an experimental porting of HIPL. Otherwise, if the choice is to apply to non-rooted Android devices to maximize market penetration rate, the currently viable solution is to adopt a more complex solution like LISPmob.

B. LISPmob

LISPmob is an open-source LISP and LISP Mobile Node (LISP-MN) implementation for Linux, Android, and OpenWRT. With LISPmob, hosts can change their network attachment point without losing connectivity, while maintaining the same IP address. Even though several interfaces can be managed by LISPmob at the same time, on Android platforms they can only be used in an "active-backup" fashion (that is, no more than one interface used at once) because of the previously illustrated motivations.

At the time of this paper writing, two versions of LISPmob for Android exist: the non-rooted and the rooted versions. In order to properly configure and validate the performance of non-rooted LISPmob support in Android, the first steps are the setup of the `lispd` configuration file and the execution of `lispd`. The `lispd.conf` setup can be done manually by editing the file and placing it in the LISPmob root folder, or using the proper activity with the parameters given for the beta network.

IV. Experimental Evaluation and Discussion

Our experiments were made in order to quantitatively evaluate LISP-MN performance using LISPmob on non-rooted Android devices in comparison with standard IPv4. First, using LISPmob, we collected the ping response times towards a set of active EIDs via LISP IPv4-in-IPv4 tunneling; then, switching off LISPmob, we also collected the respective ping times via standard IPv4. The starting point is represented by 2 sets of $k = 53$ active EIDs. For each active EID we collected $N = 100$ ping times. So we eventually have k average values related to the standard IPv4, and k related to the LISP IPv4-in-IPv4, for a total amount of 10600 ping samples. For our purpose, each set may be considered as a discrete random variable.

We define the general sample mean vector $\bar{\mathbf{h}}$ as a column vector whose j-th element $\bar{\mathbf{h}}_j$ is the average value of the N observations of the j-th variable, that is $\bar{\mathbf{h}}_j = \frac{1}{N}\sum_{i=1}^{N} h_{i,j}$, where $j = \{1, 2, \ldots, k\}$. Thus, we define $\bar{\mathbf{x}}$ as the column vector containing the average of all the observations for each variable related to the standard IPv4, $\bar{\mathbf{y}}$ as the one related to the LISP IPv4-in-IPv4, and $\bar{\mathbf{\Delta}}$ as the column vector whose j-th element $\bar{\Delta}_j$ is the difference value between the sample means y_j and x_j.

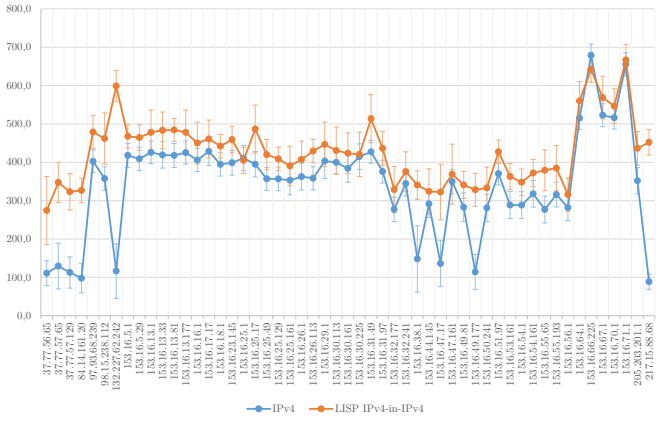

Fig. 1. Plot of $\bar{\mathbf{x}}$ (IPv4) and $\bar{\mathbf{y}}$ (LISP IPv4-in-IPv4)

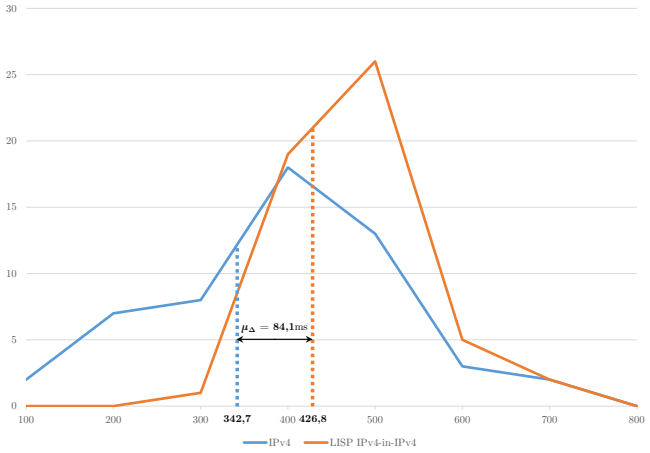

Fig. 2. Frequency Distribution of $\bar{\mathbf{x}}$ (IPv4) and $\bar{\mathbf{y}}$ (LISP IPv4-in-IPv4)

A. Delay

As we have $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ which are univariate discrete random variables, it is possible to determine their expected values $\mu_{\bar{\mathbf{x}}}$ and $\mu_{\bar{\mathbf{y}}}$ (arithmetic average) and their standard deviations $\sigma_{\bar{\mathbf{x}}}$ and $\sigma_{\bar{\mathbf{y}}}$ by the common formulas. The results are reported in Table III along with the percentage of lost packets and the expected value $\mu_{\bar{\mathbf{\Delta}}}$ (again the arithmetic average) and the standard deviation $\sigma_{\bar{\mathbf{\Delta}}}$. The mean value $\mu_{\bar{\mathbf{\Delta}}}$ represents exactly the LISP delay as the average time difference between whether using LISP or not, that is 84.1 ms. The frequency distribution chart of $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ is reported in Fig. 2.

The motivations behind LISP delay are various. The LISP decoupling of host identity from its location information is achieved by replacing IP addresses with two separate name

	μ (ms)	σ (ms)	λ (%)
$\bar{\mathbf{x}}$	342.7	130.1	1.2
$\bar{\mathbf{y}}$	426.8	84.6	7.0
$\bar{\mathbf{\Delta}}$	84.1	89.7	

TABLE III. Comparison of IPv4 and LISP IPv4-in-IPv4 performance

$\bar{\mathbf{x}}$: IPv4, $\bar{\mathbf{y}}$: LISP IPv4-in-IPv4.
μ: mean value, σ: standard deviation, λ: percentage of packet loss

spaces: Endpoint IDentifier (EID), and Routing LOCator (RLOC). Apps bind to host's EID, which is used as the address (either IPv4 or IPv6) for transport connections, while RLOCs are the addresses (either IPv4 or IPv6) used for routing through transit networks [22].

In IPv4, the inner ICMP packet is $64\mathrm{B}$, while the IPv4 header is $20\mathrm{B}$, for a total size of $84\mathrm{B}$. In LISP, instead, the tunneling requires a change into the IPv4 header where the IP addresses are replaced with the source and destination EIDs, and moreover there is an overhead which consists of $8\mathrm{B}$ of LISP header, $8\mathrm{B}$ of UDP header, and other $20\mathrm{B}$ of IPv4 header where the IP addresses are the source and destination RLOCs, for a total size of $120\mathrm{B}$. Obviously this overhead itself generated the growth of the network latency, but also traffic encapsulation and decapsulation. On the other hand, the RLOCs' improved handling of routing tables and the consequent optimized paths, plus other fine-tuning techniques into the RLOC space, can considerably decrease the network latency.

B. Speed and Precision

Other parameters like the average traffic speeds $v_{\bar{\mathbf{x}}}$ and $v_{\bar{\mathbf{y}}}$ are defined similarly as above depending on the situation where either the whole data "*in-the-wire*" (w) or only the payload (p) is considered.

	$\boldsymbol{w}$ (B)	$\boldsymbol{v_w}$ (B/s)	$\boldsymbol{p}$ (B)	$\boldsymbol{v_p}$ (B/s)
$\bar{\mathbf{x}}$	84	245.1	54	157.6
$\bar{\mathbf{y}}$	120	281.2	54	126.5

TABLE IV. IPv4 versus LISP speeds

$\bar{\mathbf{x}}$: IPv4, $\bar{\mathbf{y}}$: LISP IPv4-in-IPv4.
$\boldsymbol{w}$: packet size "*in-the-wire*", $\boldsymbol{p}$: payload, $\boldsymbol{v}$: average speed

The results reported in Table IV are relevant and partially unexpected: even if there is a considerable delay between LISP and standard IP, the effective bandwidth experienced by the whole data "*in-the-wire*" using LISP is greater. Instead, if the speed is measured in terms of payload only (what really perceived and experienced by final users), LISP's speed performance suffers an acceptable relative difference of $-19.71\,\%$. Another important consideration to make is about associated standard error and deviation whose results are reported in Table V.

	$\mathbf{SE}_\mu$ (ms)	$\mathbf{RSE}_\mu$ (%)	**RSD** (%)
$\bar{\mathbf{x}}$	17.9	5.2	38.0
$\bar{\mathbf{y}}$	11.6	2.7	19.8

TABLE V. IPv4 versus LISP statistics

$\mathbf{SE}_\mu$: standard error, $\mathbf{RSE}_\mu$: relative standard error,
RSD: relative standard deviation

Therefore, we discovered that LISP has almost the *half* of the relative standard deviation in comparison with IP: this means that communications over LISP tunneling are definitely very stable and this has obvious good consequences in terms of channel throughput.

Nevertheless, probably due also to the very recent release of the exploited LISP implementation, LISP has shown to be affected by $7.0\,\%$ of packet loss, which is undoubtedly non-negligible for almost all Internet applications. It must be said, however, that in our experiments (and in the results reported above) we have considered as lost the packets with ping times greater than $800.0\,\mathrm{ms}$, as reasonable in many delay-sensitive applications. Furthermore, it must be taken into account that this value refers to ICMP ping packets exchanged without supporting/implementing any additional error-recovery strategy: hence the reported results can be considered the lowest upper bound. Still, this LISP under-performance of $7.0\,\%$ is too far from the most usual IP $1.2\,\%$ to take into consideration the adoption of LISP on a world-wide scale very soon.

V. Conclusive Remarks and Future Work

This paper aims at presenting a concise and fresh overview of the state-of-the-art of the available implementations for multihoming support, with specific focus on how to exploit multihoming functionality over Android devices. We have concentrated on how to efficiently, effectively, and feasibly exploit LISP-like supports on Android smartphones, with no need of rooted-device administration rights, in order to be reasonably usable by a large public of final users. The paper originally reports about this practical experience and some related performance results.

The experience made and the reported quantitative results show that, in general, LISP is now a reasonable multihoming solution that achieves a good tradeoff between performance and rapid deployability. LISPmob, as an Android-oriented implementation of the LISP approach, has demonstrated to be mature enough to call for limited configuration work by final users, even if some of its performance results have demonstrated to be still a non-negligible limit for its widespread utilization. Nevertheless, against an average delay of $84.1\,\mathrm{ms}$ and a relative difference of speeds of $-19.71\,\%$ if compared with standard IPv4, the advantages brought out by LISP, in terms of multihoming capabilities as well as of network scalability, are undoubtedly relevant.

Based on the encouraging results already achieved, we are working to proceed with this research activity along a few primary directions. On the one hand, so far we have tested the LISP solution in IPv4-in-IPv4 contexts: we have recently started to work on validating and evaluating the performance of the same approach in the IPv6-in-IPv6 case, as well as in the other possible combinations of deployment environments. On the other hand, rooting or not rooting dilemma, so far we have chosen not to take into consideration rooted Android devices as the primary targets; however, for a complete performance evaluation and comparison, it could be relevant to report also the rooted implementation of LISPmob in different usage scenarios and deployment environments. Integrated multihoming management tools, capable of dynamically considering the interworking of rooted and non-rooted devices with their differentiated multihoming support implementations, is part of our medium-term objectives.

REFERENCES

[1] R. Praveen, J. Prashant, and R. Hemant, "A Multihoming solution for medium sized enterprises," International Institute of Information Technology, Hyderabad, November 2005.

[2] M. Hoefling, M. Menth, and M. Hartmann, "A Survey of Mapping Systems for Locator/Identifier Split Internet Routing," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1842–1858, Fourth 2013.

[3] S. Haeri, R. Gill, M. Hay, T. Wong, and L. Trajkovic, "Multihoming with Locator/ID Separation Protocol: An Experimental Testbed," February 2015.

[4] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar, "Making Use of All the Networks Around Us: A Case Study in Android," *CellNet'12*, August 2013.

[5] A. Dhraief and A. Belghith, "Multihoming support in the Internet: A state of the art," in *MICS 2010: International Conference on Models of Information and Communication Systems*, Rabat, Morocco, November 2010.

[6] T. Aura, P. Nikander, and G. Camarillo, "Effects of Mobility and Multihoming on Transport-Protocol Security," in *S&P 2004: IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2004, pp. 12–26.

[7] B. M. Sousa, K. Pentikousis, and M. Curado, "Multihoming Management for Future Networks," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 505–517, August 2011.

[8] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol," RFC 3963 (Proposed Standard), Internet Engineering Task Force, Jan. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3963.txt

[9] K. Shima, Y. Uo, N. Ogashiwa, and S. Uda, "Operational Experiment of Seamless Handover of a Mobile Router using Multiple Care-of Address Registration," *Journal of Networks*, vol. 1, no. 3, July 2006.

[10] K. Shima, K. Mitsuya, R. Wakikawa, T. Momose, and K. Uehara, "SHISA: The Mobile IPv6/NEMO BS Stack Implementation Current Status," in *Asia BSD Conference 2007*, March 2007.

[11] N. Veiga, M. Antunes, V. Santos, and A. Santos, "Experiments with IPv6 Network Mobility Using NEMO Protocol," in *IADIS International Telecommunications, Networks and Systems*, 2007.

[12] H. Naderi and B. E. Carpenter, "A Review of IPv6 Multihoming Solutions," in *ICN 2011, 10th International Conference on Networks*, 2011.

[13] P. Conrad, G. Heinz, J. Caro, A.L., P. Amer, and J. Fiore, "SCTP in Battlefield Networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, October 2001, pp. 289–295 vol.1.

[14] A. Kanungo, T. Janani, and P. Narayanasamy, "Dynamic IP reconfiguration in Stream Control Transmission Protocol," in *IEEE TENCON 2003*, 2003.

[15] M. Watari, A. Tagami, and S. Ano, "Evaluating the Performance of Locator/ID Separation Based on LISP Map Cache Emulation," in *Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on*, July 2012, pp. 296–301.

[16] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill, "IPv4 Multihoming Practices and Limitations," RFC 4116 (Informational), Internet Engineering Task Force, July 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4116.txt

[17] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol," RFC 5201 (Experimental), Internet Engineering Task Force, April 2008, updated by RFC 6253. [Online]. Available: http://www.ietf.org/rfc/rfc5201.txt

[18] E. Nordmark and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6," RFC 5533 (Proposed Standard), Internet Engineering Task Force, June 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5533.txt

[19] R. J. Atkinson and S. N. Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description," RFC 6740 (Experimental), Internet Engineering Task Force, November 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6740.txt

[20] Google Inc. (2013, December) ConnectivityManager. Android APIs Reference. [Online]. Available: http://developer.android.com/reference/android/net/ConnectivityManager.html

[21] F. Hoguet, "Network mobility for multi-homed Android mobile devices," Master's thesis, Nicta – Université de Technologie de Compiègne, Eveleigh, Sydney, NSW, Australia – Compiègne, France, September 2012.

[22] A. Rodríguez-Natal, L. Jakab, M. Portolés, V. Ermagan, P. Natarajan, F. Maino, D. Meyer, and A. Cabellos-Aparicio, "LISPmob: Mobile Networking through LISP," *Proceedings of Springer Wireless Personal Communications Journal*, 2012.